

Principio di base dell'affidabilità



Principio di base dell'affidabilità: Framework AWS Well-Architected

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Riassunto e introduzione	1
Introduzione	1
Affidabilità	3
Modello di responsabilità condivisa per la resilienza	3
Principi di progettazione	7
Definizioni	8
La resilienza e i componenti dell'affidabilità	8
Zona di disponibilità	9
Obiettivi di ripristino di emergenza (DR)	13
Approfondimento sulle esigenze di disponibilità	14
Fondamenti	16
Gestione di quote e vincoli di servizio	16
REL01-BP01 Consapevolezza su quote e vincoli di servizio	17
REL01-BP02 Gestione delle quote di servizio in più account e regioni	22
REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura	26
REL01-BP04 Monitoraggio e gestione delle quote	30
REL01-BP05 Automazione della gestione delle quote	34
REL01-BP06 Creazione di un divario sufficiente tra le quote attuali e l'utilizzo massimo per consentire eventuali failover	36
Pianificazione della topologia di rete	40
REL02-BP01 Utilizzo di una connettività di rete a disponibilità elevata per gli endpoint pubblici del carico di lavoro	41
REL02-BP02 Esecuzione del provisioning di connettività ridondante tra reti private nel cloud e negli ambienti on-premise.	46
REL02-BP03 Verifica che l'allocazione delle sottoreti IP consenta l'espansione e la disponibilità:	49
REL02-BP04 Preferire topologie hub-and-spoke rispetto a mesh da-molti-a-molti	52
REL02-BP05 Applicazione di intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi con indirizzi privati a cui sono connessi	54
Architettura del carico di lavoro	57
Progettazione dell'architettura del servizio di carico di lavoro	57
REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro	58
REL03-BP02 Creazione di servizi focalizzati su domini e funzionalità aziendali specifici	61
REL03-BP03 Fornitura di contratti di servizio per API	65

Progetta interazioni in un sistema distribuito per prevenire guasti	69
REL04-BP01 Identificazione del tipo di sistema distribuito da cui si dipende	69
REL04-BP02 Implementazione di dipendenze "loosely coupled"	75
REL04-BP03 Esecuzione di un lavoro costante	80
REL04-BP04 Rendere tutte le risposte idempotenti	81
Progettazione di interazioni in un sistema distribuito per mitigare o affrontare gli errori	83
REL05-BP01 Implementazione della normale riduzione delle prestazioni per trasformare le dipendenze forti applicabili in dipendenze deboli	83
REL05-BP02 Richieste di limitazione (della larghezza di banda della rete)	87
REL05-BP03 Controllo e limitazione delle chiamate di ripetizione	91
REL05-BP04 Anticipazione degli errori e limitazione delle code	95
REL05-BP05 Impostazione dei timeout dei client	98
REL05-BP06 Utilizzo dei sistemi stateless laddove possibile	102
REL05-BP07 Implementazione di leve di emergenza	104
Gestione delle modifiche	108
Monitoraggio delle risorse del carico di lavoro	108
REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro (generazione)	109
REL06-BP02 Definizione e calcolo dei parametri (aggregazione)	113
REL06-BP03 Invio di notifiche (elaborazione e avvisi in tempo reale)	114
REL06-BP04 Automatizzazione delle risposte (elaborazione e avvisi in tempo reale)	118
REL06-BP05 Analisi	121
REL06-BP06 Esecuzione di revisioni periodiche	123
REL06-BP07 Monitoraggio del tracciamento end-to-end delle richieste attraverso il sistema	125
Progettazione di un carico di lavoro in grado di adattarsi ai cambiamenti della domanda	128
REL07-BP01 Utilizzo dell'automazione per l'acquisizione o il dimensionamento delle risorse	128
REL07-BP02 Ottenimento di risorse quando viene rilevata la compromissione di un carico di lavoro	132
REL07-BP03 Ottenimento di risorse dopo aver rilevato che sono necessarie più risorse per un carico di lavoro	134
REL07-BP04 Esecuzione di un test di carico sul carico di lavoro	135
Implementazione della modifica	137
REL08-BP01 Utilizzo di runbook per attività standard come l'implementazione	138
REL08-BP02 Esecuzione di test funzionali come parte integrante dell'implementazione	139
REL08-BP03 Esecuzione di test di resilienza come parte integrante dell'implementazione ..	141

REL08-BP04 Esecuzione dell'implementazione utilizzando un'infrastruttura immutabile	143
REL08-BP05 Implementazione delle modifiche tramite automazione	148
Gestione degli errori	152
Eseguire il backup dei dati	153
REL09-BP01 Identificazione e backup di tutti i dati che richiedono un backup o riproduzione dei dati dalle origini	153
REL09-BP02 Protezione e crittografia dei backup	157
REL09-BP03 Esecuzione del backup dei dati in automatico	159
REL09-BP04 Ripristino periodico dei dati per verificare l'integrità e i processi di backup:	162
Utilizzo dell'isolamento dei guasti per proteggere il carico di lavoro	166
REL10-BP01 Implementazione del carico di lavoro in diversi luoghi	166
REL10-BP02 Selezione delle posizioni appropriate per la tua implementazione multiposizione	172
REL10-BP03 Ripristino automatico dei componenti vincolati a una singola posizione	177
REL10-BP04 Utilizzo di architetture a scomparti per limitare la portata dell'impatto	179
Progettazione di un carico di lavoro resistente agli errori dei componenti	183
REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti	184
REL11-BP02 Failover e passaggio a risorse integre	187
REL11-BP03 Automatizzazione della riparazione a tutti i livelli	191
REL11-BP04 Fare affidamento al piano dati invece che al piano di controllo durante il ripristino	195
REL11-BP05 Utilizzo della stabilità statica per evitare un comportamento bimodale	200
REL11-BP06 Invio di notifiche quando gli eventi influiscono sulla disponibilità	204
REL11-BP07 Progettazione del prodotto in modo da soddisfare gli obiettivi di disponibilità e i contratti sul livello di servizio per i tempi di attività	207
Test dell'affidabilità	210
REL12-BP01 Utilizzo dei playbook per analizzare gli errori	211
REL12-BP02 Esecuzione di analisi post-incidente	213
REL12-BP03 Test dei requisiti funzionali	216
REL12-BP04 Test dei requisiti di dimensionamento e prestazioni	217
REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos	218
REL12-BP06 Esecuzione regolare di giornate di gioco	229
Pianificazione per il disaster recovery (DR)	231
REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati	231

REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino	238
REL13-BP03 Esecuzione di test sull'implementazione del ripristino di emergenza per convalidare l'implementazione	252
REL13-BP04 Gestione della deviazione di configurazione nel sito o nella Regione del ripristino di emergenza	254
REL13-BP05 Automatizzazione del ripristino	255
Esempi di implementazioni per obiettivi di disponibilità	258
Selezione delle dipendenze	258
Scenari a regione singola	259
Scenario a due 9 (99%)	259
Scenario a tre 9 (99,9%)	261
Scenario a quattro 9 (99,99%)	265
Scenari multi-regione	268
Scenario a tre 9 e ½ (99,95%) con un tempo di ripristino compreso tra 5 e 30 minuti	269
Scenario a cinque 9 (99,999%) o superiore con un tempo di ripristino inferiore a 1 minuto ...	273
Risorse	277
Documentazione	277
I corsi	277
Collegamenti esterni	277
Libri	277
Conclusione	278
Collaboratori	279
Approfondimenti	280
Revisioni del documento	281

Pilastro dell'affidabilità – Framework AWS Well-Architected

Data di pubblicazione: 27 giugno 2024 ([Revisioni del documento](#))

Questo whitepaper tratta del principio dell'affidabilità del [Framework AWS Well-Architected](#). Fornisce istruzioni per aiutare i clienti ad applicare best practice per la progettazione, la distribuzione e la manutenzione degli ambienti Amazon Web Services (AWS).

Introduzione

Al [Framework AWS Well-Architected](#) aiuta a comprendere i pro e i contro delle decisioni prese durante la creazione dei carichi di lavoro in AWS. Utilizzando il Framework, scoprirai le best practice architetturali per progettare e gestire carichi di lavoro affidabili, sicuri, efficienti, convenienti e sostenibili nel cloud. Esso fornisce un modo per misurare coerentemente le architetture rispetto alle best practice e identificare le aree da migliorare. Riteniamo che avere un carico di lavoro ben architettato aumenti notevolmente la probabilità di successo aziendale.

Il Framework AWS Well-Architected si basa su sei pilastri di base:

- Eccellenza operativa
- Sicurezza
- Affidabilità
- Efficienza delle prestazioni
- Ottimizzazione dei costi
- Sostenibilità

Questo documento si concentra sul principio di base dell'affidabilità e su come applicarlo alle tue soluzioni. Il raggiungimento dell'affidabilità può essere difficile negli ambienti in locale tradizionali a causa di singoli punti di errore, mancanza di automazione e di elasticità. Adottando le prassi di questo documento, creerai architetture che abbiano solide basi, un'architettura resiliente, una gestione coerente delle modifiche e processi di ripristino dei guasti comprovati.

Questo documento è rivolto a chi ricopre ruoli nell'ambito della tecnologia, ad esempio ai Chief Technology Officer (CTO), ai progettisti, agli sviluppatori e ai membri dei team operativi. Dopo aver letto questo documento, comprenderai le best practice e le strategie AWS da utilizzare durante la

progettazione di architetture affidabili in un ambiente cloud. Questo documento include dettagli di implementazione di alto livello e modelli architetturali, nonché riferimenti a risorse aggiuntive.

Affidabilità

Il principio dell'affidabilità comprende la capacità di un carico di lavoro di eseguire la funzione attesa in modo corretto e coerente quando previsto. Include la possibilità di utilizzare e testare il carico di lavoro per tutto il ciclo di vita. Questo documento fornisce linee guida dettagliate sulle best practice per l'implementazione di carichi di lavoro affidabili in AWS.

Argomenti

- [Modello di responsabilità condivisa per la resilienza](#)
- [Principi di progettazione](#)
- [Definizioni](#)
- [Approfondimento sulle esigenze di disponibilità](#)

Modello di responsabilità condivisa per la resilienza

La resilienza è una responsabilità condivisa tra te e AWS. È importante comprendere, nell'ambito della resilienza, il funzionamento del ripristino di emergenza e della disponibilità in questo modello condiviso.

Responsabilità di AWS: resilienza del cloud

AWS è responsabile della protezione dell'infrastruttura su cui vengono eseguiti tutti i servizi offerti nel Cloud AWS. Questa infrastruttura include l'hardware, il software, le reti e le strutture su cui vengono eseguiti i servizi Cloud AWS. AWS prende iniziative commercialmente sensate per rendere disponibili questi servizi Cloud AWS, garantendo che la disponibilità soddisfi o superi i [contratti sul livello di servizio AWS](#).

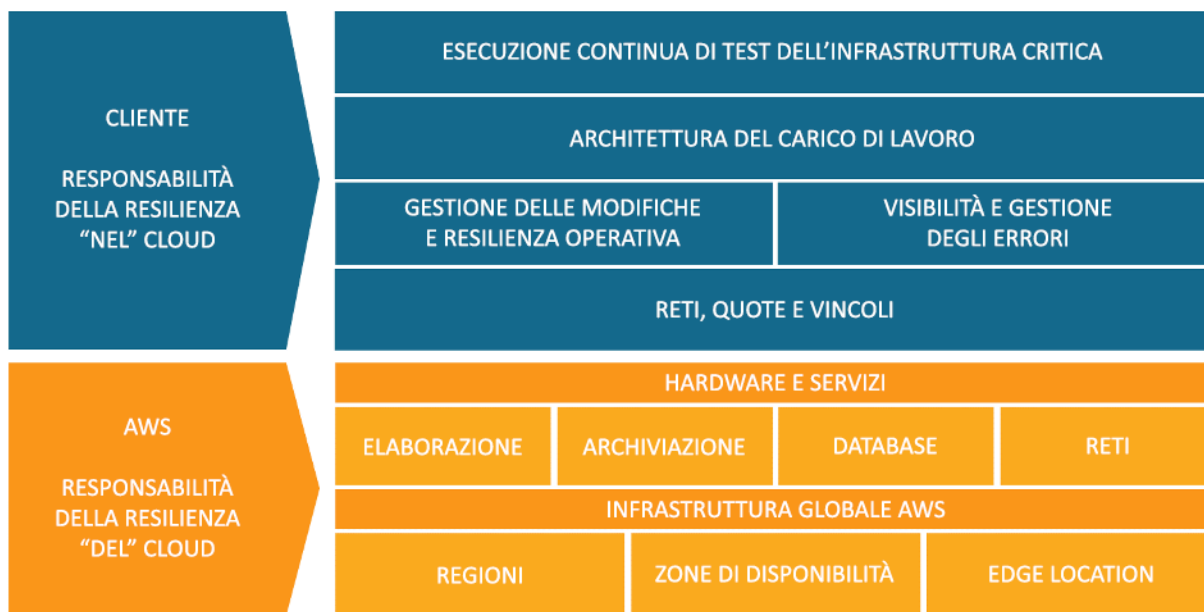
L'[infrastruttura cloud globale di AWS](#) è progettata per permettere ai clienti di creare architetture dei carichi di lavoro altamente resilienti. Ogni Regione AWS è completamente isolata e costituita da più [zone di disponibilità](#), che sono partizioni fisicamente isolate dell'infrastruttura. Le zone di disponibilità isolano gli errori che potrebbero influire sulla resilienza del carico di lavoro, impedendo loro di interessare altre zone nella regione. Allo stesso tempo, tutte le zone in una Regione AWS sono interconnesse con reti a larghezza di banda elevata e a bassa latenza, su una fibra ottica metropolitana dedicata e completamente ridondante che fornisce connettività ad alta velocità di trasmissione effettiva e a bassa latenza tra zone. Tutto il traffico tra zone è crittografato. Le prestazioni di rete sono sufficienti per eseguire la replica sincrona tra zone. Se un'applicazione viene

partizionata tra zone di disponibilità, le aziende sono isolate e protette meglio da problemi come interruzioni dell'alimentazione, fulmini, tornado, uragani e altro ancora.

Responsabilità del cliente: resilienza nel cloud

La tua responsabilità è determinata dai servizi Cloud AWS che scegli. La scelta definisce la quantità di attività di configurazione che devi eseguire nell'ambito delle tue responsabilità verso la resilienza. Ad esempio, un servizio come Amazon Elastic Compute Cloud (Amazon EC2) richiede che il cliente esegua tutte le attività di configurazione e gestione della resilienza necessarie. I clienti che implementano istanze Amazon EC2 sono responsabili dell'[implementazione di istanze Amazon EC2 in più posizioni](#) (ad esempio, zone di disponibilità AWS), dell'[implementazione della riparazione automatica](#) usando servizi come Auto Scaling e dell'uso di [best practice per un'architettura del carico di lavoro resiliente](#) per le applicazioni installate nelle istanze. Per i servizi gestiti come Amazon S3 e Amazon DynamoDB, AWS si occupa del livello dell'infrastruttura, del sistema operativo e delle piattaforme, mentre i clienti accedono agli endpoint per archiviare e recuperare i dati. Tu hai la responsabilità della gestione della resilienza dei dati, incluse le strategie di backup, controllo delle versioni e replica.

L'implementazione del carico di lavoro in più zone di disponibilità in una Regione AWS è parte di una strategia di disponibilità elevata progettata per proteggere i carichi di lavoro isolando i problemi in una zona di disponibilità, usando la ridondanza delle altre zone di disponibilità per continuare a gestire le richieste. Un'architettura multi-AZ è parte anche di una strategia di ripristino di emergenza progettata per isolare e proteggere meglio i carichi di lavoro da problemi come le interruzioni dell'alimentazione, i fulmini, i tornado, i terremoti e altri ancora. Le strategie di ripristino di emergenza possono usare anche più Regioni AWS. Ad esempio, in una configurazione con approccio attivo/passivo, il servizio per il carico di lavoro esegue il failover dalla regione attiva alla regione di ripristino di emergenza se la regione attiva non può più gestire le richieste.



Responsabilità della resilienza nel e del cloud per i clienti e AWS.

Puoi usare servizi AWS per realizzare gli obiettivi di resilienza. Come cliente, sei responsabile della gestione degli aspetti seguenti del sistema per realizzare la resilienza nel cloud. Per ulteriori informazioni su ogni servizio specifico, consulta la [documentazione AWS](#).

Reti, quote e vincoli

- Le best practice per questa area del Modello di responsabilità condivisa vengono descritte in [Concetti fondamentali](#).
- Pianifica l'architettura con spazio adeguato per la scalabilità e identifica le [quote di servizio](#) e i vincoli dei servizi scelti, in base all'aumento previsto delle richieste di caricamento laddove applicabile.
- Progetta la [topologia di rete](#) in modo che sia altamente disponibile, ridondante e scalabile.

Gestione delle modifiche e resilienza operativa

- La [gestione delle modifiche](#) include come introdurre e gestire le modifiche nell'ambiente. L'[implementazione di modifiche](#) richiede la creazione e l'aggiornamento di runbook e strategie di implementazione per l'applicazione e l'infrastruttura.
- Una strategia resiliente per il [monitoraggio delle risorse del carico di lavoro](#) tiene conto di tutti i componenti, incluse metriche tecniche e aziendali, notifiche, automazione e analisi.

- I carichi di lavoro nel cloud devono [adattarsi alle variazioni della domanda](#), dimensionandosi in risposta ai problemi o alle fluttuazioni di utilizzo.

Visibilità e gestione degli errori

- La visibilità sugli errori tramite il monitoraggio è necessaria per automatizzare la riparazione, in modo che i carichi di lavoro possano [resistere a errori dei componenti](#).
- La [gestione degli errori](#) richiede il [backup dei dati](#), l'applicazione delle best practice per permettere al carico di lavoro di resistere a errori dei componenti e la [pianificazione per il ripristino di emergenza](#).

Architettura del carico di lavoro

- L'[architettura del carico di lavoro](#) include il modo in cui progetti servizi attorno a diversi ambiti aziendali, applichi la progettazione di architetture orientate ai servizi (SOA) e sistemi distribuiti per impedire gli errori e crei funzionalità come la limitazione (della larghezza di banda della rete), la ripetizione di tentativi, la gestione delle code, i timeout e le leve di emergenza.
- Usa [soluzioni AWS](#) collaudate, la [Liberia dei costruttori di Amazon](#) e [modelli serverless](#) per applicare le best practice e avviare rapidamente le implementazioni.
- Usa miglioramenti continui per scomporre il sistema in servizi distribuiti per una scalabilità e un'innovazione più rapide. Usa linee guida per i [microservizi AWS](#) e opzioni per servizi gestiti per semplificare e accelerare l'introduzione di modifiche e favorire l'innovazione.

Esecuzione continua di test dell'infrastruttura critica

- L'[esecuzione di test dell'affidabilità](#) significa testare l'infrastruttura a livello funzionale, di prestazioni e di chaos engineering, nonché adottare procedure di analisi degli eventi imprevisti e simulazione per sviluppare esperienza nel risolvere problemi poco compresi.
- Per applicazioni interamente nel cloud e ibride, la conoscenza del loro comportamento quando si verificano problemi o in caso di arresto dei componenti permette di recuperare rapidamente e in modo affidabile dalle interruzioni.
- Crea e documenta esperimenti ripetibili per identificare il comportamento del sistema in situazioni impreviste. Questi test dimostreranno l'efficacia della resilienza complessiva e forniranno un ciclo di feedback per le procedure operative prima di affrontare scenari di errore reali.

Principi di progettazione

Nel cloud, sono presenti una serie di principi che possono aiutarti ad aumentare l'affidabilità. Tieni presente quanto segue quando si discute delle best practice:

- Ripristino automatico in caso di guasto: monitorando un carico di lavoro in base agli indicatori chiave di prestazioni (KPI), puoi attivare l'automazione al superamento di una soglia. Questi KPI dovrebbero essere una misura del valore aziendale, non degli aspetti tecnici del funzionamento del servizio. Ciò consente la notifica e il tracciamento automatici degli errori e i processi di recupero automatizzati che aggirano o riparano l'errore. Con un'automazione più sofisticata è possibile anticipare e correggere gli errori prima che si verifichino.
- Test delle procedure di ripristino: in un ambiente on-premise vengono spesso eseguiti test per dimostrare che il carico di lavoro funzionerà in uno scenario specifico. I test non vengono generalmente utilizzati per convalidare le strategie di recupero. Nel cloud, puoi testare il modo in cui il carico di lavoro incorre nell'errore e convalidare le procedure di ripristino. È possibile utilizzare l'automazione per simulare diversi errori o per ricreare scenari che in precedenza hanno portato a errori. Questo approccio presenta percorsi di errore che puoi testare e correggere prima che si verifichi uno scenario di errore reale, riducendo il rischio.
- Dimensionamento orizzontale per aumentare la disponibilità del carico di lavoro aggregato: sostituisci una risorsa di grandi dimensioni con più risorse più piccole per ridurre l'impatto di un singolo errore sul carico di lavoro complessivo. Distribuisci le richieste su molteplici risorse più piccole per garantire che non condividano un punto di errore comune.
- Eliminazione delle congetture sulla capacità: una causa comune di errori in carichi di lavoro on-premise è la saturazione delle risorse, quando le richieste a cui è sottoposto il carico di lavoro ne superano la capacità (questo è spesso l'obiettivo degli attacchi di tipo Denial of Service). Nel cloud, è possibile monitorare la domanda e l'utilizzo dei carichi di lavoro, nonché automatizzare l'aggiunta o la rimozione di risorse per mantenere il livello ottimale, al fine di soddisfare la domanda senza un provisioning eccessivo o inferiore. Esistono ancora dei limiti, ma alcune quote possono essere controllate e altre possono essere gestite (consulta [Gestione di quote e vincoli di servizio](#)).
- Gestione delle modifiche tramite l'automazione: le modifiche all'infrastruttura devono essere apportate usando l'automazione. Le modifiche che devono essere gestite includono le modifiche all'automazione, che possono quindi essere monitorate e revisionate.

Definizioni

Questo whitepaper si occupa dell'affidabilità nel cloud, illustrando le best practice per queste quattro aree:

- Fondamenti
- Architettura del carico di lavoro
- Gestione delle modifiche
- Gestione degli errori

Per ottenere affidabilità, è necessario iniziare dalle basi: un ambiente in cui le quote di servizio e la topologia di rete sono in grado di supportare il carico di lavoro. L'architettura del carico di lavoro del sistema distribuito deve essere progettata per prevenire e mitigare gli errori. Il carico di lavoro deve gestire le variazioni nella domanda o nei requisiti e deve essere progettato per rilevare l'errore e correggersi automaticamente.

Argomenti

- [La resilienza e i componenti dell'affidabilità](#)
- [Disponibilità](#)
- [Obiettivi di ripristino di emergenza \(DR\)](#)

La resilienza e i componenti dell'affidabilità

L'affidabilità di un carico di lavoro nel cloud dipende da diversi fattori, il principale dei quali è la resilienza:

- La resilienza è la capacità di un carico di lavoro di ripristinarsi in seguito a interruzioni dell'infrastruttura o del servizio, acquisire dinamicamente risorse di calcolo per soddisfare la domanda e mitigare le interruzioni, ad esempio dovute a configurazioni errate o problemi di rete temporanei.

Gli altri fattori che influiscono sull'affidabilità del carico di lavoro sono:

- Eccellenza operativa, che include l'automazione delle modifiche, l'uso di playbook per rispondere ai guasti e le valutazioni di prontezza operativa (ORR) per confermare che le applicazioni sono pronte per le operazioni di produzione.

- Sicurezza, che include la prevenzione di danni ai dati o all'infrastruttura da parte di malintenzionati, che comprometterebbero la disponibilità. Ad esempio, crittografare i backup per garantire la sicurezza dei dati.
- Efficienza delle prestazioni, che include la progettazione di tassi massimi di richiesta e la riduzione al minimo delle latenze per il carico di lavoro.
- Ottimizzazione dei costi, che include compromessi quali spendere di più sulle istanze EC2 per ottenere stabilità statica oppure fare affidamento sulla scalabilità automatica quando è necessaria una maggiore capacità.

La resilienza è l'obiettivo principale di questo whitepaper.

Anche gli altri quattro aspetti sono importanti e vengono presentati nei rispettivi principi del [Framework AWS Well-Architected](#). Molte delle best practice qui presenti affrontano anche gli aspetti relativi all'affidabilità, ma l'attenzione è focalizzata sulla resilienza.

Disponibilità

La disponibilità, nota anche come disponibilità del servizio, è sia una metrica comunemente usata per misurare quantitativamente la resilienza sia un obiettivo di resilienza.

- La disponibilità è la percentuale di tempo in cui un carico di lavoro è disponibile per l'uso.

Disponibile per l'uso significa che il carico di lavoro esegue correttamente la funzione concordata quando necessario.

Questa percentuale viene calcolata su un periodo di tempo, ad esempio un mese, un anno o un tre anni consecutivi. Applicando l'interpretazione più rigida possibile, la disponibilità viene ridotta ogni volta che l'applicazione non funziona normalmente, incluse le interruzioni pianificate e non pianificate. Definiamo la disponibilità in questo modo:

$$\textit{Disponibilità} = \frac{\textit{Tempo di disponibilità per l'uso}}{\textit{Tempo totale}}$$

- La disponibilità è una percentuale di uptime (come il 99,9%) per un periodo di tempo (normalmente un mese o un anno)

- Un modo di dire comune si riferisce solo al "numero di nove", ad esempio "cinque nove" significa una disponibilità del 99,999%
- Alcuni clienti scelgono di escludere i tempi di inattività pianificati del servizio (ad esempio, la manutenzione programmata) dal tempo totale nella formula. Tuttavia, questo approccio non è consigliato, poiché gli utenti probabilmente vorranno usare il tuo servizio in quei momenti.

Ecco una tabella per gli obiettivi di progettazione di disponibilità delle applicazioni comuni e il periodo di tempo massimo durante il quale le interruzioni possono verificarsi entro un anno pur raggiungendo l'obiettivo. La tabella contiene esempi dei tipi di applicazioni che comunemente vediamo ad ogni livello di disponibilità. In questo documento, ci riferiamo a questi valori.

Disponibilità	Indisponibilità massima (per anno)	Categorie dell'applicazione
<u>99%</u>	3 giorni 15 ore	Elaborazione batch, estrazioni e dati, trasferimento e caricamento di lavori
<u>99,9%</u>	8 ore 45 minuti	Strumenti interni come knowledge management, monitoraggio dei progetti
<u>99,95%</u>	4 ore 22 minuti	Commercio online, POS
<u>99,99%</u>	52 minuti	Trasmissione di video, carichi di lavoro di trasmissione
<u>99,999%</u>	5 minuti	Transazioni bancomat, carichi di lavoro di telecomunicazione

Misurazione della disponibilità in base alle richieste. Per il tuo servizio potrebbe essere più facile conteggiare le richieste fallite e quelle andate a buon fine, invece del "tempo disponibile per l'utilizzo". In questo caso può essere utilizzato il seguente calcolo:

$$\text{Disponibilità} = \frac{\text{Risposte corrette}}{\text{Richieste valide}}$$

Questo viene spesso misurato per periodi di un minuto o di cinque minuti. Quindi è possibile calcolare una percentuale di tempo di attività mensile (misurazione della disponibilità in base al tempo) dalla media di questi periodi. Se non vengono ricevute richieste in un dato periodo, viene conteggiato come disponibile al 100% per quel periodo.

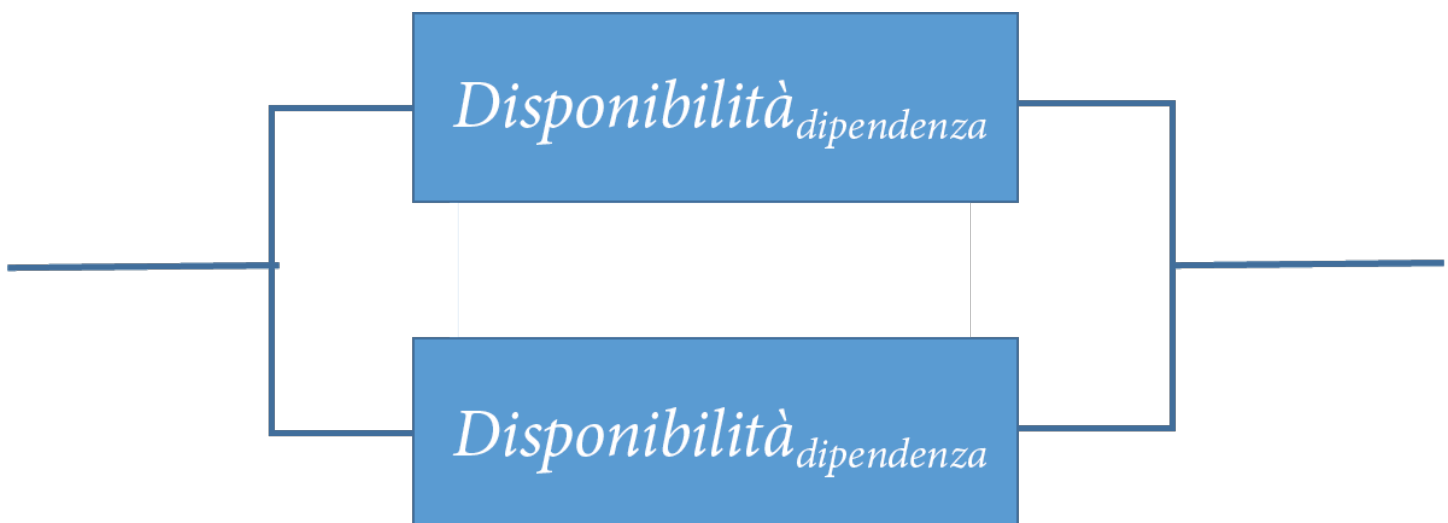
Calcolo della disponibilità con forti dipendenze. Molti sistemi hanno forti dipendenze da altri sistemi, nei quali un'interruzione in un sistema dipendente si traduce direttamente in un'interruzione del sistema di invocazione. Ciò si contrappone a una dipendenza leggera, in cui un errore del sistema dipendente viene compensato nell'applicazione. Laddove esistono tali forti dipendenze, la disponibilità del sistema di invocazione è il prodotto delle disponibilità dei sistemi dipendenti. Ad esempio, se disponi di un sistema progettato per una disponibilità del 99,99% che ha una forte dipendenza da due altri sistemi indipendenti, ciascuno progettato per una disponibilità del 99,99%, il carico di lavoro può teoricamente raggiungere il 99,97% di disponibilità:

$$\text{Disponibilità}_{\text{invocazione}} \times \text{disponibilità}_{\text{dip1}} \times \text{disponibilità}_{\text{dip2}} = \text{Avail}_{\text{carico di lavoro}}$$

$$99,99\% \times 99,99\% \times 99,99\% = 99,97\%$$

È quindi importante comprendere le tue dipendenze e i loro obiettivi di progettazione della disponibilità mentre calcoli i tuoi.

Calcolo della disponibilità con componenti ridondanti. Quando un sistema prevede l'uso di componenti indipendenti e ridondanti (ad esempio, risorse ridondanti in diverse zone di disponibilità), la disponibilità teorica viene calcolata come 100% meno il prodotto delle percentuali di errore dei componenti. Ad esempio, se un sistema utilizza due componenti indipendenti, ciascuno con una disponibilità del 99,9%, la disponibilità effettiva di questa dipendenza è del 99,9999%:



$$\text{Disponibilità}_{\text{convenienza}} = \text{Disponibilità}_{\text{MAX}} - ((100\% - \text{Avail}_{\text{dipendenza}}) \times (100\% - \text{Avail}_{\text{dipendenza}}))$$

$$99.9999\% = 100\% - (0.1\% \times 0.1\%)$$

Calcolo rapido: se le disponibilità di tutti i componenti inclusi nel calcolo consiste esclusivamente nella cifra nove, puoi sommare il totale del numero di cifre nove per ottenere la risposta. Nell'esempio precedente, due componenti indipendenti ridondanti con disponibilità a tre nove totalizzano sei nove.

Calcolo della disponibilità delle dipendenze. Alcune dipendenze forniscono linee guida sulla loro disponibilità, inclusi gli obiettivi di progettazione della disponibilità per molti servizi AWS. Tuttavia, nei casi in cui ciò non è possibile, ad esempio un componente in cui il produttore non pubblica informazioni sulla disponibilità, un modo per effettuare una stima è determinare il tempo medio tra guasti (MTBF) e il tempo medio di ripristino (MTTR). Una stima della disponibilità può essere stabilita da:

$$\text{Disponibilità}_{\text{STIMATA}} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

Ad esempio, se l'MTBF è di 150 giorni e l'MTTR è di 1 ora, la stima della disponibilità è del 99,97%.

Per ulteriori informazioni, consulta [Disponibilità e oltre: comprensione e miglioramento della resilienza di sistemi distribuiti su AWS](#), che fornisce informazioni utili per calcolare la disponibilità.

Costi di disponibilità. La progettazione di applicazioni per livelli più elevati di disponibilità comporta in genere un aumento dei costi, perciò è opportuno identificare le reali esigenze di disponibilità prima di iniziare la progettazione dell'applicazione. Elevati livelli di disponibilità impongono requisiti più severi per i test e la convalida in scenari di errore esaustivi. Questi richiedono l'automazione per il ripristino da tutti i tipi di guasti e richiedono che tutti gli aspetti delle operazioni di sistema siano costruiti e testati in modo simile secondo gli stessi standard. Ad esempio, l'aggiunta o la rimozione di capacità, la distribuzione o il rollback del software aggiornato o le modifiche alla configurazione o la migrazione dei dati di sistema devono essere condotti all'interno dell'obiettivo di disponibilità desiderato. Combinando i costi per lo sviluppo del software, a livelli molto elevati di disponibilità, l'innovazione soffre a causa della necessità di muoversi più lentamente nella distribuzione dei sistemi. Il suggerimento, pertanto, è di essere rigorosi nell'applicazione degli standard e nel considerare l'obiettivo di disponibilità appropriato per l'intero ciclo di vita del funzionamento del sistema.

Un altro modo in cui i costi aumentano nei sistemi che operano con obiettivi di progettazione di disponibilità più elevata è la selezione delle dipendenze. A questi obiettivi più elevati, il set di software o servizi che possono essere scelti come dipendenze diminuisce in base a quali di questi servizi hanno avuto i grandi investimenti che abbiamo descritto in precedenza. Man mano che l'obiettivo di progettazione della disponibilità aumenta, è tipico trovare meno servizi multiuso (ad esempio un database relazionale) e più servizi dedicati. Questo perché questi ultimi sono più facili da valutare, testare e automatizzare e hanno un potenziale ridotto di interazioni imprevedibili con funzionalità incluse, ma inutilizzate.

Obiettivi di ripristino di emergenza (DR)

Oltre agli obiettivi di disponibilità, la strategia di resilienza deve includere obiettivi di ripristino di emergenza (DR) basati su strategie per recuperare il carico di lavoro in caso di un evento di fallimento. Il ripristino di emergenza è incentrato su obiettivi di ripristino una tantum in risposta a disastri naturali, errori tecnici su larga scala o minacce umane, come attacchi o errori. Si tratta di un parametro diverso rispetto alla disponibilità che misura la resilienza su un periodo di tempo in risposta a fallimenti di componenti, picchi di caricamenti o bug di software.

Obiettivo del tempo di ripristino (RTO). Definito dall'organizzazione. L'RTO è il ritardo massimo accettabile tra l'interruzione del servizio ed il suo ripristino. Questo determina ciò che viene considerato un intervallo di tempo accettabile quando il servizio non è disponibile.

Obiettivo del punto di ripristino (RPO). Definito dall'organizzazione. L'RPO è il periodo di tempo massimo accettabile dall'ultimo punto di ripristino dei dati. Questo determina ciò che viene considerato una perdita di dati accettabile tra l'ultimo punto di ripristino e l'interruzione del servizio.

Continuità aziendale

Quanti dati puoi permetterti di ricreare o perdere?

**Quanto velocemente riesci a eseguire il recupero?
Qual è il costo dell'indisponibilità del servizio?**



Relazione tra RPO (Recovery Point Objective), RTO (Recovery Time Objective) e l'evento di emergenza.

L'RTO è simile all'MTTR (tempo medio di ripristino), in quanto entrambi misurano il tempo tra l'inizio di un'interruzione e il ripristino del carico di lavoro. Tuttavia, l'MTTR è un valore medio acquisito in occasione di diversi eventi che hanno impatto sulla disponibilità in un certo periodo di tempo, mentre l'RTO è un obiettivo, o un valore massimo consentito, per un singolo evento che ha impatto sulla disponibilità.

Approfondimento sulle esigenze di disponibilità

Di solito si pensa inizialmente alla disponibilità di un'applicazione come a un singolo obiettivo per l'applicazione nel suo insieme. Tuttavia, con un'analisi più approfondita, riscontriamo spesso che alcuni aspetti di un'applicazione o di un servizio hanno requisiti di disponibilità diversi. Ad esempio, alcuni sistemi potrebbero dare priorità alla capacità di ricevere e archiviare nuovi dati prima del recupero dei dati esistenti. Altri sistemi danno la priorità alle operazioni in tempo reale rispetto alle operazioni che cambiano la configurazione o l'ambiente di un sistema. I servizi potrebbero avere requisiti di disponibilità molto elevati durante determinate ore del giorno, ma possono tollerare periodi di interruzione molto più lunghi al di fuori di questi orari. Questi sono alcuni dei modi in cui è possibile scomporre una singola applicazione in parti costitutive e valutare i requisiti di disponibilità per ciascuna. Il vantaggio che ne trai è di concentrare i tuoi sforzi (e le spese) sulla disponibilità in base a esigenze specifiche, piuttosto che progettare l'intero sistema in base ai requisiti più severi.

Consiglio

Valuta criticamente gli aspetti unici delle tue applicazioni e, dove necessario, differenzia gli obiettivi di progettazione della disponibilità e del ripristino di emergenza per riflettere le esigenze della tua azienda.

All'interno di AWS, dividiamo comunemente i servizi in "piano dati" e "piano di controllo". Il piano dei dati è responsabile della fornitura del servizio in tempo reale mentre i piani di controllo vengono utilizzati per configurare l'ambiente. Ad esempio, le istanze Amazon EC2, i database Amazon RDS e le operazioni di lettura/scrittura delle tabelle di Amazon DynamoDB sono tutte operazioni sul piano dei dati. Al contrario, l'avvio di nuove istanze EC2 o database RDS o l'aggiunta o la modifica di metadati delle tabelle di DynamoDB sono tutte considerate operazioni sul piano di controllo. Sebbene alti livelli di disponibilità siano importanti per tutte queste funzionalità, i piani dei dati hanno generalmente obiettivi di progettazione di disponibilità più elevati rispetto ai piani di controllo. Pertanto, i carichi di lavoro con requisiti di alta disponibilità dovrebbero evitare dipendenze di runtime su operazioni di piano di controllo.

Molti clienti AWS adottano un approccio analogo alla valutazione critica delle loro applicazioni e all'identificazione di componenti secondari con diverse esigenze di disponibilità. Gli obiettivi di progettazione della disponibilità vengono quindi adattati ai diversi aspetti e vengono compiuti gli appropriati sforzi di lavoro per progettare il sistema. AWS ha un'esperienza significativa nella progettazione di applicazioni con una serie di obiettivi di progettazione della disponibilità, inclusi servizi con una disponibilità del 99,999% o superiore. AWS Solution Architects (SA) possono aiutarti a progettare in modo appropriato in base ai tuoi obiettivi di disponibilità. Il coinvolgimento di AWS nelle prime fasi del processo di progettazione migliora la nostra capacità di aiutarti a raggiungere i tuoi obiettivi di disponibilità. La pianificazione della disponibilità non viene eseguita solo prima dell'avvio del carico di lavoro. Viene anche eseguita continuamente per perfezionare la progettazione man mano che acquisisci esperienza operativa, impari da eventi reali e superi errori di diversi tipi. È quindi possibile applicare lo sforzo di lavoro appropriato per migliorare l'implementazione.

Le esigenze di disponibilità necessarie per un carico di lavoro devono essere allineate alle esigenze e alla criticità aziendali. Definendo innanzitutto il framework di criticità aziendale con RTO, RPO e disponibilità definiti, è possibile valutare ogni carico di lavoro. Tale approccio richiede che le persone coinvolte nell'implementazione del carico di lavoro siano informate del framework e dell'impatto che il loro carico di lavoro esercita sulle esigenze aziendali.

Fondamenti

I requisiti di base sono quelli il cui ambito si estende oltre un singolo carico di lavoro o progetto. Prima di progettare qualsiasi sistema, devono essere stabiliti i requisiti fondamentali che influenzano l'affidabilità. Ad esempio, è necessario disporre di una larghezza di banda di rete sufficiente verso il data center.

In un ambiente locale, questi requisiti possono causare tempi di consegna lunghi a causa delle dipendenze e pertanto devono essere incorporati durante la pianificazione iniziale. Con AWS, tuttavia, la maggior parte di questi requisiti di base è già incorporata o può essere affrontata in base alle esigenze. Il cloud è progettato per essere quasi illimitato, perciò è responsabilità di AWS soddisfare i requisiti di capacità di rete e di elaborazione sufficienti, lasciandoti libero di modificare le dimensioni delle risorse e le allocazioni on demand.

Nelle sezioni seguenti vengono illustrate le best practice incentrate su queste considerazioni per l'affidabilità.

Argomenti

- [Gestione di quote e vincoli di servizio](#)
- [Pianificazione della topologia di rete](#)

Gestione di quote e vincoli di servizio

Per le architetture di carichi di lavoro basate sul cloud, esistono quote di servizio (definite anche come restrizioni dei servizi). Queste quote sono presenti per evitare di effettuare accidentalmente il provisioning di più risorse di quelle necessarie e limitare i tassi di richiesta sulle operazioni API in modo da proteggere i servizi da un uso illecito. Esistono anche vincoli di risorse, ad esempio la velocità con cui è possibile trasferire i bit su un cavo in fibra ottica o la quantità di storage su un disco fisico.

Se utilizzi applicazioni AWS Marketplace, devi comprendere le limitazioni di tali applicazioni. Se utilizzi servizi Web o Software as a Service di terze parti, devi conoscere anche tali restrizioni.

Best practice

- [REL01-BP01 Consapevolezza su quote e vincoli di servizio](#)
- [REL01-BP02 Gestione delle quote di servizio in più account e regioni](#)

- [REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura](#)
- [REL01-BP04 Monitoraggio e gestione delle quote](#)
- [REL01-BP05 Automazione della gestione delle quote](#)
- [REL01-BP06 Creazione di un divario sufficiente tra le quote attuali e l'utilizzo massimo per consentire eventuali failover](#)

REL01-BP01 Consapevolezza su quote e vincoli di servizio

Conosci le quote predefinite e gestisci le richieste di aumento delle quote per l'architettura del carico di lavoro. Sai quali vincoli delle risorse cloud, ad esempio disco o rete, sono potenzialmente influenti.

Risultato desiderato: i clienti possono evitare il degrado dei servizi o l'interruzione in Account AWS implementando linee guida appropriate per il monitoraggio di metriche chiave, revisioni dell'infrastruttura e fasi di correzione dell'automazione per verificare che non vengano raggiunte quote e vincoli dei servizi che potrebbero causare degrado o interruzione del servizio.

Anti-pattern comuni:

- Distribuzione di un carico di lavoro senza comprendere le quote hard o soft e i relativi limiti per i servizi utilizzati.
- Distribuzione di un carico di lavoro sostitutivo senza analizzare e riconfigurare le quote necessarie o contattare preventivamente l'assistenza.
- Supposizione che i servizi cloud non abbiano limiti e che i servizi possano essere utilizzati senza tener conto di tariffe, limiti, conteggi, quantità.
- Supposizione che le quote verranno aumentate automaticamente.
- Mancata conoscenza del processo e della scadenza delle richieste di quote.
- Supposizione che la quota predefinita del servizio cloud sia identica per ogni servizio rispetto alle varie regioni.
- Supposizione che i vincoli del servizio possano essere violati e che i sistemi si autoscalino o aumentino il limite oltre i vincoli della risorsa
- Nessun test dell'applicazione nei momenti di picco del traffico, per stressare l'utilizzo delle sue risorse.
- Provisioning della risorsa senza analisi della dimensione della risorsa richiesta.
- Provisioning in eccesso di capacità scegliendo tipi di risorse che vanno ben oltre il fabbisogno effettivo o i picchi previsti.

- Nessuna valutazione dei requisiti di capacità per nuovi livelli di traffico prima di un nuovo evento cliente o dell'implementazione di una nuova tecnologia.

Vantaggi derivanti dall'adozione di questa best practice: monitoraggio e gestione automatizzata delle quote di servizio e limiti delle risorse possono ridurre i guasti in modo proattivo. I cambiamenti nei modelli di traffico per il servizio di un cliente possono causare un'interruzione o un degrado se non si seguono le best practice. Monitorando e gestendo questi valori in tutte le regioni e in tutti gli account, le applicazioni possono avere una maggiore resilienza in caso di eventi avversi o non pianificati.

Livello di rischio associato se questa best practice non fosse adottata: Elevato

Guida all'implementazione

Service Quotas è un servizio AWS che ti aiuta a gestire le quote per oltre 250 servizi AWS da un'unica posizione. Oltre a cercare i valori delle quote, si possono anche richiedere e monitorare gli aumenti delle quote stesse tramite la console Service Quotas o tramite l'SDK AWS. AWS Trusted Advisor offre un controllo delle quote di servizio che mostra l'utilizzo e le quote per alcuni aspetti di determinati servizi. Le quote di servizio predefinite per servizio sono indicate anche nella documentazione AWS di ciascun servizio (ad esempio vedi [Quote di Amazon VPC](#)).

Alcuni limiti dei servizi, come i limiti di velocità sulle API con throttling vengono impostati all'interno di Amazon API Gateway stesso configurando un piano di utilizzo. Altri limiti impostati come configurazione per i rispettivi servizi includono capacità di IOPS allocata, archiviazione Amazon RDS allocato e allocazioni di volumi Amazon EBS. Amazon Elastic Compute Cloud dispone di un proprio pannello di controllo sui limiti del servizio che consente di gestire l'istanza, Amazon Elastic Block Store e i limiti degli indirizzi IP elastici. Se hai un caso d'uso in cui le quote di servizio influiscono sulle prestazioni della tua applicazione e non sono adattabili alle tue esigenze, contatta AWS Support per vedere se sono possibili riduzioni.

Le quote di servizio possono essere specifiche per ogni regione o di natura globale. L'uso di un servizio AWS che raggiunge la sua quota non si comporterà come previsto nell'uso normale e potrebbe causare interruzioni o degrado del servizio. Ad esempio, una quota di servizio limita il numero di DL Amazon EC2 che può essere usato in una Regione e tale limite può essere raggiunto durante un evento di dimensionamento del traffico tramite gruppi Auto Scaling (ASG).

Le quote di servizio per ogni account devono essere valutate regolarmente per determinare quali siano i limiti di servizio appropriati per quell'account. Queste quote di servizio esistono come guardrail operativi, per evitare di fornire accidentalmente più risorse di quelle necessarie. Servono anche a limitare i tassi di richiesta delle operazioni API per proteggere i servizi dagli abusi.

I limiti dei servizi sono diversi dalle quote dei servizi. I vincoli di servizio rappresentano i limiti di una particolare risorsa, definiti da quel tipo di risorsa. Questi possono essere la capacità di archiviazione (ad esempio, gp2 ha un limite di dimensione di 1 GB - 16 TB) o il throughput del disco (10.000 iops). È essenziale che il vincolo di un tipo di risorsa sia progettato e valutato costantemente per l'utilizzo che potrebbe raggiungere il suo limite. Se un vincolo viene raggiunto inaspettatamente, le applicazioni o i servizi dell'account possono essere degradati o interrotti.

Se hai un caso d'uso in cui le quote di servizio influiscono sulle prestazioni della tua applicazione e non sono adattabili alle tue esigenze, contatta AWS Support per vedere se sono possibili mitigazioni. Per maggiori dettagli su come modificare le quote fisse vedi [REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura](#).

Esistono alcuni servizi e strumenti AWS per monitorare e gestire Service Quotas. Il servizio e gli strumenti devono essere sfruttati per fornire controlli automatici o manuali dei livelli di quota.

- AWS Trusted Advisor offre un controllo delle quote di servizio che mostra l'utilizzo e le quote per alcuni aspetti di alcuni servizi. Può aiutare a identificare i servizi vicini alle quote.
- AWS Management Console fornisce metodi per visualizzare i valori delle quote dei servizi, gestire, richiedere nuove quote, monitorare lo stato delle richieste di quote e visualizzare la cronologia delle quote.
- AWS CLI e CDK offrono metodi programmatici per gestire e monitorare automaticamente l'utilizzo e i livelli delle quote di servizio.

Passaggi dell'implementazione

Per Service Quotas:

- [Revisione di AWS Service Quotas](#).
- Per essere certo delle quote di servizio esistenti, stabilisci i servizi (come IAM Access Analyzer) usati. Esistono circa 250 servizi AWS controllati da quote di servizio. Quindi stabilisci il nome della quota di servizio specifica che potrebbe essere usata all'interno di ogni account e regione. Esistono circa 3000 nomi di quote di servizio per regione.
- Aumenta questa analisi delle quote con AWS Config per trovare tutte le [risorse AWS](#) usate nei tuoi Account AWS.
- Utilizza i [dati AWS CloudFormation](#) per stabilire le risorse AWS utilizzate. Esamina le risorse create in AWS Management Console o con il comando [list-stack-resources](#) AWS CLI. È anche possibile vedere le risorse configurate da distribuire nel modello stesso.

- Stabilisci tutti i servizi necessari per il tuo carico di lavoro analizzando il codice di implementazione.
- Determina le quote di servizio applicabili. Utilizza le informazioni accessibili in modo programmatico da Trusted Advisor e Service Quotas.
- Stabilisci un metodo di monitoraggio automatizzato (vedi [REL01-BP02 Gestione delle quote di servizio in più account e regioni](#) e [REL01-BP04 Monitoraggio e gestione delle quote](#)) per avvisare e informare se le quote di servizio sono vicine o hanno raggiunto il limite.
- Stabilisci un metodo automatico e programmatico per verificare se una quota di servizio è stata modificata in una regione ma non in altre regioni dello stesso account. (consulta [REL01-BP02 Gestione delle quote di servizio in più account e regioni](#) e [REL01-BP04 Monitoraggio e gestione delle quote](#)).
- Automatizza la scansione dei log e delle metriche delle applicazioni per determinare se ci sono errori di quota o di vincoli di servizio. Se sono presenti errori, invia gli allarmi al sistema di monitoraggio.
- Stabilisci procedure di progettazione per calcolare la modifica richiesta nella quota (vedi [REL01-BP05 Automazione della gestione delle quote](#)) una volta stabilito che per alcuni servizi specifici sono richieste quote maggiori.
- Crea un flusso di lavoro di provisioning e di approvazione per richiedere modifiche alla quota di servizio. Questo dovrebbe includere un flusso di lavoro di eccezione in caso di rifiuto della richiesta o di approvazione parziale.
- Crea un metodo ingegneristico per rivedere le quote dei servizi prima del provisioning e dell'utilizzo di nuovi servizi AWS prima del roll-out in ambienti di produzione o carichi (ad esempio, account di test di carico).

Per i vincoli dei servizi:

- Stabilisci metodi di monitoraggio e metrica per avvisare se le risorse si avvicinano ai loro limiti. Sfrutta CloudWatch in base alle necessità per le metriche o il monitoraggio dei log.
- Stabilisci soglie di allarme per ogni risorsa che ha un vincolo significativo per l'applicazione o il sistema.
- Crea procedure di gestione del flusso di lavoro e dell'infrastruttura per cambiare il tipo di risorsa se il vincolo è prossimo all'utilizzo. Questo flusso di lavoro dovrebbe includere test di carico come best practice per verificare che quello nuovo sia il tipo di risorsa corretto in base ai nuovi vincoli.
- Migra la risorsa identificata al nuovo tipo di risorsa consigliato, utilizzando le procedure e i processi esistenti.

Risorse

Best practice correlate:

- [REL01-BP02 Gestione delle quote di servizio in più account e regioni](#)
- [REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura](#)
- [REL01-BP04 Monitoraggio e gestione delle quote](#)
- [REL01-BP05 Automazione della gestione delle quote](#)
- [REL01-BP06 Creazione di un divario sufficiente tra le quote attuali e l'utilizzo massimo per consentire eventuali failover](#)
- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)
- [REL10-BP01 Implementazione del carico di lavoro in diversi luoghi](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)
- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)
- [REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos](#)

Documenti correlati:

- [Principio dell'affidabilità di AWS Well-Architected Framework: disponibilità](#)
- [AWS Service Quotas \(precedentemente note come restrizioni dei servizi\)](#)
- [Controlli delle best practice AWS Trusted Advisor \(consulta la sezione Restrizioni dei servizi\)](#)
- [Monitoraggio limite su AWS su risposte AWS](#)
- [Quote di servizio Amazon EC2](#)
- [Che cos'è Service Quotas?](#)
- [Come richiedere un aumento delle quote](#)
- [Endpoint e quote dei servizi](#)
- [Guida per l'utente di Service Quotas](#)
- [Monitoraggio delle quote per AWS](#)
- [Limiti di isolamento dei guasti di AWS](#)
- [Disponibilità con ridondanza](#)
- [AWS for Data](#)
- [In cosa consiste l'Integrazione continua?](#)
- [In cosa consiste la Distribuzione continua?](#)

- [Partner APN: partner per la gestione della configurazione](#)
- [Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS](#)
- [Gestione e monitoraggio della limitazione delle API nei carichi di lavoro](#)
- [Visualizza i suggerimenti di AWS Trusted Advisor su scala con AWS Organizations](#)
- [Automazione dell'aumento dei limiti di servizio e supporto aziendale con AWS Control Tower](#)

Video correlati:

- [AWS Live re:Inforce 2019 - Service Quotas](#)
- [Visualizza e gestisci quote per i servizi AWS con Service Quotas](#)
- [Demo delle quote AWS IAM](#)

Strumenti correlati:

- [Amazon CodeGuru Reviewer](#)
- [AWS CodeDeploy](#)
- [AWS CloudTrail](#)
- [Amazon CloudWatch](#)
- [Amazon EventBridge](#)
- [Amazon DevOps Guru](#)
- [AWS Config](#)
- [AWS Trusted Advisor](#)
- [AWS CDK](#)
- [AWS Systems Manager](#)
- [Marketplace AWS](#)

REL01-BP02 Gestione delle quote di servizio in più account e regioni

Se utilizzi più account o Regioni, assicurati di richiedere le quote appropriate in tutti gli ambienti in cui vengono eseguiti i carichi di lavoro di produzione.

Risultato desiderato: i servizi e le applicazioni non dovrebbero essere interessati dall'esaurimento delle quote di servizio per le configurazioni che si estendono su account o Regioni o con progetti di resilienza che utilizzano il failover di zona, Regione o account.

Anti-pattern comuni:

- Consentire l'aumento dell'utilizzo delle risorse in una Regione di isolamento senza alcun meccanismo per mantenere la capacità nelle altre.
- Impostare manualmente tutte le quote in modo indipendente nelle Regioni di isolamento.
- Non considerare l'effetto delle architetture di resilienza (come quelle attive o passive) nelle future esigenze di quote durante un degrado nella Regione non primaria.
- Non valutare regolarmente le quote e apportare le modifiche necessarie in ogni Regione e account in cui viene gestito il carico di lavoro.
- Non sfruttare [modelli di richiesta delle quote](#) per richiedere incrementi su più Regioni e account.
- Non aggiornare le quote dei servizi, perché si pensa erroneamente che l'aumento delle quote abbia implicazioni di costo, come le richieste di prenotazione di calcolo.

Vantaggi derivanti dall'adozione di questa best practice: verificare che sia possibile gestire il proprio carico attuale in Regioni o account secondari in caso di mancata disponibilità dei servizi regionali. Questo consente di ridurre il numero di errori o livelli di degrado che si verificano durante la perdita di Regioni.

Livello di rischio associato se questa best practice non fosse adottata: Elevato

Guida all'implementazione

Le quote di servizio vengono monitorate per account. Salvo diversa indicazione, ogni quota è specifica della Regione AWS. Oltre agli ambienti di produzione, gestisci anche le quote in tutti gli ambienti non di produzione applicabili, in modo che i test e lo sviluppo non siano ostacolati. Il mantenimento di un elevato grado di resilienza richiede una valutazione continua delle quote di servizio (sia automatica che manuale).

Con più carichi di lavoro in diverse Regioni a causa dell'implementazione di progetti che usano approcci Active/Active, Active/Passive – Hot, Active/Passive-Cold e Active/Passive-Pilot Light, è fondamentale comprendere tutti i livelli di quote di account e Regioni. I modelli di traffico passati non sono sempre un buon indicatore per stabilire se la quota di servizio è impostata correttamente.

Altrettanto importante è il fatto che il limite di nome della quota di servizio non è sempre lo stesso per ogni Regione. In una Regione il valore potrebbe essere cinque, in un'altra potrebbe essere dieci. La gestione di queste quote deve riguardare tutti gli stessi servizi, account e Regioni per garantire una resilienza costante sotto carico.

Riconciliare tutte le differenze di quota di servizio tra le diverse Regioni (Regione attiva o passiva) e creare processi per riconciliare continuamente queste differenze. I piani di test dei failover passivi delle Regioni sono raramente scalati in base alla capacità attiva di picco, il che significa che gli esercizi di game day o table top possono non riuscire a trovare le differenze nelle quote di servizio tra le Regioni e a mantenere i limiti corretti.

Deviazione delle quote di servizio, è molto importante da monitorare e valutare la condizione in cui i limiti delle quote di servizio per una specifica quota nominata vengono modificati in una Regione e non in tutte le Regioni. Si dovrebbe prendere in considerazione la possibilità di modificare la quota nelle Regioni con traffico o potenzialmente in grado di trasportare traffico.

- Seleziona gli account e le regioni pertinenti in base ai tuoi requisiti di servizio, di latenza, normativi e di ripristino di emergenza.
- Identifica le quote dei servizi per tutti gli account, le regioni e le zone di disponibilità pertinenti. Le restrizioni si riferiscono ad account e regione. Questi valori devono essere confrontati per far emergere le differenze.

Passaggi dell'implementazione

- Rivedi i valori Service Quotas che potrebbero aver superato il livello di rischio di utilizzo. AWS Trusted Advisor offre allarmi per la violazione di soglie dell'80% e del 90%.
- Rivedi i valori per le quote di servizio in qualsiasi Regione Passiva (in un progetto Attivo/Passivo). Verifica che il carico venga eseguito correttamente nelle Regioni secondarie in caso di guasto nella Regione primaria.
- Valuta automaticamente se si è verificata una deviazione delle quote di servizio tra le Regioni dello stesso account e agisci di conseguenza per modificare i limiti.
- Se le Unità Organizzative (UO) del cliente sono strutturate nel modo supportato, i modelli di quote di servizio devono essere aggiornati per riflettere le modifiche alle quote da applicare a più Regioni e account.
 - Crea un modello e associa le Regioni alla modifica della quota.
 - Rivedi tutti i modelli delle quote di servizio esistenti per qualsiasi modifica richiesta (Regione, limiti e account).

Risorse

Best practice correlate:

- [REL01-BP01 Consapevolezza su quote e vincoli di servizio](#)
- [REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura](#)
- [REL01-BP04 Monitoraggio e gestione delle quote](#)
- [REL01-BP05 Automazione della gestione delle quote](#)
- [REL01-BP06 Creazione di un divario sufficiente tra le quote attuali e l'utilizzo massimo per consentire eventuali failover](#)
- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)
- [REL10-BP01 Implementazione del carico di lavoro in diversi luoghi](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)
- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)
- [REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos](#)

Documenti correlati:

- [Principio dell'affidabilità di AWS Well-Architected Framework: disponibilità](#)
- [AWS Service Quotas \(precedentemente note come restrizioni dei servizi\)](#)
- [Controlli delle best practice AWS Trusted Advisor \(consulta la sezione Restrizioni dei servizi\)](#)
- [Monitoraggio limite su AWS su risposte AWS](#)
- [Quote di servizio Amazon EC2](#)
- [Che cos'è Service Quotas?](#)
- [Come richiedere un aumento delle quote](#)
- [Endpoint e quote dei servizi](#)
- [Guida per l'utente di Service Quotas](#)
- [Monitoraggio delle quote per AWS](#)
- [Limiti di isolamento dei guasti di AWS](#)
- [Disponibilità con ridondanza](#)
- [AWS for Data](#)
- [In cosa consiste l'Integrazione continua?](#)
- [In cosa consiste la Distribuzione continua?](#)
- [Partner APN: partner per la gestione della configurazione](#)
- [Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS](#)

- [Gestione e monitoraggio della limitazione delle API nei carichi di lavoro](#)
- [Visualizza i suggerimenti di AWS Trusted Advisor su scala con AWS Organizations](#)
- [Automazione dell'aumento dei limiti di servizio e supporto aziendale con AWS Control Tower](#)

Video correlati:

- [AWS Live re:Inforce 2019 - Service Quotas](#)
- [Visualizza e gestisci quote per i servizi AWS con Service Quotas](#)
- [Demo delle quote AWS IAM](#)

Servizi correlati:

- [Amazon CodeGuru Reviewer](#)
- [AWS CodeDeploy](#)
- [AWS CloudTrail](#)
- [Amazon CloudWatch](#)
- [Amazon EventBridge](#)
- [Amazon DevOps Guru](#)
- [AWS Config](#)
- [AWS Trusted Advisor](#)
- [AWS CDK](#)
- [AWS Systems Manager](#)
- [Marketplace AWS](#)

REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura

Identifica attentamente quote di servizio, vincoli del servizio e limiti delle risorse fisiche che non possono essere modificati. Progetta architetture per le applicazioni e i servizi in modo da impedire che questi limiti abbiano impatto sull'affidabilità.

Alcuni esempi includono la larghezza di banda di rete, le dimensioni di payload delle chiamate di funzioni serverless, il tasso di espansione dei limiti per un gateway API e le connessioni utente simultanee a un database.

Risultato desiderato: l'applicazione o il servizio ha le prestazioni previste in condizioni di traffico normale ed elevato. L'applicazione o il servizio è stato progettato per operare entro i limiti dei vincoli o delle quote di servizio fissi della risorsa.

Anti-pattern comuni:

- Scelta di una progettazione che usa una risorsa di un servizio, senza essere al corrente della presenza di vincoli che causeranno errori di progettazione durante il dimensionamento.
- Esecuzione di benchmark poco realistici e che raggiungono le quote di servizio fisse durante i test. Ad esempio, l'esecuzione di test a un limite di espansione per un periodo di tempo prolungato.
- Scelta di una progettazione che non può essere dimensionata o modificata in caso di superamento delle quote di servizio fisse. Ad esempio, dimensioni dei payload SQS di 256 KB.
- Mancata progettazione e implementazione della visibilità per monitorare e segnalare le soglie per le quote di servizio a rischio durante eventi di traffico elevato.

Vantaggi dell'adozione di questa best practice: possibilità di verificare che l'applicazione verrà eseguita a tutti i livelli di carico dei servizi previsti senza interruzioni o errori.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

Diversamente dalle risorse e dalle quote di servizio flessibili che possono essere sostituite con unità di capacità maggiori, le quote di servizio fisse in AWS non possono essere modificate. Di conseguenza, tutti i servizi AWS di questo tipo devono essere valutati per identificare i possibili limiti fissi di capacità quando vengono usati per la progettazione di un'applicazione.

I limiti fissi vengono mostrati nella console Service Quotas. Se le colonne indicano REGOLABILE = No, il servizio ha un limite fisso. I limiti fissi vengono mostrati anche in alcune pagine di configurazione delle risorse. Ad esempio, per Lambda è previsto un limite fisso specifico che non può essere modificato.

Ad esempio, durante la progettazione di un'applicazione Python da eseguire in una funzione Lambda, l'applicazione deve essere valutata per determinare la probabilità che Lambda venga eseguito per più di 15 minuti. Se il codice potrebbe restare in esecuzione oltre questo limite della quota di servizio, devi prendere in considerazione tecnologie o progettazioni alternative. Se il limite viene raggiunto dopo l'implementazione nell'ambiente di produzione, l'applicazione sarà soggetta a errori o

interruzioni finché non viene corretta. Diversamente dalle quote flessibili, non esiste alcun metodo per modificare i limiti, anche in caso di eventi di emergenza con livello di gravità 1.

Dopo aver implementato l'applicazione in un ambiente di test, è necessario adottare una strategia per determinare se vi sia la probabilità di raggiungere i limiti fissi. I test di stress, di carico e di chaos engineering devono fare parte del piano di test iniziale.

Passaggi dell'implementazione

- Esamina l'elenco completo dei servizi AWS che possono essere usati nella fase di progettazione dell'applicazione.
- Esamina i limiti di quota flessibili e fissi per tutti i servizi. Non tutti i limiti vengono indicati nella console Service Quotas. Alcuni servizi [descrivono questi limiti in posizioni diverse](#).
- Nel progettare l'applicazione, esamina i principali fattori commerciali e tecnologici del carico di lavoro, come risultati aziendali, casi d'uso, sistemi dipendenti, obiettivi di disponibilità e oggetti di ripristino di emergenza. Fai in modo che siano questi fattori commerciali e tecnologici a orientare il processo di identificazione del sistema distribuito corretto per il carico di lavoro.
- Analizza il carico dei servizi tra regioni e account. Molti limiti fissi per i servizi variano a seconda della regione. Tuttavia, alcuni limiti dipendono dagli account.
- Analizza le architetture di resilienza per l'utilizzo delle risorse durante un guasto a livello di zona e di regione. Nel corso dello sviluppo di progettazioni multi-regione che usano approcci attivo/attivo, attivo/passivo con standby a caldo, attivo/passivo con standby a freddo e attivo/passivo con Pilot Light i casi di errore determineranno un utilizzo più elevato. Questo comportamento crea un possibile caso d'uso per il raggiungimento dei limiti fissi.

Risorse

Best practice correlate:

- [REL01-BP01 Consapevolezza su quote e vincoli di servizio](#)
- [REL01-BP02 Gestione delle quote di servizio in più account e regioni](#)
- [REL01-BP04 Monitoraggio e gestione delle quote](#)
- [REL01-BP05 Automazione della gestione delle quote](#)
- [REL01-BP06 Creazione di un divario sufficiente tra le quote attuali e l'utilizzo massimo per consentire eventuali failover](#)
- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)

- [REL10-BP01 Implementazione del carico di lavoro in diversi luoghi](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)
- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)
- [REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos](#)

Documenti correlati:

- [Principio dell'affidabilità di AWS Well-Architected Framework: disponibilità](#)
- [AWS Service Quotas \(precedentemente note come restrizioni dei servizi\)](#)
- [Controlli delle best practice AWS Trusted Advisor \(consulta la sezione Restrizioni dei servizi\)](#)
- [Monitoraggio limite su AWS su risposte AWS](#)
- [Quote di servizio Amazon EC2](#)
- [Che cos'è Service Quotas?](#)
- [Come richiedere un aumento delle quote](#)
- [Endpoint e quote dei servizi](#)
- [Guida per l'utente di Service Quotas](#)
- [Monitoraggio delle quote per AWS](#)
- [Limiti di isolamento dei guasti di AWS](#)
- [Disponibilità con ridondanza](#)
- [AWS for Data](#)
- [In cosa consiste l'Integrazione continua?](#)
- [In cosa consiste la Distribuzione continua?](#)
- [Partner APN: partner per la gestione della configurazione](#)
- [Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS](#)
- [Gestione e monitoraggio della limitazione delle API nei carichi di lavoro](#)
- [Visualizza i suggerimenti di AWS Trusted Advisor su scala con AWS Organizations](#)
- [Automazione dell'aumento dei limiti di servizio e supporto aziendale con AWS Control Tower](#)
- [Operazioni, risorse e chiavi di condizione per Service Quotas](#)

Video correlati:

- [AWS Live re:Inforce 2019 - Service Quotas](#)
- [Visualizza e gestisci quote per i servizi AWS con Service Quotas](#)
- [Demo delle quote AWS IAM](#)
- [AWS re:Invent 2018: Cicli chiusi e menti aperte: come assumere il controllo di sistemi grandi e piccoli](#)

Strumenti correlati:

- [AWS CodeDeploy](#)
- [AWS CloudTrail](#)
- [Amazon CloudWatch](#)
- [Amazon EventBridge](#)
- [Amazon DevOps Guru](#)
- [AWS Config](#)
- [AWS Trusted Advisor](#)
- [AWS CDK](#)
- [AWS Systems Manager](#)
- [Marketplace AWS](#)

REL01-BP04 Monitoraggio e gestione delle quote

Valuta il tuo utilizzo potenziale e aumenta le quote in modo appropriato per una crescita pianificata dell'utilizzo.

Risultato desiderato: implementazione di sistemi attivi e automatici per la gestione e il monitoraggio. Queste soluzioni operative indicano che le soglie di utilizzo delle quote stanno per essere raggiunte. Questo problema può essere risolto in modo proattivo tramite modifiche alle quote richieste.

Anti-pattern comuni:

- Mancata configurazione del monitoraggio per verificare le soglie delle quote di servizio.
- Mancata configurazione del monitoraggio dei limiti fissi, anche se i valori non possono essere modificati.

- Valutazione errata della quantità di tempo necessaria per richiedere e ottenere la modifica di una quota flessibile, supponendo che sia immediata o rapida.
- Configurazione di allarmi per l'avvicinamento alle quote di servizio, ma senza alcun processo di risposta a un avviso.
- Configurazione di allarmi solo per i servizi supportati da AWS Service Quotas, senza monitorare altri servizi AWS.
- Valutazione errata della gestione delle quote per progettazioni di resilienza in più regioni, come gli approcci attivo/attivo, attivo/passivo con standby a caldo, attivo/passivo con standby a freddo e attivo/passivo con Pilot Light.
- Mancata valutazione delle differenze di quota tra regioni.
- Mancata valutazione delle esigenze in ogni regione per una richiesta di aumento di quota specifica.
- Mancato utilizzo di [modelli per la gestione delle quote in più regioni](#).

Vantaggi dell'adozione di questa best practice: il monitoraggio automatico di AWS Service Quotas e il monitoraggio dell'utilizzo rispetto alle quote permettono di identificare l'avvicinamento a un limite di quota. Puoi usare questi dati di monitoraggio per limitare eventuali errori dovuti all'esaurimento della quota.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

Per i servizi supportati, puoi monitorare le quote configurando servizi diversi in grado di eseguire una valutazione e quindi inviare avvisi o allarmi. In questo modo, il monitoraggio dell'utilizzo è più semplice e puoi ricevere avvisi all'avvicinamento delle quote. Gli allarmi possono essere attivati da AWS Config, funzioni Lambda, Amazon CloudWatch o AWS Trusted Advisor. Puoi anche usare filtri delle metriche in file di log CloudWatch per cercare ed estrarre modelli nei log, in modo da determinare se l'utilizzo si avvicina alle soglie delle quote.

Passaggi dell'implementazione

Per il monitoraggio:

- Acquisisci informazioni sull'attuale consumo di risorse, ad esempio bucket o istanze. Usa operazioni API dei servizi come l'API Amazon EC2 `DescribeInstances` per raccogliere informazioni sull'attuale consumo di risorse.
- Acquisisci le attuali quote essenziali e valide per i servizi usando:

- AWS Service Quotas
- AWS Trusted Advisor
- Documentazione di AWS
- Pagine specifiche dei servizi AWS
- AWS Command Line Interface (AWS CLI)
- AWS Cloud Development Kit (AWS CDK)
- Usa AWS Service Quotas, un servizio AWS che semplifica la gestione delle quote per oltre 250 servizi AWS da un'unica posizione.
- Usa i limiti del servizio Trusted Advisor per monitorare gli attuali limiti del servizio a soglie diverse.
- Usa la cronologia delle quote di servizio (console o AWS CLI) per verificare gli aumenti regionali.
- Confronta la modifica delle quote di servizio in ogni regione e ogni account per creare equivalenze, se necessario.

Per la gestione:

- Automatica: configura una regola AWS Config personalizzata per analizzare le quote di servizio tra regioni e confrontarle per individuare le differenze.
- Automatica: configura una regola Lambda personalizzata per analizzare le quote di servizio tra regioni e confrontarle per individuare le differenze.
- Manuale: analizza le quote di servizio tramite l'AWS CLI, l'API o la console AWS per esaminare le quote nelle diverse regioni e confrontarle per individuare le differenze. Segnala le differenze.
- Se vengono identificate differenze nelle quote tra regioni, richiedi una modifica della quota, se necessario.
- Esamina il risultato di tutte le richieste.

Risorse

Best practice correlate:

- [REL01-BP01 Consapevolezza su quote e vincoli di servizio](#)
- [REL01-BP02 Gestione delle quote di servizio in più account e regioni](#)
- [REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura](#)
- [REL01-BP05 Automazione della gestione delle quote](#)

- [REL01-BP06 Creazione di un divario sufficiente tra le quote attuali e l'utilizzo massimo per consentire eventuali failover](#)
- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)
- [REL10-BP01 Implementazione del carico di lavoro in diversi luoghi](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)
- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)
- [REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos](#)

Documenti correlati:

- [Principio dell'affidabilità di AWS Well-Architected Framework: disponibilità](#)
- [AWS Service Quotas \(precedentemente note come restrizioni dei servizi\)](#)
- [Controlli delle best practice AWS Trusted Advisor \(consulta la sezione Restrizioni dei servizi\)](#)
- [Monitoraggio limite su AWS su risposte AWS](#)
- [Quote di servizio Amazon EC2](#)
- [Che cos'è Service Quotas?](#)
- [Come richiedere un aumento delle quote](#)
- [Endpoint e quote dei servizi](#)
- [Guida per l'utente di Service Quotas](#)
- [Monitoraggio delle quote per AWS](#)
- [Limiti di isolamento dei guasti di AWS](#)
- [Disponibilità con ridondanza](#)
- [AWS for Data](#)
- [In cosa consiste l'Integrazione continua?](#)
- [In cosa consiste la Distribuzione continua?](#)
- [Partner APN: partner per la gestione della configurazione](#)
- [Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS](#)
- [Gestione e monitoraggio della limitazione delle API nei carichi di lavoro](#)
- [Visualizza i suggerimenti di AWS Trusted Advisor su scala con AWS Organizations](#)
- [Automazione dell'aumento dei limiti di servizio e supporto aziendale con AWS Control Tower](#)
- [Operazioni, risorse e chiavi di condizione per Service Quotas](#)

Video correlati:

- [AWS Live re:Inforce 2019 - Service Quotas](#)
- [Visualizza e gestisci quote per i servizi AWS con Service Quotas](#)
- [Demo delle quote AWS IAM](#)
- [AWS re:Invent 2018: Cicli chiusi e menti aperte: come assumere il controllo di sistemi grandi e piccoli](#)

Strumenti correlati:

- [AWS CodeDeploy](#)
- [AWS CloudTrail](#)
- [Amazon CloudWatch](#)
- [Amazon EventBridge](#)
- [Amazon DevOps Guru](#)
- [AWS Config](#)
- [AWS Trusted Advisor](#)
- [AWS CDK](#)
- [AWS Systems Manager](#)
- [Marketplace AWS](#)

REL01-BP05 Automazione della gestione delle quote

Implementa strumenti per ricevere avvisi quando le soglie stanno per essere raggiunte. Puoi automatizzare le richieste di aumento delle quote utilizzando le API AWS Service Quotas.

Se integri il tuo database di gestione della configurazione (CMDB) o il sistema di ticketing con le Service Quotas, puoi automatizzare il monitoraggio delle richieste di aumento delle quote e delle quote correnti. Oltre all'SDK AWS, Service Quotas offre automazione utilizzando AWS Command Line Interface (AWS CLI).

Anti-pattern comuni:

- Monitoraggio delle quote e dell'utilizzo nei fogli di calcolo.

- Esecuzione di report sull'utilizzo giornaliero, settimanale o mensile e successivo confronto dell'utilizzo con le quote.

Vantaggi dell'adozione di questa best practice: Il monitoraggio automatico delle quote di servizio AWS e il monitoraggio dell'utilizzo rispetto a tale quota ti consentiranno di sapere quando stai per raggiungere una quota. Puoi configurare l'automazione affinché ti aiuti a richiedere un aumento della quota quando necessario. Puoi decidere di ridurre alcune quote quando il tuo utilizzo tende alla direzione opposta per ottenere i vantaggi di riduzione del rischio (in caso di credenziali compromesse) e dei costi.

Livello di rischio associato se questa best practice non fosse adottata: Medium

Guida all'implementazione

- Impostazione del monitoraggio automatico: implementa strumenti utilizzando gli SDK per ricevere avvisi quando le soglie stanno per essere raggiunte.
 - Utilizza Service Quotas e potenzia il servizio con una soluzione di monitoraggio automatico delle quote come AWS Limit Monitor o un'offerta di Marketplace AWS.
 - [What is Service Quotas? \(Che cos'è Service Quotas?\)](#)
 - [Monitoraggio delle quota su AWS – Soluzione AWS](#)
 - Impostazione di risposte attivate in base alle soglie delle quote tramite l'utilizzo delle API di Amazon SNS e AWS Service Quotas.
 - Automazione dei test.
 - Configura le soglie delle restrizioni.
 - Integrazione con eventi di modifica di AWS Config, pipeline di implementazione, Amazon EventBridge o terze parti.
 - Imposta artificialmente soglie basse per le quote in modo da testare le risposte.
 - Configura i trigger per eseguire azioni adeguate in seguito alle notifiche e contatta AWS Support se necessario.
 - Attiva manualmente gli eventi di modifica.
 - Esegui una giornata di gioco per testare il processo di modifica dell'aumento delle quote.

Risorse

Documenti correlati:

- [Partner APN: partner per la gestione della configurazione](#)
- [Marketplace AWS: prodotti CMDB per il monitoraggio delle restrizioni](#)
- [AWS Service Quotas \(precedentemente note come restrizioni dei servizi\)](#)
- [Elenco di controllo delle best practice di AWS Trusted Advisor \(consulta la sezione Restrizioni dei servizi\)](#)
- [Monitoraggio delle quota su AWS – Soluzione AWS](#)
- [Quote di servizio di Amazon EC2](#)
- [What is Service Quotas? \(Che cos'è Service Quotas?\)](#)

Video correlati:

- [AWS Live re:Inforce 2019 - Service Quotas](#)

REL01-BP06 Creazione di un divario sufficiente tra le quote attuali e l'utilizzo massimo per consentire eventuali failover

Quando una risorsa restituisce un errore o è inaccessibile, può comunque essere conteggiata rispetto a una quota finché non viene terminata. Verifica che le quote tengano conto della sovrapposizione di risorse in errore o inaccessibili e della rispettiva sostituzione. Nel calcolare questo divario, devi considerare casi d'uso come errori di rete, regionali o delle zone di disponibilità.

Risultato desiderato: possibilità di gestire errori di piccola o grande entità relativi alle risorse o all'accessibilità delle risorse all'interno delle attuali soglie di servizio, tenendo conto degli errori delle zone, di rete o addirittura regionali nella pianificazione delle risorse.

Anti-pattern comuni:

- Impostazione delle quote di servizio in base alle esigenze attuali senza tenere conto degli scenari di failover.
- Calcolo della quota massima per un servizio senza tenere conto dei principali aspetti della stabilità statica.
- Calcolo della quota totale necessaria per ogni regione senza tenere conto delle potenziali risorse inaccessibili.
- Valutazione errata dei limiti di isolamento degli errori per alcuni servizi AWS e dei possibili modelli di utilizzo anomalo.

Vantaggi dell'adozione di questa best practice: quando eventi di interruzione dei servizi hanno impatto sulla disponibilità delle applicazioni, il cloud permette di implementare strategie per mitigare questi eventi o ripristinare i servizi. Queste strategie spesso includono la creazione di risorse aggiuntive per sostituire quelle in errore o inaccessibili. La strategia di gestione delle quote soddisferebbe queste condizioni di failover senza aggiungere altri fattori negativi dovuti al raggiungimento dei limiti dei servizi.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

Nel valutare i limiti di quota, tieni conto dei casi di failover che possono verificarsi a causa di un peggioramento della situazione. È bene considerare i tipi di casi di failover seguenti:

- Un VPC interrotto o inaccessibile.
- Una sottorete inaccessibile.
- Una zona di disponibilità sufficientemente compromessa da avere impatto sull'accessibilità di molte risorse.
- Diverse route di rete o punti di ingresso e uscita bloccati o che sono stati modificati.
- Una regione sufficientemente compromessa da avere impatto sull'accessibilità di molte risorse.
- Presenza di più risorse, ma non tutte interessate da un errore in una regione o in una zona di disponibilità.

Errori come quelli elencati sopra possono essere il fattore scatenante dell'avvio di un evento di failover. La decisione relativa all'avvio del failover è unica per ogni situazione e cliente, in quanto l'impatto aziendale può variare notevolmente. Tuttavia, nel decidere operativamente l'avvio del failover dell'applicazione o dei servizi, la pianificazione della capacità delle risorse nella posizione di failover e delle quote correlate deve essere gestita prima dell'evento.

Esamina le quote per ogni servizio tenendo conto di possibili picchi più elevati del previsto. Questi picchi possono essere correlati a risorse ancora attive raggiungibili a causa di reti o autorizzazioni. Le risorse attive non terminate continuano a essere conteggiate rispetto al limite di quota del servizio.

Passaggi dell'implementazione

- Assicurati che vi sia una differenza sufficiente tra la quota di servizio e l'utilizzo massimo in modo da gestire un failover o la perdita di accessibilità.

- Determina le quote di servizio, specificando i pattern di implementazione, i requisiti di disponibilità e la crescita dei consumi.
- Richiedi aumenti delle quote, se necessario. Pianifica tenendo conto del tempo necessario affinché le richieste di aumento delle quote siano soddisfatte.
- Determina i requisiti di affidabilità, noti anche come numero di 9.
- Determina gli scenari di errore (ad esempio, perdita di un componente, una zona di disponibilità o una regione).
- Stabilisci la metodologia di implementazione (ad esempio, canary, blu/verde, rosso/nero o rolling).
- Includi un buffer appropriato (ad esempio, 15%) rispetto alla restrizione attuale.
- Includi calcoli per la stabilità statica (zonale e regionale) laddove appropriato.
- Pianifica la crescita dei consumi (ad esempio, monitora le tendenze dei consumi).
- Tieni conto dell'impatto della stabilità statica per i carichi di lavoro più critici. Valuta la conformità delle risorse a un sistema statisticamente stabile in tutte le regioni e le zone di disponibilità.
- Valuta se usare prenotazioni della capacità on demand per pianificare la capacità in anticipo rispetto a qualsiasi failover. Questa strategia può essere utile durante le pianificazioni aziendali più critiche per ridurre i possibili rischi legati all'ottenimento della quantità e del tipo di risorse corretti durante il failover.

Risorse

Best practice correlate:

- [REL01-BP01 Consapevolezza su quote e vincoli di servizio](#)
- [REL01-BP02 Gestione delle quote di servizio in più account e regioni](#)
- [REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura](#)
- [REL01-BP04 Monitoraggio e gestione delle quote](#)
- [REL01-BP05 Automazione della gestione delle quote](#)
- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)
- [REL10-BP01 Implementazione del carico di lavoro in diversi luoghi](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)
- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)
- [REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos](#)

Documenti correlati:

- [Principio dell'affidabilità di AWS Well-Architected Framework: disponibilità](#)
- [AWS Service Quotas \(precedentemente note come restrizioni dei servizi\)](#)
- [Controlli delle best practice AWS Trusted Advisor \(consulta la sezione Restrizioni dei servizi\)](#)
- [Monitoraggio limite su AWS su risposte AWS](#)
- [Quote di servizio Amazon EC2](#)
- [Che cos'è Service Quotas?](#)
- [Come richiedere un aumento delle quote](#)
- [Endpoint e quote dei servizi](#)
- [Guida per l'utente di Service Quotas](#)
- [Monitoraggio delle quote per AWS](#)
- [Limiti di isolamento dei guasti di AWS](#)
- [Disponibilità con ridondanza](#)
- [AWS for Data](#)
- [In cosa consiste l'Integrazione continua?](#)
- [In cosa consiste la Distribuzione continua?](#)
- [Partner APN: partner per la gestione della configurazione](#)
- [Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS](#)
- [Gestione e monitoraggio della limitazione delle API nei carichi di lavoro](#)
- [Visualizza i suggerimenti di AWS Trusted Advisor su scala con AWS Organizations](#)
- [Automazione dell'aumento dei limiti di servizio e supporto aziendale con AWS Control Tower](#)
- [Operazioni, risorse e chiavi di condizione per Service Quotas](#)

Video correlati:

- [AWS Live re:Inforce 2019 - Service Quotas](#)
- [Visualizza e gestisci quote per i servizi AWS con Service Quotas](#)
- [Demo delle quote AWS IAM](#)
- [AWS re:Invent 2018: Cicli chiusi e menti aperte: come assumere il controllo di sistemi grandi e piccoli](#)

Strumenti correlati:

- [AWS CodeDeploy](#)
- [AWS CloudTrail](#)
- [Amazon CloudWatch](#)
- [Amazon EventBridge](#)
- [Amazon DevOps Guru](#)
- [AWS Config](#)
- [AWS Trusted Advisor](#)
- [AWS CDK](#)
- [AWS Systems Manager](#)
- [Marketplace AWS](#)

Pianificazione della topologia di rete

I carichi di lavoro sono spesso presenti in più ambienti. Questi includono più ambienti cloud (sia pubblicamente accessibili sia privati) e, potenzialmente, l'infrastruttura del data center esistente. I piani devono includere considerazioni di rete, ad esempio connettività intrasistema e intersistema, gestione di indirizzi IP pubblici, gestione di indirizzi IP privati e risoluzione dei nomi di dominio.

Quando si progettano sistemi che utilizzano reti basate su indirizzi IP, è necessario pianificare la topologia di rete e l'indirizzamento in anticipo di possibili guasti, nonché consentire la crescita futura e l'integrazione con altri sistemi e le relative reti.

Amazon Virtual Private Cloud (Amazon VPC) consente di effettuare il provisioning di una sezione del Cloud AWS privata e isolata dove è possibile lanciare risorse AWS in una rete virtuale.

Best practice

- [REL02-BP01 Utilizzo di una connettività di rete a disponibilità elevata per gli endpoint pubblici del carico di lavoro](#)
- [REL02-BP02 Esecuzione del provisioning di connettività ridondante tra reti private nel cloud e negli ambienti on-premise.](#)
- [REL02-BP03 Verifica che l'allocazione delle sottoreti IP consenta l'espansione e la disponibilità:](#)
- [REL02-BP04 Preferire topologie hub-and-spoke rispetto a mesh da-molti-a-molti](#)

- [REL02-BP05 Applicazione di intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi con indirizzi privati a cui sono connessi](#)

REL02-BP01 Utilizzo di una connettività di rete a disponibilità elevata per gli endpoint pubblici del carico di lavoro

La creazione di connettività di rete a disponibilità elevata agli endpoint pubblici dei carichi di lavoro può ridurre i tempi di inattività dovuti a perdita di connettività e migliorare la disponibilità e il contratto sul livello di servizio del carico di lavoro. Per ottenere questo risultato, usa un servizio DNS a disponibilità elevata, reti di distribuzione di contenuti (CDN), API Gateway, bilanciamento del carico o proxy inversi.

Risultato desiderato: è essenziale pianificare, creare e rendere operativa una connettività di rete ad alta disponibilità per gli endpoint pubblici. Se il carico di lavoro diventa irraggiungibile a causa della perdita di connettività, il sistema apparirà ai clienti come arrestato, anche se il carico di lavoro è in esecuzione e disponibile. Combinando connettività di rete a disponibilità elevata e resiliente per gli endpoint pubblici del carico di lavoro, insieme a un'architettura resiliente per il carico di lavoro stesso, puoi offrire ai clienti la disponibilità e il livello di servizio migliori possibile.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, funzioni URL AWS Lambda, API AWS AppSync e Elastic Load Balancing (ELB) forniscono tutti endpoint pubblici a disponibilità elevata. Amazon Route 53 offre un servizio DNS altamente disponibile per la risoluzione dei nomi di dominio che permette di verificare che gli indirizzi degli endpoint pubblici possano essere risolti.

Puoi anche valutare applicazioni software Marketplace AWS per il bilanciamento del carico e l'esecuzione di proxy.

Anti-pattern comuni:

- Progettazione di un carico di lavoro a disponibilità elevata senza pianificare connettività DNS e di rete per la disponibilità elevata.
- Uso di indirizzi Internet pubblici su singoli container o istanze e gestione della connettività tramite DNS.
- Uso di indirizzi IP anziché nomi di dominio per l'individuazione dei servizi.
- Mancata esecuzione di test su scenari con perdita di connettività agli endpoint pubblici.
- Mancata analisi delle esigenze di velocità di trasmissione effettiva della rete e dei modelli di distribuzione.

- Nessuna attività di test e pianificazione per scenari di possibile interruzione della connettività di rete Internet agli endpoint pubblici del carico di lavoro.
- Distribuzione di contenuti (pagine Web, asset statici o file multimediali) in un'area geografica di grandi dimensioni senza usare una rete di distribuzione di contenuti.
- Nessuna pianificazione per la prevenzione di attacchi DDoS (Distributed Denial of Service). Gli attacchi DDoS rischiano di arrestare il traffico legittimo e di ridurre la disponibilità per gli utenti.

Vantaggi dell'adozione di questa best practice: la progettazione di connettività di rete altamente disponibile e resiliente garantisce che il carico di lavoro sia accessibile e disponibile per gli utenti.

Livello di rischio associato alla mancata adozione di questa best practice: elevato

Guida all'implementazione

Alla base della creazione di connettività di rete a disponibilità elevata agli endpoint pubblici vi è l'instradamento del traffico. Per verificare che il traffico possa raggiungere gli endpoint, il servizio DNS deve essere in grado di risolvere i nomi di dominio negli indirizzi IP corrispondenti. Usa un [sistema dei nomi di dominio \(DNS\)](#) altamente disponibile e scalabile come Amazon Route 53 per gestire i record DNS del dominio. Puoi usare anche i controlli dell'integrità forniti da Amazon Route 53. I controlli dell'integrità verificano che l'applicazione sia raggiungibile, disponibile e funzionale e possono essere configurati in modo da simulare il comportamento degli utenti, come la richiesta di una pagina Web o un URL specifico. In caso di errore, Amazon Route 53 risponde alle richieste di risoluzione DNS e indirizza il traffico solo agli endpoint integri. Puoi anche valutare se usare le funzionalità di instradamento basato sulla latenza e GeoDNS offerte da Amazon Route 53.

Per verificare che il carico di lavoro stesso abbia disponibilità elevata, usa Elastic Load Balancing (ELB). Puoi usare Amazon Route 53 per indirizzare il traffico a ELB, che lo distribuisce alle istanze di calcolo di destinazione. Puoi anche usare Amazon API Gateway insieme a AWS Lambda per una soluzione serverless. I clienti possono anche eseguire carichi di lavoro in più Regioni AWS. Con il [modello attivo/attivo multisito](#), il carico di lavoro può distribuire il traffico da più regioni. Con un modello attivo/passivo multisito, il carico di lavoro distribuisce il traffico dalla regione attiva, mentre i dati vengono replicati nella regione secondaria e diventano attivi in caso di errore nella regione primaria. Puoi usare i controlli dell'integrità in Route 53 per controllare il failover DNS da qualsiasi endpoint in una regione primaria a un endpoint in una regione secondaria, verificando che il carico di lavoro sia raggiungibile e disponibile per gli utenti.

Amazon CloudFront offre una semplice API per la distribuzione di contenuti con bassa latenza e velocità di trasferimento dati elevate gestendo le richieste tramite una rete di posizioni edge

in tutto il mondo. Le reti di distribuzione di contenuti (CDN) operano per i clienti distribuendo i contenuti situati o memorizzati nella cache in una posizione vicina all'utente. In questo modo, la disponibilità dell'applicazione migliora, in quanto il carico del contenuto viene allontanato dai server verso [posizioni edge](#) di CloudFront. Le posizioni edge e le cache edge regionali includono copie memorizzate nella cache del contenuto vicino agli utenti, per il recupero rapido e una raggiungibilità e una disponibilità maggiori del carico di lavoro.

Per i carichi di lavoro con utenti distribuiti in più aree geografiche, AWS Global Accelerator contribuisce a migliorare la disponibilità e le prestazioni delle applicazioni. AWS Global Accelerator fornisce indirizzi IP statici anycast che operano come punto di ingresso statico alle applicazioni ospitate in una o più Regioni AWS. In questo modo, il traffico può entrare nella rete globale AWS il più vicino possibile agli utenti, migliorando la raggiungibilità e la disponibilità del carico di lavoro. AWS Global Accelerator monitora anche l'integrità degli endpoint dell'applicazione usando controlli dell'integrità TCP, HTTP e HTTPS. Eventuali variazioni dell'integrità o della configurazione degli endpoint attivano il reindirizzamento del traffico degli utenti a endpoint integri che offrono le prestazioni e la disponibilità migliori agli utenti. Inoltre, AWS Global Accelerator ha una progettazione di isolamento degli errori che usa due indirizzi IPv4 statici gestiti da zone di rete indipendenti, migliorando la disponibilità delle applicazioni.

Per contribuire alla protezione dei clienti da attacchi DDoS, AWS offre AWS Shield Standard. Shield Standard è abilitato per impostazione predefinita e protegge da attacchi comuni contro l'infrastruttura (livelli 3 e 4), come i flood SYN/UDP e gli attacchi di riflessione, in modo da supportare la disponibilità elevata delle applicazioni in AWS. Per altre soluzioni di protezione da attacchi più sofisticati e di maggiore entità (come i flood UDP) e di tipo state-exhaustion (come i flood TCP SYN) e per proteggere le applicazioni in esecuzione su Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator e Route 53, puoi valutare se usare AWS Shield Advanced. Per la protezione da attacchi a livello di applicazione come i flood HTTP POST o GET, usa AWS WAF. AWS WAF può usare indirizzi IP, intestazioni HTTP, corpo HTTP, stringhe URI, iniezione SQL e condizioni di scripting cross-site per determinare se una richiesta debba essere bloccata o consentita.

Passaggi dell'implementazione

1. Configura un sistema DNS a disponibilità elevata: Amazon Route 53 è un servizio Web altamente disponibile e scalabile che opera come [sistema dei nomi di dominio \(DNS\)](#). Route 53 connette le richieste utente ad applicazioni Internet in esecuzione in AWS o on-premise. Per ulteriori informazioni, consulta [Configurazione di Amazon Route 53 come servizio DNS](#).

2. Configura controlli dell'integrità: quando usi Route 53, verifica che solo le destinazioni integre siano risolvibili. Per iniziare, [crea controlli dell'integrità in Route 53 e configura il failover DNS](#). Nel configurare controlli dell'integrità, è importante tenere conto degli aspetti seguenti:
 - a. [Modo in cui Amazon Route 53 determina se un controllo dell'integrità ha esito positivo](#)
 - b. [Creazione, aggiornamento ed eliminazione di controlli dell'integrità](#)
 - c. [Monitoraggio dello stato dei controlli dell'integrità e ricezione di notifiche](#)
 - d. [Best practice per DNS in Amazon Route 53](#)
3. [Connessione del servizio DNS agli endpoint](#).
 - a. Quando usi Elastic Load Balancing come destinazione per il traffico, usa Amazon Route 53 per creare un [record alias](#) che punti all'endpoint regionale del sistema di bilanciamento del carico. Durante la creazione del record alias, imposta l'opzione Valutazione dello stato target su Sì.
 - b. Per carichi di lavoro serverless o API private con API Gateway, usa [Route 53 per indirizzare il traffico ad API Gateway](#).
4. Opta per una rete di distribuzione di contenuti (CDN).
 - a. Per distribuire contenuti usando posizioni edge più vicine all'utente, inizia acquisendo familiarità con il [modo in cui CloudFront distribuisce contenuti](#).
 - b. Inizia con una [distribuzione semplice di CloudFront](#). CloudFront sa quindi determinare dove vuoi distribuire i contenuti e come monitorare e gestire la distribuzione di contenuti. Nel configurare la distribuzione di CloudFront, è importante tenere conto degli aspetti seguenti:
 - i. [Funzionamento della memorizzazione nella cache con posizioni edge CloudFront](#)
 - ii. [Aumento della proporzione di richieste gestite direttamente dalle cache CloudFront \(tasso di riscontri nella cache\)](#)
 - iii. [Uso di Amazon CloudFront Origin Shield](#)
 - iv. [Ottimizzazione della disponibilità elevata con il failover delle origini in CloudFront](#)
5. Configura la protezione a livello di applicazione: AWS WAF semplifica la protezione da exploit Web e bot comuni che possono compromettere la disponibilità e la sicurezza o consumare risorse eccessive. Per approfondire questi concetti, consulta [Funzionamento di AWS WAF](#) e prima di implementare protezioni da flood HTTP POST e GET a livello di applicazione, consulta [Nozioni di base su AWS WAF](#). Puoi anche usare AWS WAF con CloudFront. Consulta la documentazione sul [funzionamento di AWS WAF con funzionalità di Amazon CloudFront](#).
6. Configura protezione aggiuntiva da attacchi DDoS: per impostazione predefinita, tutti i clienti AWS ricevono protezione gratuita dagli attacchi DDoS comuni e più frequenti a livello di rete e [di trasporto che prendono di mira il sito Web o l'applicazione con AWS Shield Standard](#). Per una

protezione aggiuntiva delle applicazioni con connessione Internet su Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator e Amazon Route 53, puoi prendere in considerazione [AWS Shield Advanced](#) e consultare gli [esempi di architetture resilienti ad attacchi DDoS](#). Per proteggere il carico di lavoro e gli endpoint pubblici da attacchi DDoS, consulta [Nozioni di base su AWS Shield Advanced](#).

Risorse

Best practice correlate:

- [REL10-BP01 Implementazione del carico di lavoro in diversi luoghi](#)
- [REL10-BP02 Selezione delle posizioni appropriate per la tua implementazione multiposizione](#)
- [REL11-BP04 Fare affidamento al piano dati invece che al piano di controllo durante il ripristino](#)
- [REL11-BP06 Invio di notifiche quando gli eventi influiscono sulla disponibilità](#)

Documenti correlati:

- [Partner APN: partner per la pianificazione della rete](#)
- [Marketplace AWS per l'infrastruttura di rete](#)
- [Che cos'è AWS Global Accelerator?](#)
- [Che cos'è Amazon CloudFront?](#)
- [Che cos'è Amazon Route 53?](#)
- [Che cos'è Elastic Load Balancing?](#)
- [Funzionalità di connettività di rete – Come definire gli aspetti di base del cloud](#)
- [Che cos'è Amazon API Gateway?](#)
- [Che cosa sono AWS WAF, AWS Shield e AWS Firewall Manager?](#)
- [Che cos'è il Sistema di controllo Amazon Route 53 per il ripristino di applicazioni?](#)
- [Configurazione di controlli dell'integrità personalizzati per il failover DNS](#)

Video correlati:

- [AWS re:Invent 2022: Miglioramento delle prestazioni e della disponibilità con AWS Global Accelerator](#)

- [AWS re:Invent 2020: Gestione del traffico globale con Amazon Route 53](#)
- [AWS re:Invent 2022: Esecuzione di applicazioni multi-AZ a disponibilità elevata](#)
- [AWS re:Invent 2022: Approfondimento dell'infrastruttura di rete AWS](#)
- [AWS re:Invent 2022: Creazione di reti resilienti](#)

Esempi correlati:

- [Ripristino di emergenza con il Sistema di controllo Amazon Route 53 per il ripristino di applicazioni \(ARC\)](#)
- [Workshop sull'affidabilità](#)
- [Workshop su AWS Global Accelerator](#)

REL02-BP02 Esecuzione del provisioning di connettività ridondante tra reti private nel cloud e negli ambienti on-premise.

Implementa la ridondanza delle connessioni tra reti private nel cloud e negli ambienti on-premises per ottenere la resilienza della connettività. A tal fine, puoi implementare due o più collegamenti e percorsi di traffico, preservando la connettività in caso di errori di rete.

Anti-pattern comuni:

- Dipendi da una sola connessione di rete che crea un singolo punto di errore.
- Utilizzi un solo tunnel VPN o più tunnel che terminano nella stessa zona di disponibilità.
- Ti affidi a un solo ISP per la connettività VPN, suscettibile di guasto totale in caso di interruzione dell'ISP.
- Non implementi i protocolli di instradamento dinamico come BGP, fondamentali per reindirizzare il traffico durante le interruzioni di rete.
- Ignori i limiti di larghezza di banda dei tunnel VPN e sopravvaluti le capacità di backup.

Vantaggi dell'adozione di questa best practice: implementando una connettività ridondante tra il tuo ambiente cloud e l'ambiente aziendale/on-premises, i servizi dipendenti tra i due ambienti possono comunicare in maniera affidabile.

Livello di rischio associato se questa best practice non fosse adottata: elevato

Guida all'implementazione

Quando si utilizza AWS Direct Connect per connettere la rete on-premises ad AWS, è possibile ottenere la massima resilienza di rete (SLA del 99,99%) impiegando connessioni separate che terminano su dispositivi diversi in più di una posizione on-premises e in più di una posizione AWS Direct Connect. Questa topologia offre resilienza agli errori dei dispositivi, ai problemi di connettività e alle interruzioni complete della posizione. In alternativa, puoi ottenere un'elevata resilienza (SLA del 99,9%) utilizzando due singole connessioni a più posizioni, con ciascuna posizione on-premises connessa a una singola posizione Direct Connect. Questo approccio offre protezione dalle interruzioni della connettività causate da errori della fibra o guasti dei dispositivi e aiuta a mitigare le interruzioni complete della posizione. Il kit di strumenti di resilienza di AWS Direct Connect può aiutarti a progettare la tua topologia AWS Direct Connect.

Puoi anche considerare di utilizzare AWS Site-to-Site VPN che termina su AWS Transit Gateway come soluzione conveniente di backup sulla connessione primaria AWS Direct Connect. Questa configurazione abilita l'instradamento equal-cost multi-path (ECMP) su più tunnel VPN, consentendo un throughput fino a 50 Gbps, anche se ogni tunnel VPN è limitato a 1,25 Gbps. È importante notare, tuttavia, che AWS Direct Connect è ancora la scelta più efficace per ridurre al minimo le interruzioni di rete e fornire una connettività stabile.

Quando utilizzi le VPN su Internet per connettere l'ambiente cloud al tuo data center on-premises, configura due tunnel VPN come parte di un'unica connessione VPN sito-sito. Ogni tunnel deve terminare in una zona di disponibilità diversa per garantire l'alta disponibilità e utilizzare hardware ridondante per prevenire gli errori dei dispositivi on-premises. Inoltre, prendi in considerazione di utilizzare più connessioni Internet di vari provider di servizi Internet (ISP) per la tua posizione on-premises per evitare l'interruzione completa della connettività VPN dovuta al guasto di un singolo ISP. La scelta di ISP con instradamento e infrastrutture diversi, in particolare quelli con percorsi fisici separati verso gli endpoint AWS, offre un'elevata disponibilità della connettività.

Oltre alla ridondanza fisica con più connessioni AWS Direct Connect e più tunnel VPN, o una combinazione di entrambi, è fondamentale anche l'implementazione dell'instradamento dinamico del Border Gateway Protocol (BGP). Il BGP dinamico fornisce il reinstradamento automatico del traffico da un percorso all'altro in base alle condizioni della rete in tempo reale e alle policy configurate. Questo comportamento dinamico è particolarmente utile per mantenere la disponibilità della rete e la continuità del servizio in caso di errori di collegamento o rete. Seleziona rapidamente percorsi alternativi, migliorando la resilienza e l'affidabilità della rete.

Passaggi dell'implementazione

- Acquisisci la connettività ad alta disponibilità tra AWS e l'ambiente on-premises.
 - Utilizza più connessioni AWS Direct Connect o tunnel VPN tra reti private implementate separatamente.
 - Utilizza più posizioni AWS Direct Connect per l'alta disponibilità.
 - Se utilizzi più Regioni AWS, garantisci la ridondanza in almeno due di esse.
- Usa AWS Transit Gateway, quando possibile, per terminare la [connessione VPN](#).
- Valuta le appliance di Marketplace AWS per terminare le VPN o [estendere la SD-WAN ad AWS](#). Se utilizzi appliance di Marketplace AWS, distribuisce le istanze ridondanti per la disponibilità elevata in diverse zone di disponibilità.
- Fornisci una connessione ridondante all'ambiente on-premises.
 - Per soddisfare le esigenze di disponibilità, possono essere necessarie connessioni ridondanti a più Regioni AWS.
 - Utilizza il [kit di strumenti di resilienza di AWS Direct Connect](#) per iniziare.

Risorse

Documenti correlati:

- [AWS Direct Connect Resiliency Recommendations](#)
- [Using Redundant Site-to-Site VPN Connections to Provide Failover](#)
- [Policy di routing e community BGP](#)
- [Active/Active and Active/Passive Configurations in AWS Direct Connect](#)
- [Partner APN: partner per la pianificazione della rete](#)
- [Marketplace AWS per l'infrastruttura di rete](#)
- [Amazon Virtual Private Cloud Connectivity Options Whitepaper](#)
- [Creazione di un'infrastruttura di rete AWS scalabile e sicura con più VPC](#)
- [Using redundant Site-to-Site VPN connections to provide failover](#)
- [Using the AWS Direct Connect Resiliency Toolkit to get started](#)
- [VPC Endpoints and VPC Endpoint Services \(AWS PrivateLink\)](#)
- [What is Amazon VPC?](#)
- [What is a transit gateway?](#)

- [What is AWS Site-to-Site VPN?](#)
- [Working with Direct Connect gateways](#)

Video correlati:

- [AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC](#)
- [AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs](#)

REL02-BP03 Verifica che l'allocazione delle sottoreti IP consenta l'espansione e la disponibilità:

Gli intervalli di indirizzi IP dei Amazon VPC devono essere sufficientemente ampi per soddisfare i requisiti del carico di lavoro, tenendo conto anche dell'espansione futura e dell'allocazione degli indirizzi IP alle sottoreti nelle zone di disponibilità. Sono inclusi sistemi di bilanciamento del carico, istanze EC2 e applicazioni basate su container.

Quando si pianifica la topologia di rete, il primo passo è definire lo spazio stesso degli indirizzi IP. Gli intervalli di indirizzi IP privati (secondo le linee guida RFC 1918) dovrebbero essere allocati per ogni VPC. Nell'ambito di questo processo, soddisfa i seguenti requisiti:

- Lascia spazi per indirizzi IP per più di un VPC per Regione.
- All'interno di un VPC, lascia spazio per più sottoreti affinché coprano più zone di disponibilità.
- Prendi in considerazione di lasciare spazio per un blocco CIDR inutilizzato all'interno di un VPC per un'espansione futura.
- Assicurati che sia disponibile spazio per gli indirizzi IP, al fine di soddisfare le esigenze di qualsiasi parco istanze Amazon EC2 transitorio che puoi utilizzare, ad esempio parchi istanze spot per il machine learning, cluster Amazon EMR o cluster Amazon Redshift. Una considerazione analoga andrebbe fatta per i cluster Kubernetes, come Amazon Elastic Kubernetes Service (Amazon EKS), poiché per impostazione predefinita a ciascun pod Kubernetes viene assegnato un indirizzo instradabile dal blocco CIDR VPC.
- Tieni presente che i primi quattro indirizzi IP e l'ultimo indirizzo IP in ogni blocco CIDR della sottorete sono riservati e non disponibili per l'uso.
- Tieni presente che il blocco CIDR VPC iniziale allocato al VPC non può essere modificato o eliminato, ma puoi aggiungere ulteriori blocchi CIDR non sovrapposti al VPC. I CIDR IPv4 della sottorete non possono essere modificati, mentre ciò è possibile con i CIDR IPv6.

- Il blocco CIDR VPC più grande possibile è /16 e il più piccolo è /28.
- Prendi in considerazione altre reti connesse (VPC, on-premises o altri provider cloud) e assicurati che lo spazio degli indirizzi IP non si sovrapponga. Per ulteriori informazioni, consulta [REL02-BP05 Applicazione di intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi con indirizzi privati a cui sono connessi](#).

Risultato desiderato: Una sottorete IP scalabile può aiutarti a far fronte alla crescita futura e a evitare inutili sprechi.

Anti-pattern comuni:

- Non prendere in considerazione la crescita futura, con conseguenti blocchi CIDR troppo piccoli e che richiedono una riconfigurazione, il che comporta tempi di inattività.
- Stima erronea del numero di indirizzi IP utilizzabili da un bilanciatore del carico.
- Distribuzione di numerosi sistemi di bilanciamento del carico a traffico elevato nelle stesse sottoreti.
- Utilizzo di meccanismi di dimensionamento automatico senza monitorare il consumo di indirizzi IP.
- Definizione di intervalli CIDR eccessivamente ampi ben oltre le aspettative di crescita futura, il che può portare a difficoltà di peering con altre reti con intervalli di indirizzi sovrapposti.

Vantaggi dell'adozione di questa best practice: In questo modo puoi consentire la crescita dei carichi di lavoro e continuare a fornire disponibilità man mano che incrementi le dimensioni.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

Pianificazione della rete in base a crescita, compliance normativa e integrazione con altre reti. Senza una pianificazione adeguata, la crescita può essere sottovalutata, la compliance normativa può cambiare e l'implementazione di acquisizioni o di connessioni a reti private può rivelarsi difficile.

- Seleziona gli Account AWS e le Regioni pertinenti in base ai tuoi requisiti di servizio, di latenza, normativi e di ripristino di emergenza.
- Identifica le esigenze delle implementazioni di VPC regionali.
- Identifica le dimensioni dei VPC.
 - Stabilisci se intendi implementare connettività multi-VPC.
 - [Che cos'è un Transit Gateway?](#)

- [Connettività multi-VPC a singola Regione](#)
- Stabilisci se hai bisogno di reti separate a causa di requisiti normativi.
- Crea VPC con blocchi CIDR di dimensioni adeguate per soddisfare le tue esigenze attuali e future.
 - Se non hai definito proiezioni di crescita, potresti preferire blocchi CIDR più grandi per ridurre il potenziale di riconfigurazione futura
- Prendi in considerazione l'utilizzo di [un indirizzo IPv6](#) per le sottoreti come parte di un VPC dual-stack. Un IPv6 è adatto per essere utilizzato in sottoreti private contenenti pochi istanze o contenitori temporanei che altrimenti richiederebbero un numero elevato di indirizzi IPv4.

Risorse

Best practice Well-Architected correlate:

- [REL02-BP05 Applicazione di intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi con indirizzi privati a cui sono connessi](#)

Documenti correlati:

- [Partner APN: partner per la pianificazione della rete](#)
- [Marketplace AWS per l'infrastruttura di rete](#)
- [Whitepaper: Opzioni di connettività di Amazon Virtual Private Cloud](#)
- [Connettività di rete di elevata disponibilità in più data center](#)
- [Connettività multi-VPC a singola Regione](#)
- [Che cos'è Amazon VPC?](#)
- [IPv6 su AWS](#)
- [IPv6 on reference architectures](#)
- [Amazon Elastic Kubernetes Service launches IPv6 support](#)

Video correlati:

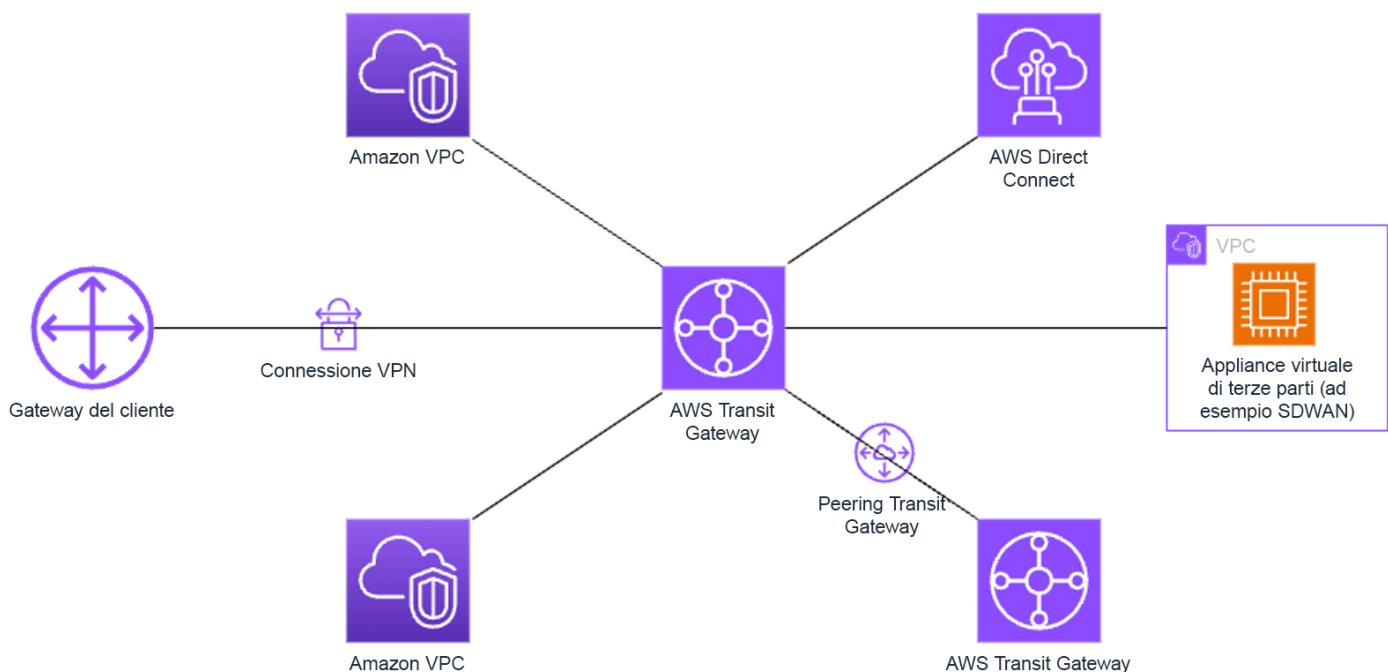
- [AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC \(Progettazione avanzata di VPC e nuove funzionalità per Amazon VPC\) \(NET303\)](#)
- [AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs \(NET406-R1\)](#)

- [AWS re:Invent 2023: AWS Ready for what's next? Designing networks for growth and flexibility \(NET310\)](#)

REL02-BP04 Preferire topologie hub-and-spoke rispetto a mesh da-molti-a-molti

Quando connetti più reti private, come cloud privati virtuali (VPC) e reti locali, è opportuno scegliere una topologia hub-and-spoke rispetto a una mesh. A differenza delle topologie mesh, in cui ogni rete si connette direttamente alle altre e aumenta la complessità e il sovraccarico di gestione, l'architettura hub-and-spoke centralizza le connessioni tramite un unico hub. Questa centralizzazione semplifica la struttura della rete e ne migliora il funzionamento, la scalabilità e il controllo.

AWS Transit Gateway è un servizio gestito, scalabile e a disponibilità elevata progettato per la creazione di reti hub-and-spoke su AWS. Serve da hub centrale della rete che fornisce la segmentazione, il routing centralizzato e la connessione semplificata agli ambienti cloud e on-premises. La figura seguente illustra come è possibile utilizzare AWS Transit Gateway per creare la topologia hub-and-spoke.



Anti-pattern comuni:

- Si complicano eccessivamente le policy di routing in un'architettura hub-and-spoke, riducendo l'efficienza della rete e ostacolando sia la risoluzione dei problemi sia la gestione proattiva.
- Una segmentazione insufficiente basata sul routing all'interno dell'hub potrebbe comportare vulnerabilità che potenzialmente espongono la rete ad accessi non autorizzati.
- Senza un'attenta ottimizzazione, il traffico instradato attraverso l'hub può comportare elevati costi di trasferimento dei dati, in particolare per il traffico che attraversa zone di disponibilità e regioni. L'uso di efficaci strategie di gestione del traffico è essenziale per controllare le spese.

Vantaggi dell'adozione di questa best practice: con l'aumento del numero di reti connesse, la gestione e l'espansione della connettività mesh diventano sempre più difficili. AWS Transit Gateway offre un hub gestito dimensionabile e affidabile per la creazione e il funzionamento delle topologie hub-and-spoke. Con AWS Transit Gateway puoi stabilire connessioni e centralizzare il routing del traffico su più reti.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

- Pianifica la rete.
- Crea un AWS Transit Gateway.
- Collega i VPC.
- Se necessario, crea connessioni VPN o gateway Direct Connect e associali a Transit Gateway.
- Definisci come viene instradato il traffico tra i VPC connessi e altre connessioni tramite la configurazione delle tabelle di routing di Transit Gateway.
- Usa Amazon CloudWatch per monitorare e modificare come necessario le configurazioni per l'ottimizzazione delle prestazioni e dei costi.

Risorse

Documenti correlati:

- [What Is a Transit Gateway?](#)
- [Creazione di un'infrastruttura di rete AWS multi-VPC sicura e scalabile](#)
- [Building a global network using AWS Transit Gateway Inter-Region peering](#)
- [Amazon Virtual Private Cloud Connectivity Options](#)

- [APN Partner: partners that can help plan your networking](#)
- [Marketplace AWS for Network Infrastructure](#)

Video correlati:

- [AWS re:Invent 2023 - AWS networking foundations](#)
- [AWS re:Invent 2023 - Advanced VPC designs and new capabilities](#)

REL02-BP05 Applicazione di intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi con indirizzi privati a cui sono connessi

Gli intervalli di indirizzi IP di ogni VPC non devono sovrapporsi quando sono collegati in peering o connessi tramite Transit Gateway o VPN. Evita i conflitti di indirizzi IP tra VPC e ambienti on-premises o altri provider di servizi cloud utilizzati. Bisogna inoltre disporre di un modo per allocare gli intervalli di indirizzi IP privati quando necessario. Un sistema di gestione degli indirizzi IP (IPAM) può aiutarti ad automatizzare l'allocazione.

Risultato desiderato:

- Nessun conflitto di intervalli di indirizzi IP tra VPC, ambienti on-premises o altri provider di servizi cloud.
- La corretta gestione degli indirizzi IP consente di scalare più facilmente l'infrastruttura di rete per supportare la crescita e i cambiamenti dei requisiti di rete.

Anti-pattern comuni:

- Utilizzo nel VPC dello stesso intervallo di indirizzi IP usato on-premises, nella rete aziendale o in altro provider di servizi cloud.
- Non tenere traccia degli intervalli IP dei VPC utilizzati per distribuire i carichi di lavoro.
- Ricorso a processi manuali di gestione degli indirizzi IP, come i fogli di calcolo.
- Utilizzo di blocchi CIDR sovradimensionati o sottodimensionati, con conseguente spreco di indirizzi IP o spazio di indirizzi insufficiente per il carico di lavoro.

Vantaggi derivanti dall'adozione di questa best practice: la pianificazione attiva della rete garantisce di non avere più occorrenze dello stesso indirizzo IP nelle reti interconnesse. In questo modo si evitano problemi di instradamento in parti del carico di lavoro che utilizzano le diverse applicazioni.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

Utilizza un sistema IPAM, ad esempio [Amazon VPC IP Address Manager](#), per monitorare e gestire l'uso del CIDR. Su Marketplace AWS sono disponibili anche diversi IPAM. Valuta il tuo utilizzo potenziale su AWS, aggiungi intervalli CIDR ai VPC esistenti e crea i VPC per consentire la crescita pianificata dell'utilizzo.

Passaggi dell'implementazione

- Acquisisci il consumo attuale del CIDR, ad esempio VPC e sottoreti.
 - Utilizza le operazioni delle API di servizi per raccogliere il consumo attuale di CIDR.
 - Usa [Amazon VPC IP Address Manager per individuare le risorse](#).
- Acquisisci l'utilizzo attuale delle sottoreti.
 - Utilizza le operazioni delle API di servizio per [raccogliere le sottoreti](#) per ogni VPC di ciascuna regione.
 - Usa [Amazon VPC IP Address Manager per individuare le risorse](#).
- Registra l'uso attuale.
- Verifica se hai creato intervalli di indirizzi IP sovrapposti.
- Calcola la capacità inutilizzata.
- Individua gli intervalli di indirizzi IP sovrapposti. Puoi eseguire la migrazione a un nuovo intervallo di indirizzi o prendere in considerazione l'utilizzo di tecniche quali il [gateway NAT privato](#) o [AWS PrivateLink](#), se hai l'esigenza di connettere intervalli sovrapposti.

Risorse

Best practice correlate:

- [Protezione delle reti](#)

Documenti correlati:

- [Partner APN: partner per la pianificazione della rete](#)
- [Marketplace AWS per l'infrastruttura di rete](#)
- [Amazon Virtual Private Cloud Connectivity Options Whitepaper](#)
- [Connettività di rete di elevata disponibilità in più data center](#)
- [Connecting Networks with Overlapping IP Ranges](#)
- [What is Amazon VPC?](#)
- [Che cos'è IPAM?](#)

Video correlati:

- [AWS re:Invent 2023 - Advanced VPC designs and new capabilities](#)
- [AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs](#)
- [AWS re:Invent 2023 - Ready for what's next? Designing networks for growth and flexibility](#)
- [AWS re:Invent 2021 - {New Launch} Manage your IP addresses at scale on AWS](#)

Architettura del carico di lavoro

Un carico di lavoro affidabile comincia con decisioni iniziali di progettazione sia per il software sia per l'infrastruttura. Le tue scelte architetturali avranno un impatto sul comportamento del carico di lavoro su tutti e sei i pilastri di base del Framework Well-Architected. Per l'affidabilità, è necessario seguire modelli specifici.

Nelle sezioni seguenti vengono illustrate le best practice da utilizzare con questi modelli per l'affidabilità.

Argomenti

- [Progettazione dell'architettura del servizio di carico di lavoro](#)
- [Progetta interazioni in un sistema distribuito per prevenire guasti](#)
- [Progettazione di interazioni in un sistema distribuito per mitigare o affrontare gli errori](#)

Progettazione dell'architettura del servizio di carico di lavoro

Creazione di carichi di lavoro altamente scalabili e affidabili utilizzando un'architettura orientata ai servizi (SOA) o un'architettura di microservizi. L'architettura orientata ai servizi (SOA) è la pratica di rendere i componenti software riutilizzabili tramite interfacce di servizio. L'architettura dei microservizi va oltre, per rendere i componenti più piccoli e semplici.

Le interfacce di architettura orientata ai servizi (SOA) utilizzano standard di comunicazione comuni in modo che possano essere rapidamente integrate in nuovi carichi di lavoro. La SOA ha sostituito la pratica di costruire architetture monolitiche, che consistevano in unità interdipendenti e indivisibili.

In AWS, abbiamo sempre utilizzato la SOA, ma ora abbiamo adottato la creazione dei nostri sistemi utilizzando microservizi. Anche se i microservizi hanno diverse qualità interessanti, il vantaggio più importante per la disponibilità è che i microservizi sono più piccoli e semplici. Consentono di differenziare la disponibilità richiesta di diversi servizi, concentrando quindi gli investimenti in modo più specifico sui microservizi che hanno le maggiori esigenze di disponibilità. Ad esempio, per distribuire pagine di informazioni sui prodotti su Amazon.com ("pagine dei dettagli"), vengono invocati centinaia di microservizi per creare porzioni discrete della pagina. Sebbene ci siano alcuni servizi che devono essere disponibili per fornire il prezzo e i dettagli del prodotto, la maggior parte dei contenuti della pagina può essere semplicemente esclusa se il servizio non è disponibile. Anche elementi come foto e recensioni non sono necessari per fornire un'esperienza in cui un cliente può acquistare un prodotto.

Best practice

- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)
- [REL03-BP02 Creazione di servizi focalizzati su domini e funzionalità aziendali specifici](#)
- [REL03-BP03 Fornitura di contratti di servizio per API](#)

REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro

La segmentazione del carico di lavoro è importante quando vengono determinati i requisiti di resilienza dell'applicazione. L'architettura monolitica deve essere evitata se possibile. Valuta invece con particolare attenzione quali componenti dell'applicazione possono essere suddivisi in microservizi. A seconda dei requisiti dell'applicazione, ciò potrebbe risultare in una combinazione di architettura orientata ai servizi (SOA) e microservizi, laddove possibile. I carichi di lavoro stateless sono maggiormente idonei a essere implementati come microservizi.

Risultato desiderato: i carichi di lavoro devono essere supportabili, scalabili e devono essere caratterizzati dalla minore interdipendenza possibile.

Quando scegli come segmentare il carico di lavoro, trova il giusto compromesso tra i vantaggi e le complessità. Ciò che è giusto per un nuovo prodotto al primo lancio è diverso dai requisiti di un carico di lavoro creato per ridimensionare le risorse. Durante la rifattorizzazione (riprogettazione) di un monolito, dovrai considerare la capacità dell'applicazione di supportare la suddivisione in servizi stateless. La suddivisione dei servizi in elementi più piccoli consente a team ristretti e ben definiti di svilupparli e gestirli. Tuttavia, servizi di piccole dimensioni possono introdurre complessità, che includono un eventuale aumento della latenza, un debug più complesso e un maggiore carico operativo.

Anti-pattern comuni:

- Il [microservizio Death Star](#) rappresenta una situazione in cui i componenti atomici diventano così interdipendenti che un errore verificatosi in un componente genera un errore molto più grande, rendendo i componenti rigidi e fragili se considerati come monolito.

Vantaggi dell'adozione di questa best practice:

- Segmenti più specifici comportano maggiore agilità, flessibilità organizzativa e scalabilità.
- Riduzione dell'impatto derivante dall'interruzione dei servizi.

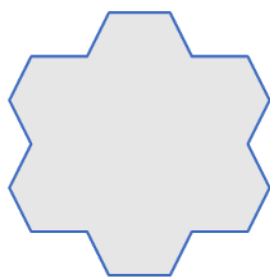
- I componenti dell'applicazione possono avere requisiti di disponibilità diversi, che a loro volta possono essere supportati da una segmentazione più atomica.
- Responsabilità ben definite per i team che supportano il carico di lavoro.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

Scegli il tipo di architettura in base al tipo di segmentazione del carico di lavoro. Scegli una SOA o un'architettura di microservizi (o, in alcuni rari casi, un'architettura monolitica). Anche se scegli di iniziare con un'architettura monolitica, devi assicurarti che sia modulare e possa evolvere in SOA o microservizi man mano che il prodotto si dimensiona con l'adozione da parte degli utenti. La SOA e i microservizi offrono rispettivamente una segmentazione più piccola, preferita come architettura moderna scalabile e affidabile, ma ci sono compromessi da considerare soprattutto quando si distribuisce un'architettura di microservizi.

Uno dei principali compromessi è che ora disponi di un'architettura di calcolo distribuita che può rendere più difficile il raggiungimento dei requisiti di latenza degli utenti ed è presente un'ulteriore complessità nel debug e nel tracciamento delle interazioni degli utenti. Puoi utilizzare AWS X-Ray per risolvere questo problema. Un altro effetto da considerare è l'aumento della complessità operativa man mano che aumenta il numero di applicazioni che gestisci, che richiede la distribuzione di più componenti di indipendenza.



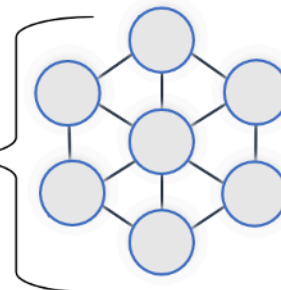
Applicazione monolitica

Esegue tutto
Pipeline di rilascio condivisa
Dimensionamento rigido
Elevato impatto della modifica
È complicato adottare nuove tecnologie



Orientato ai servizi

Esegue alcune attività
Servizi esposti mediante protocollo di comunicazione
Accoppiamento di alcuni servizi



Microservizi

Esegue una sola attività
Implementazioni indipendenti
Ridimensionamento indipendente
Impatto ridotto della modifica
Possibilità di scelta della tecnologia

Architettura monolitica, orientata ai servizi e di microservizi

Passaggi dell'implementazione

- Determina l'architettura più appropriata per rifattorizzare (riprogettare) o creare l'applicazione. SOA e microservizi offrono segmentazione rispettivamente di dimensioni minori, preferita in quanto architettura moderna, scalabile e affidabile. SOA può essere un buon compromesso per ottenere una segmentazione di dimensioni minori, evitando al contempo alcune delle complessità dei microservizi. Per ulteriori dettagli, consulta [I compromessi dei microservizi](#).
- Se il carico di lavoro è adatto e la tua organizzazione può supportarla, è consigliabile utilizzare un'architettura di microservizi per ottenere la massima agilità e affidabilità. Per ulteriori dettagli, consulta [Implementing Microservices on AWS \(Implementazione di microservizi in AWS\)](#).
- Considera l'ipotesi di attenerti al modello [Strangler Fig](#) per eseguire la rifattorizzazione (riprogettazione) di un monolito in componenti più piccoli. Ciò comporta la graduale sostituzione di componenti specifici dell'applicazione con nuove applicazioni e nuovi servizi. [AWS Migration Hub Refactor Spaces](#) funge da punto di partenza per la rifattorizzazione incrementale. Per ulteriori dettagli, consulta [Seamlessly migrate on-premises legacy workloads using a strangler pattern \(Migrazione senza problemi di carichi di lavoro legacy on-premise mediante un modello Strangler\)](#).
- L'implementazione di microservizi può richiedere un meccanismo di individuazione dei servizi per consentire ai servizi distribuiti di comunicare tra loro. [AWS App Mesh](#) può essere utilizzato con architetture orientate ai servizi per offrire rilevamento e accesso affidabili ai servizi. [AWS Cloud Map](#) può inoltre essere utilizzato per il rilevamento dinamico dei servizi basato su DNS.
- In caso di migrazione da un monolito a una SOA, [Amazon MQ](#) può aiutare a colmare il divario come bus del servizio durante la riprogettazione delle applicazioni legacy nel cloud.
- Per i monoliti esistenti con un unico database condiviso, scegli come riorganizzare i dati in segmenti più piccoli. Questa riorganizzazione può avvenire per unità aziendale, schema di accesso o struttura dei dati. A questo punto del processo di rifattorizzazione (riprogettazione), deve orientare la scelta verso un database di tipo relazionale o non relazionale (NoSQL). Per ulteriori dettagli, consulta [From SQL to NoSQL \(Da SQL a NoSQL\)](#).

Livello di impegno per il piano di implementazione: alto

Risorse

Best practice correlate:

- [REL03-BP02 Creazione di servizi focalizzati su domini e funzionalità aziendali specifici](#)

Documenti correlati:

- [Amazon API Gateway: configurazione di una REST API mediante OpenAPI](#)
- [Cosa si intende per architettura orientata ai servizi?](#)
- [Bounded Context \(un modello centrale in Domain-Driven Design\)](#)
- [Implementazione di microservizi in AWS](#)
- [I compromessi dei microservizi](#)
- [Microservizi: una definizione di questo nuovo termine di architettura](#)
- [Implementazione di microservizi in AWS](#)
- [What is AWS App Mesh? \(Che cos'è AWS App Mesh?\)](#)

Esempi correlati:

- [Iterative App Modernization Workshop \(Workshop sulla modernizzazione delle applicazioni interattive\)](#)

Video correlati:

- [Delivering Excellence with Microservices on AWS \(Implementazione dell'eccellenza con i microservizi in AWS\)](#)

REL03-BP02 Creazione di servizi focalizzati su domini e funzionalità aziendali specifici

L'architettura orientata ai servizi (SOA) definisce servizi con funzioni ben delineate determinate dalle esigenze aziendali. I microservizi utilizzano modelli di dominio e contesto delimitato per tracciare i limiti dei servizi lungo i confini del contesto aziendale. Concentrarsi sui domini e sulle funzionalità aziendali aiuta i team a definire requisiti di affidabilità indipendenti per i propri servizi. I contesti delimitati isolano e incapsulano la logica aziendale, consentendo ai team di ragionare meglio su come gestire gli errori.

Risultato desiderato: ingegneri e parti interessate aziendali definiscono congiuntamente contesti delimitati e li utilizzano per progettare sistemi come servizi che soddisfano funzioni aziendali specifiche. Questi team utilizzano pratiche consolidate come l'event storming per definire i requisiti. Le nuove applicazioni sono concepite come servizi con confini ben definiti e con accoppiamento

debole. I monoliti esistenti vengono scomposti in [contesti delimitati](#) e la progettazione dei sistemi si sposta verso architetture SOA o microservizi. Quando i monoliti vengono rifattorizzati, vengono applicati approcci consolidati come contesti a bolle e schemi di decomposizione dei monoliti.

I servizi orientati al dominio vengono eseguiti come uno o più processi che non condividono lo stato. Rispondono in modo indipendente alle fluttuazioni della domanda e gestiscono gli scenari di errore alla luce dei requisiti specifici del dominio.

Anti-pattern comuni:

- I team sono formati su domini tecnici specifici come UI e UX, middleware o database anziché su domini aziendali specifici.
- Le applicazioni coprono le responsabilità di dominio. I servizi che coprono contesti delimitati possono essere più difficili da gestire, richiedere maggiori sforzi di test ed esigere la partecipazione di più team di dominio agli aggiornamenti software.
- Le dipendenze a livello di dominio, come le librerie di entità di dominio, sono condivise tra i servizi, in modo che le modifiche per il dominio di un servizio richiedano modifiche ad altri domini dei servizi.
- I contratti di servizio e la logica aziendale non esprimono le entità in un linguaggio di dominio comune e coerente, con il risultato di livelli di traduzione che complicano i sistemi e aumentano le attività di debug.

Vantaggi dell'adozione di questa best practice: le applicazioni sono progettate come servizi indipendenti limitati da domini aziendali e utilizzano un linguaggio aziendale comune. I servizi sono testabili e implementabili in modo indipendente. I servizi soddisfano i requisiti di resilienza specifici del dominio implementato.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

La decisione basata sul dominio (DDD) costituisce l'approccio fondamentale alla progettazione e alla creazione di software attorno ai domini aziendali. È utile utilizzare un framework esistente quando si creano servizi incentrati sui domini aziendali. Quando si utilizzano applicazioni monolitiche esistenti, è possibile sfruttare i modelli di decomposizione che forniscono tecniche consolidate per modernizzare le applicazioni in servizi.



Decisione basata sul dominio

Passaggi dell'implementazione

- I team possono organizzare eventi di [event storming](#) per identificare rapidamente eventi, comandi, aggregati e domini.
- Una volta che le entità e le funzioni di dominio sono state definite in un contesto di dominio, puoi dividere il tuo dominio in servizi utilizzando il [contesto delimitato](#), dove le entità che condividono caratteristiche e attributi simili vengono raggruppate insieme. Con il modello diviso in contesti, emerge un modello su come delimitare i microservizi.
 - Ad esempio, le entità del sito Web Amazon.com possono includere elementi quali pacchetti, distribuzione, pianificazione, prezzo, sconto e valuta.
 - Il pacchetto, la distribuzione e la pianificazione sono raggruppati nel contesto di spedizione, mentre il prezzo, lo sconto e la valuta sono raggruppati nel contesto dei prezzi.
- [Scomporre i monoliti in microservizi](#) delinea i modelli per la rifattorizzazione dei microservizi. L'utilizzo di modelli per la decomposizione in base a capacità aziendale, sottodominio o transazione si allinea bene agli approcci basati sul dominio.
- Tecniche di strategia come il [contesto a bolle](#) consentono di introdurre la decisione basata sul dominio (DDD) in applicazioni esistenti o precedenti senza riscritture anticipate e impegni completi nei confronti di DDD. In un approccio basato sul contesto a bolle, viene stabilito un contesto delimitato utilizzando una mappatura e un coordinamento dei servizi, oppure il [livello anti-danneggiamento \(ACL\)](#), che protegge il modello di dominio appena definito dalle influenze esterne.

Dopo aver eseguito l'analisi del dominio e definito le entità e i contratti di servizio, i team possono utilizzare i servizi AWS per implementare la progettazione basata sul dominio come servizi basati sul cloud.

- Inizia a sviluppare definendo test che applichino le regole aziendali del tuo dominio. Lo sviluppo basato sui test (TDD) e lo sviluppo basato sul comportamento (BDD) aiutano i team a focalizzare i servizi sulla risoluzione dei problemi aziendali.
- Seleziona i [servizi AWS](#) che soddisfano al meglio i requisiti del tuo dominio aziendale e l'[architettura di microservizi](#):
 - [AWS Serverless](#) consente al team di concentrarsi su una logica di dominio specifica anziché sulla gestione di server e infrastrutture.
 - [I container in AWS](#) semplificano la gestione della tua infrastruttura, in modo da poterti concentrare sui requisiti del tuo dominio.
 - [I database dedicati](#) ti aiutano ad adattare i requisiti del tuo dominio al tipo di database più idoneo.
- [La creazione di architetture esagonali su AWS](#) delinea un framework per integrare la logica aziendale nei servizi che funzionano a ritroso da un dominio aziendale per soddisfare i requisiti funzionali e, quindi, per collegare adattatori di integrazione. I modelli che separano i dettagli dell'interfaccia dalla logica aziendale con i servizi AWS aiutano i team a concentrarsi sulla funzionalità del dominio e a migliorare la qualità del software.

Risorse

Best practice correlate:

- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)
- [REL03-BP03 Fornitura di contratti di servizio per API](#)

Documenti correlati:

- [Microservizi AWS](#)
- [Implementazione di microservizi in AWS](#)
- [How to break a Monolith into Microservices \(Come trasformare un monolite in microservizi\)](#)
- [Getting Started with DDD when Surrounded by Legacy Systems \(Iniziare con il DDD quando si è circondati da sistemi legacy\)](#)
- [Domain-Driven Design: Tackling Complexity in the Heart of Software \(Progettazione basata sul dominio: affrontare la complessità al cuore del software\)](#)
- [La creazione di architetture esagonali su AWS](#)
- [Scomporre i monoliti in microservizi](#)

- [Event Storming](#)
- [Messaggi tra contesti limitati](#)
- [Microservizi](#)
- [Sviluppo basato su test](#)
- [Sviluppo basato sul comportamento](#)

Esempi correlati:

- [Workshop sul cloud aziendale nativo](#)
- [Progettazione di microservizi cloud nativi su AWS \(da DDD/EventStormingWorkshop\)](#)

Strumenti correlati:

- [Database su Cloud AWS](#)
- [Serverless su AWS](#)
- [Container in AWS](#)

REL03-BP03 Fornitura di contratti di servizio per API

I contratti di assistenza sono accordi documentati tra produttori di API e utenti definiti in una definizione di API leggibile dal computer. Una strategia di controllo delle versioni dei contratti consente agli utenti di continuare a utilizzare l'API esistente e migrare le applicazioni a un'API più recente quando sono pronte. L'implementazione da parte del produttore può avvenire in qualsiasi momento, purché il processo sia conforme al contratto. Il team dei servizi può utilizzare lo stack tecnologico scelto per soddisfare il contratto API.

Risultato desiderato:

Anti-pattern comuni: Le applicazioni realizzate con architetture orientate ai servizi o con architetture di microservizi sono in grado di funzionare in modo indipendente pur essendo caratterizzate da una dipendenza dal runtime integrata. Le modifiche apportate a un utente o produttore di API non pregiudicano la stabilità dell'intero sistema quando entrambe le parti sono conformi a un contratto API comune. I componenti che comunicano tramite le API di servizio possono eseguire release funzionali indipendenti, aggiornamenti delle dipendenze di runtime o eseguire il failover su un sito di ripristino di emergenza con un impatto reciproco minimo o nullo. Inoltre, i servizi discreti sono in grado di

eseguire il dimensionamento in modo indipendente assorbendo la richiesta di risorse senza che gli altri servizi debbano ridimensionarsi di conseguenza.

- Creazione di API di servizio senza schemi fortemente tipizzati. Ciò si traduce in API che non possono essere utilizzate per generare collegamenti API e payload che non possono essere convalidati a livello di codice.
- Non adottare una strategia di controllo delle versioni, che costringa gli utenti delle API all'aggiornamento e rilascio o all'esito negativo dell'operazione al variare dei contratti di servizio.
- Messaggi di errore che divulgano dettagli sull'implementazione del servizio sottostante anziché descrivere errori di integrazione nel contesto e nel linguaggio del dominio.
- Non utilizzare contratti API per sviluppare casi di test e simulare implementazioni API per consentire test indipendenti dei componenti del servizio.

Vantaggi dell'adozione di questa best practice: i sistemi distribuiti composti da componenti che comunicano tramite contratti di servizio API possono migliorare l'affidabilità. Gli sviluppatori possono rilevare potenziali problemi nelle prime fasi del processo di sviluppo con il controllo del tipo durante la compilazione per verificare che le richieste e le risposte siano conformi al contratto API e che i campi obbligatori siano presenti. I contratti API forniscono una chiara interfaccia di documentazione automatica per le API e garantiscono una migliore interoperabilità tra sistemi e linguaggi di programmazione diversi.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

Dopo aver individuato i domini aziendali e determinato la segmentazione del carico di lavoro, puoi sviluppare le API dei tuoi servizi. Innanzitutto, definisci contratti di servizio leggibili dal computer per le API, quindi implementa una strategia di controllo delle versioni delle API. Quando sei pronto per integrare servizi su protocolli comuni come REST, GraphQL o eventi asincroni, puoi incorporare servizi AWS nell'architettura per integrare i componenti con contratti API fortemente tipizzati.

I servizi AWS per i contratti API di servizio

includono servizi AWS come [Amazon API Gateway](#), [AWS AppSync](#) [Amazon EventBridge](#) nell'architettura per utilizzare i contratti di servizio API nell'applicazione. Amazon API Gateway è un valido supporto per l'integrazione con i servizi AWS direttamente nativi e altri servizi Web. API Gateway supporta la [specifica OpenAPI](#) e il controllo delle versioni. AWS AppSync è un endpoint

[gestito da GraphQL](#) configurato definendo uno schema GraphQL per definire un'interfaccia di servizio per query, mutazioni e sottoscrizioni. Amazon EventBridge utilizza schemi di eventi per definire eventi e generare associazioni di codice per gli eventi.

Passaggi dell'implementazione

- Definisci innanzitutto un contratto per la tua API. Un contratto esprimerà le capacità di un'API e definirà oggetti e campi di dati fortemente tipizzati per l'input e l'output dell'API.
- Quando configuri le API in API Gateway, puoi importare ed esportare le specifiche OpenAPI per gli endpoint.
 - [L'importazione di una definizione OpenAPI](#) semplifica la creazione dell'API e può essere integrata con l'infrastruttura AWS come strumenti di codice come [AWS Serverless Application Model](#) e [AWS Cloud Development Kit \(AWS CDK\)](#).
 - [L'esportazione di una definizione API](#) semplifica l'integrazione con gli strumenti di test delle API e fornisce agli utenti di servizi una specifica di integrazione.
- Puoi definire e gestire le API GraphQL con AWS AppSync [mediante la definizione di un file di schema GraphQL](#) per generare l'interfaccia del contratto e semplificare l'interazione con modelli REST complessi, più tabelle di database o servizi legacy.
- [I progetti AWS Amplify](#) integrati con AWS AppSync generano file di query JavaScript fortemente tipizzati da utilizzare nell'applicazione, nonché una libreria client GraphQL AWS AppSync per le tabelle [Amazon DynamoDB](#).
- Quando si utilizzano eventi di servizio da Amazon EventBridge, gli eventi sono conformi agli schemi già esistenti nel registro degli schemi o definiti con la specifica OpenAPI. Con uno schema definito nel registro, puoi anche generare associazioni client dal contratto dello schema per integrare il codice con gli eventi.
- Estensione o definizione della versione dell'API. L'estensione di un'API è un'opzione più semplice quando si aggiungono campi che possono essere configurati con campi facoltativi o valori predefiniti per i campi obbligatori.
 - I contratti basati su JSON per protocolli come REST e GraphQL possono essere adatti per l'estensione del contratto.
 - I contratti basati su XML per protocolli come SOAP devono essere testati con gli utenti dei servizi per determinare se l'estensione del contratto è possibile.
- Quando esegui il controllo delle versioni di un'API, valuta la possibilità di implementare il controllo delle versioni proxy laddove un lato viene usato per supportare le versioni in modo che la logica possa essere gestita in un'unica base di codice.

- Con API Gateway puoi usare [mappature di richieste e risposte](#) per semplificare l'inclusione delle modifiche del contratto stabilendo un lato per fornire valori predefiniti per i nuovi campi o per eliminare i campi rimossi da una richiesta o una risposta. Con questo approccio, il servizio sottostante può avere un'unica base di codice.

Risorse

Best practice correlate:

- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)
- [REL03-BP02 Creazione di servizi focalizzati su domini e funzionalità aziendali specifici](#)
- [REL04-BP02 Implementazione di dipendenze "loosely coupled"](#)
- [REL05-BP03 Controllo e limitazione delle chiamate di ripetizione](#)
- [REL05-BP05 Impostazione dei timeout dei client](#)

Documenti correlati:

- [Cos'è un'interfaccia di programmazione dell'applicazione \(API\)?](#)
- [Implementazione di microservizi in AWS](#)
- [I compromessi dei microservizi](#)
- [Microservizi: una definizione di questo nuovo termine di architettura](#)
- [Microservizi in AWS](#)
- [Utilizzo delle estensioni API Gateway di OpenAPI](#)
- [Specifica OpenAPI](#)
- [GraphQL: schemi e tipi](#)
- [Associazioni di codice Amazon EventBridge](#)

Esempi correlati:

- [Amazon API Gateway: configurazione di una REST API mediante OpenAPI](#)
- [Applicazione CRUD da Amazon API Gateway a Amazon DynamoDB utilizzando OpenAPI](#)
- [Modelli di integrazione di applicazioni moderne in un'era serverless: integrazione dei servizi API Gateway](#)

- [Implementazione del controllo delle versioni API Gateway basato sull'intestazione con Amazon CloudFront](#)
- [AWS AppSync: creazione di un'applicazione client](#)

Video correlati:

- [Utilizzo di OpenAPI AWS SAM per la gestione di API Gateway](#)

Strumenti correlati:

- [Amazon API Gateway](#)
- [AWS AppSync](#)
- [Amazon EventBridge](#)

Progetta interazioni in un sistema distribuito per prevenire guasti

I sistemi distribuiti si basano sulle reti di comunicazione per interconnettere i componenti (ad esempio server o servizi). Il carico di lavoro deve funzionare in modo affidabile nonostante la perdita o la latenza dei dati in queste reti. I componenti del sistema distribuito devono funzionare in modo da non influire negativamente su altri componenti o sul carico di lavoro. Queste best practice prevengono gli errori e migliorano il tempo medio tra errori (MTBF).

Best practice

- [REL04-BP01 Identificazione del tipo di sistema distribuito da cui si dipende](#)
- [REL04-BP02 Implementazione di dipendenze "loosely coupled"](#)
- [REL04-BP03 Esecuzione di un lavoro costante](#)
- [REL04-BP04 Rendere tutte le risposte idempotenti](#)

REL04-BP01 Identificazione del tipo di sistema distribuito da cui si dipende

I sistemi distribuiti possono essere sincroni, asincroni o batch. I sistemi sincroni devono elaborare le richieste il più rapidamente possibile e comunicare tra loro effettuando chiamate di richiesta e risposta sincrone utilizzando i protocolli HTTP/S, REST o RPC (Remote Procedure Call). I sistemi asincroni comunicano tra loro scambiando i dati in modo asincrono tramite un servizio intermediario

senza associare singoli sistemi. I sistemi batch ricevono un grande volume di dati di input, eseguono i processi di dati automatizzati senza intervento umano e generano i dati di output.

Risultato desiderato: progetta un carico di lavoro che interagisca efficacemente con le dipendenze sincrone, asincrone e batch.

Anti-pattern comuni:

- Il carico di lavoro attende a tempo indeterminato una risposta dalle dipendenze, con eventuale timeout del client del carico di lavoro, senza informazioni sulla ricezione della richiesta.
- Il carico di lavoro utilizza una catena di sistemi dipendenti che si chiamano tra loro in modo sincrono. A tal fine, ogni sistema deve essere disponibile ed elaborare correttamente la richiesta prima che l'intera catena possa essere completata, con conseguenti comportamenti e disponibilità complessiva potenzialmente fragili.
- Il carico di lavoro comunica con le dipendenze in modo asincrono e si basa sul concetto di distribuzione garantita dei messaggi esattamente una volta, quando spesso è ancora possibile ricevere messaggi duplicati.
- Il carico di lavoro non utilizza strumenti di pianificazione batch adeguati e consente l'esecuzione simultanea dello stesso processo batch.

Vantaggi dell'adozione di questa best practice: è comune che un determinato carico di lavoro implementi uno o più stili di comunicazione tra sincrona, asincrona e batch. Questa best practice consente di identificare i diversi compromessi associati a ogni stile di comunicazione per rendere il carico di lavoro in grado di tollerare interruzioni in tutte le sue dipendenze.

Livello di rischio associato se questa best practice non fosse adottata: elevato

Guida all'implementazione

Le sezioni seguenti contengono le linee guida per l'implementazione generali e specifiche di ogni tipo di dipendenza.

Informazioni generali

- Assicurati che gli obiettivi del livello di servizio (SLO) di prestazioni e affidabilità offerti dalle dipendenze soddisfino i requisiti di prestazioni e affidabilità del tuo carico di lavoro.
- Utilizza [i servizi di osservabilità AWS](#) per [monitorare i tempi di risposta e i tassi di errore](#) e assicurarti che la dipendenza fornisca un servizio ai livelli necessari per il tuo carico di lavoro.

- Individua le potenziali sfide che il carico di lavoro può affrontare quando comunica con le dipendenze. I sistemi distribuiti [presentano un'ampia serie di sfide](#) che possono aumentare la complessità dell'architettura, gli oneri operativi e i costi. Le sfide più comuni includono latenza, interruzioni della rete, perdita dei dati, dimensionamento e ritardo nella replica dei dati.
- Implementa un solido sistema di gestione e [registrazione](#) degli errori per aiutarti a risolvere i problemi quando la dipendenza restituisce errori.

Dipendenza sincrona

Nelle comunicazioni sincrone, il carico di lavoro invia una richiesta alla dipendenza e blocca l'operazione in attesa della risposta. Quando la dipendenza riceve la richiesta, cerca di gestirla il prima possibile e invia una risposta al carico di lavoro. Una sfida significativa con la comunicazione sincrona è rappresentata dall'accoppiamento temporale, che richiede che il carico di lavoro e le sue dipendenze siano disponibili nello stesso momento. Quando il carico di lavoro deve comunicare in modo sincrono con le dipendenze, valuta le seguenti linee guida:

- Il carico di lavoro non deve fare affidamento su più dipendenze sincrone per eseguire una singola funzione. Questa catena di dipendenze aumenta la fragilità complessiva perché tutte le dipendenze nel percorso devono essere disponibili affinché la richiesta venga completata correttamente.
- Quando una dipendenza non è integra o non è disponibile, applica le strategie di gestione degli errori e riprova. Evita di usare un comportamento bimodale. Il comportamento bimodale si verifica quando il carico di lavoro presenta un comportamento diverso in modalità normale e in modalità di guasto. Per maggiori dettagli sul comportamento bimodale, consulta [REL11-BP05 Utilizzo della stabilità statica per evitare un comportamento bimodale](#).
- Tieni presente che anticipare l'errore (fail fast) è meglio che far aspettare il carico di lavoro. Ad esempio, in [AWS Lambda Developer Guide](#) viene descritto come gestire i tentativi e gli errori quando si richiamano le funzioni Lambda.
- Imposta i timeout per quando il carico di lavoro chiama la dipendenza. Questa tecnica evita di aspettare troppo a lungo o all'infinito una risposta. Per una spiegazione utile di questo problema, consulta [Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications](#).
- Riduci al minimo il numero di chiamate effettuate dal carico di lavoro alla dipendenza per soddisfare una singola richiesta. Le lunghe chiamate aumentano l'associazione e la latenza.

Dipendenza asincrona

Per disaccoppiare temporaneamente il carico di lavoro dalla dipendenza, è necessario che comunichino in modo asincrono. Con l'approccio asincrono, il carico di lavoro può continuare qualsiasi altra elaborazione senza dover attendere che la dipendenza o la catena di dipendenze invii la risposta.

Quando il carico di lavoro deve comunicare in modo asincrono con la dipendenza, tieni conto delle seguenti indicazioni:

- Determina in base al caso d'uso e ai requisiti se utilizzare la messaggistica o lo streaming di eventi. La [messaggistica](#) consente al carico di lavoro di comunicare con la dipendenza inviando e ricevendo messaggi tramite un broker di messaggi. Lo [streaming di eventi](#) consente al carico di lavoro e alle dipendenze di utilizzare un servizio di streaming per pubblicare e sottoscrivere gli eventi, distribuiti come flussi di dati continui che devono essere elaborati il prima possibile.
- La messaggistica e lo streaming di eventi gestiscono i messaggi in modo diverso, quindi devi stabilire i compromessi in base a:
 - **Priorità dei messaggi:** i broker di messaggi possono elaborare i messaggi ad alta priorità prima dei messaggi normali. Nello streaming di eventi, tutti i messaggi hanno la stessa priorità.
 - **Consumo di messaggi:** i broker di messaggi assicurano che gli utenti ricevano il messaggio. Gli utenti che utilizzano lo streaming di eventi devono tenere traccia dell'ultimo messaggio letto.
 - **Ordinamento di messaggi:** con la messaggistica non è garantita la ricezione dei messaggi nell'ordine esatto in cui vengono inviati, a meno che non si utilizzi un approccio first-in-first-out (FIFO). Lo streaming di eventi mantiene sempre l'ordine in cui i dati sono stati prodotti.
 - **Eliminazione di messaggi:** con la messaggistica, l'utente deve eliminare il messaggio dopo averlo elaborato. Il servizio di streaming di eventi aggiunge il messaggio a un flusso e lo conserva fino alla scadenza del periodo di conservazione del messaggio. Questa policy di eliminazione rende lo streaming di eventi adatto alla riproduzione dei messaggi.
- Definisci in che modo il carico di lavoro riconosce il completamento del lavoro della dipendenza. Ad esempio, quando il carico di lavoro richiama una [funzione Lambda in modo asincrono](#), Lambda inserisce l'evento in una coda e restituisce una risposta positiva senza informazioni aggiuntive. Una volta completata l'elaborazione, la funzione Lambda può [inviare il risultato a una destinazione](#), configurabile in base all'esito positivo o negativo.
- Crea il tuo carico di lavoro per gestire i messaggi duplicati utilizzando l'idempotenza. Con l'idempotenza i risultati del carico di lavoro non cambiano anche se il carico di lavoro viene generato più volte per lo stesso messaggio. È importante considerare che i servizi di [messaggistica](#)

o [streaming](#) recapitano nuovamente il messaggio se si verifica un errore di rete o se non è stata ricevuta la conferma.

- Se il carico di lavoro non riceve una risposta dalla dipendenza, deve inviare nuovamente la richiesta. Valuta la possibilità di limitare il numero di tentativi per preservare la CPU, la memoria e le risorse di rete del carico di lavoro al fine di gestire le altre richieste. Nella [documentazione di AWS Lambda](#) viene indicato come gestire gli errori per l'invocazione asincrona.
- Utilizza gli strumenti di osservabilità, debug e monitoraggio adeguati per gestire e usare la comunicazione asincrona del carico di lavoro con le relative dipendenze. Puoi utilizzare [Amazon CloudWatch](#) per monitorare i servizi di [messaggistica](#) e [streaming di eventi](#). Puoi anche ottimizzare il carico di lavoro con [AWS X-Ray](#) per [ottenere rapidamente approfondimenti](#) per la risoluzione dei problemi.

Dipendenza batch

I sistemi batch acquisiscono i dati di input, avviano una serie di processi per elaborarli e producono i dati di output, senza intervento manuale. A seconda delle dimensioni dei dati, i processi possono durare da minuti a diversi giorni in alcuni casi. Quando il carico di lavoro comunica con la dipendenza batch, tieni conto delle seguenti indicazioni:

- Definisci la finestra temporale in cui il carico di lavoro deve eseguire il processo batch. Puoi impostare un modello di ricorrenza per il carico di lavoro per richiamare il sistema batch, ad esempio ogni ora o alla fine di ogni mese.
- Determina la posizione dei dati di input e di output elaborati. Scegli un servizio di archiviazione, ad esempio [Amazon Simple Storage Services \(Amazon S3\)](#), [Amazon Elastic File System \(Amazon EFS\)](#) e [Amazon FSx for Lustre](#), per consentire al carico di lavoro di leggere e scrivere file su larga scala.
- Se il carico di lavoro deve richiamare più processi batch, puoi usare [AWS Step Functions](#) per semplificare l'orchestrazione dei processi batch eseguiti in AWS oppure on-premises. Questo [progetto di esempio](#) dimostra l'orchestrazione di processi batch utilizzando Step Functions, [AWS Batch](#) e Lambda.
- Monitora i processi batch per individuare eventuali anomalie, ad esempio un processo che richiede più tempo del dovuto per essere completato. Puoi utilizzare strumenti come [CloudWatch Container Insights](#) per monitorare ambienti e processi AWS Batch. In tal caso, il carico di lavoro interrompe l'inizio del processo successivo e comunica l'eccezione al team competente.

Risorse

Documenti correlati:

- [Cloud AWS Operations: Monitoraggio e osservabilità](#)
- [Amazon Builders' Library: Difficoltà dei sistemi distribuiti](#)
- [REL11-BP05 Utilizzo della stabilità statica per evitare un comportamento bimodale](#)
- [AWS Lambda Developer Guide: Error handling and automatic retries in AWS Lambda](#)
- [Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications](#)
- [Messaggistica in AWS](#)
- [Cosa sono i flussi di dati?](#)
- [AWS Lambda Developer Guide: Asynchronous invocation](#)
- [Domande frequenti su Amazon Simple Queue Service: Code FIFO](#)
- [Amazon Kinesis Data Streams Developer Guide: Handling Duplicate Records](#)
- [Amazon Simple Queue Service Developer Guide: Available CloudWatch metrics for Amazon SQS](#)
- [Amazon Kinesis Data Streams Developer Guide: Monitoring the Amazon Kinesis Data Streams Service with Amazon CloudWatch](#)
- [AWS X-Ray Developer Guide: AWS X-Ray concepts](#)
- [AWS Samples on GitHub: AWS Step functions Complex Orchestrator App](#)
- [AWS Batch User Guide: AWS Batch CloudWatch Container Insights](#)

Video correlati:

- [AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS \(COP310\)](#)

Strumenti correlati:

- [Amazon CloudWatch](#)
- [Amazon CloudWatch Logs](#)
- [AWS X-Ray](#)
- [Amazon Simple Storage Services \(Amazon S3\)](#)
- [Amazon Elastic File System \(Amazon EFS\)](#)
- [Amazon FSx for Lustre](#)

- [AWS Step Functions](#)
- [AWS Batch](#)

REL04-BP02 Implementazione di dipendenze "loosely coupled"

Le dipendenze come sistemi di accodamento, sistemi di streaming, flussi di lavoro e sistemi di bilanciamento del carico sono "loosely coupled" (con accoppiamento debole). L'accoppiamento debole aiuta a isolare il comportamento di un componente dagli altri componenti che dipendono da esso, aumentando la resilienza e l'agilità.

Nei sistemi con accoppiamento stretto, le modifiche a un componente possono richiedere modifiche agli altri componenti basati su di esso, con conseguente riduzione delle prestazioni di tutti i componenti. L'accoppiamento debole interrompe questa dipendenza, in modo che i componenti dipendenti debbano conoscere solo l'interfaccia con versione e pubblicata. L'implementazione di un accoppiamento debole tra dipendenze isola un errore all'interno di una dipendenza affinché non influenzi l'altra.

L'accoppiamento debole consente di modificare il codice o aggiungere funzionalità a un componente riducendo al minimo il rischio per gli altri componenti che dipendono da esso. Consente inoltre una resilienza granulare a livello di componente in cui è possibile impiegare la scalabilità orizzontale o persino modificare l'implementazione sottostante della dipendenza.

Per migliorare ulteriormente la resilienza tramite accoppiamento debole, rendi le interazioni dei componenti asincrone laddove possibile. Questo modello è idoneo a qualsiasi interazione che non richieda una risposta immediata e laddove la conferma della registrazione di una richiesta sia sufficiente. Include un componente che genera eventi e un altro che li utilizza. I due componenti non si integrano tramite un'interazione diretta point-to-point, ma in genere attraverso un livello di archiviazione intermedio durevole, come una coda Amazon SQS o una piattaforma di dati in streaming come Amazon Kinesis o AWS Step Functions.

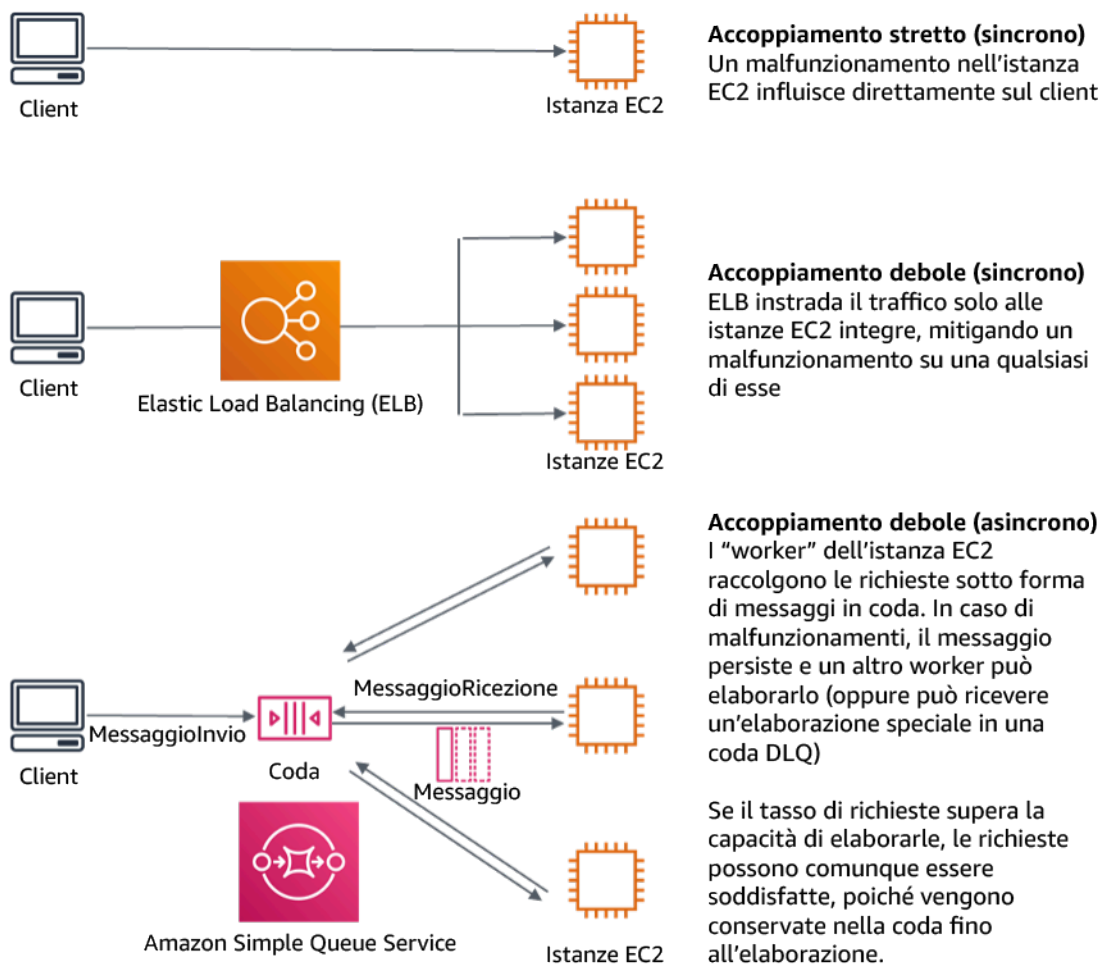


Figura 4. Le dipendenze come i sistemi di accodamento e i sistemi di bilanciamento del carico sono "loosely coupled"

Le code Amazon SQS ed Elastic Load Balancer sono solo due modi per aggiungere un livello intermedio per l'accoppiamento debole. Le architetture basate su eventi possono anche essere create in Cloud AWS utilizzando Amazon EventBridge, che può astrarre i client (produttori di eventi) dai servizi a cui fanno affidamento (consumatori di eventi). Amazon Simple Notification Service (Amazon SNS) è una soluzione efficace quando hai bisogno di messaggistica da-molti-a-molti, dalla velocità di trasmissione effettiva elevata e basata su push. Utilizzando gli argomenti di Amazon SNS, i sistemi di pubblicazione possono inviare messaggi a un numero elevato di endpoint sottoscrittori per l'elaborazione parallela.

Mentre le code offrono diversi vantaggi, nella maggior parte dei sistemi hard real-time, le richieste più vecchie di una soglia temporale (spesso secondi) dovrebbero essere considerate obsolete (il client ha abbandonato e non è più in attesa di una risposta) e non elaborate. In questo modo, è possibile elaborare invece le richieste più recenti (e probabilmente ancora valide).

Risultato desiderato: l'implementazione di dipendenze con accoppiamento debole consente di ridurre al minimo l'area esposta ai guasti a livello di componente e ciò aiuta a diagnosticare e risolvere i problemi. Inoltre, semplifica i cicli di sviluppo, consentendo ai team di implementare le modifiche a livello modulare senza pregiudicare le prestazioni di altri componenti che dipendono da esso. Questo approccio offre la possibilità di impiegare la scalabilità orizzontale a livello di componente in base al fabbisogno di risorse, nonché di utilizzare un componente che contribuisce alla competitività in termini di costi.

Anti-pattern comuni:

- Implementazione di un carico di lavoro monolitico.
- Invocazione diretta di API tra livelli di carico di lavoro senza funzionalità di failover o elaborazione asincrona della richiesta.
- Accoppiamento stretto utilizzando dati condivisi. I sistemi con accoppiamento debole dovrebbero evitare di condividere i dati tramite database condivisi o altre forme di archiviazione dei dati con accoppiamento stretto, che possono reintrodurre l'accoppiamento stretto e compromettere la scalabilità.
- Ignorare la contropressione. Il carico di lavoro dovrebbe essere in grado di rallentare o arrestare i dati in arrivo quando un componente non è in grado di elaborarli alla stessa velocità.

Vantaggi dell'adozione di questa best practice: l'accoppiamento debole aiuta a isolare il comportamento di un componente dagli altri componenti che dipendono da esso, aumentando la resilienza e l'agilità. L'errore in un componente è isolato dagli altri.

Livello di rischio associato alla mancata adozione di questa best practice: elevato

Guida all'implementazione

Implementazione di dipendenze "loosely coupled". Esistono varie soluzioni che consentono di creare applicazioni con accoppiamento debole. Queste includono, ad esempio, servizi per l'implementazione di code completamente gestite, flussi di lavoro automatizzati, reazione agli eventi e API, che possono aiutare a isolare il comportamento dei componenti dagli altri componenti e, di conseguenza, aumentare la resilienza e l'agilità.

- Crea architetture basate su eventi: [Amazon EventBridge](#) ti aiuta a creare architetture basate sugli eventi con accoppiamento debole e distribuite.
- Implementazione di code nei sistemi distribuiti: è possibile utilizzare [Amazon Simple Queue Service \(Amazon SQS\)](#) per integrare e disaccoppiare i sistemi distribuiti.

- Containerizza i componenti come microservizi: i [microservizi](#) consentono ai team di creare applicazioni composte da piccoli componenti indipendenti che comunicano tramite API ben definite. [Amazon Elastic Container Service \(Amazon ECS\)](#) e [Amazon Elastic Kubernetes Service \(Amazon EKS\)](#) possono aiutarti a iniziare a utilizzare più rapidamente i container.
- Gestisci i flussi di lavoro con Step Functions: [Step Functions](#) semplifica il coordinamento di più servizi AWS in flussi di lavoro flessibili.
- Sfrutta le architetture di messaggistica publish-subscribe (pub/sub): [Amazon Simple Notification Service \(Amazon SNS\)](#) fornisce il servizio di consegna dei messaggi dagli editori agli abbonati (noti anche come produttori e consumatori).

Passaggi dell'implementazione

- I componenti in un'architettura basata su eventi vengono avviati dagli eventi. Gli eventi sono azioni che si verificano in un sistema, ad esempio un utente che aggiunge un articolo a un carrello. Quando un'azione ha successo, viene generato un evento che attiva il successivo componente del sistema.
 - [Creazione di applicazioni basate su eventi con Amazon EventBridge](#)
 - [AWS re:Invent 2022 - Designing Event-Driven Integrations using Amazon EventBridge](#)
- I sistemi di messaggistica distribuiti sono composti da tre parti principali che devono essere implementate per un'architettura basata su code. Includono i componenti del sistema distribuito, la coda utilizzata per il disaccoppiamento (distribuita su server Amazon SQS) e i messaggi in coda. Un sistema tipico prevede produttori che inviano il messaggio alla coda e il consumatore che riceve il messaggio dalla coda. La coda archivia i messaggi su più server Amazon SQS per garantire la ridondanza.
 - [Architettura Amazon SQS di base](#)
 - [Invia messaggi tra applicazioni distribuite con Amazon Simple Queue Service](#)
- I microservizi, se ben utilizzati, migliorano la manutenibilità e aumentano la scalabilità, poiché i componenti ad accoppiamento debole sono gestiti da team indipendenti. Consentono inoltre l'isolamento dei comportamenti in un unico componente in caso di modifiche.
 - [Implementazione di microservizi in AWS](#)
 - [Let's Architect! Architecting microservices with containers](#)
- Con AWS Step Functions è possibile eseguire moltissime operazioni, ad esempio creare applicazioni distribuite, automatizzare i processi e orchestrare microservizi. L'orchestrazione di

più componenti in un flusso di lavoro automatizzato consente di disaccoppiare le dipendenze nell'applicazione.

- [Create a Serverless Workflow with AWS Step Functions and AWS Lambda](#)
- [Nozioni di base su AWS Step Functions](#)

Risorse

Documenti correlati:

- [Amazon EC2: Ensuring Idempotency](#)
- [The Amazon Builders' Library: Difficoltà dei sistemi distribuiti](#)
- [The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee](#)
- [What is Amazon EventBridge?](#)
- [What is Amazon Simple Queue Service?](#)
- [Break up with your monolith](#)
- [Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS](#)
- [Architettura Amazon SQS di base](#)
- [Architettura basata su code](#)

Video correlati:

- [AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge \(MAD205\)](#)
- [AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 \(includes loose coupling, constant work, static stability\)](#)
- [AWS re:Invent 2019: Moving to event-driven architectures \(SVS308\)](#)
- [AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda \(API304\)](#)
- [AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda](#)
- [AWS re:Invent 2022 - Designing event-driven integrations using Amazon EventBridge](#)
- [AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices](#)

REL04-BP03 Esecuzione di un lavoro costante

I sistemi possono fallire quando si verificano modifiche rapide e di grandi dimensioni nel carico. Ad esempio, se il carico di lavoro effettua un controllo dell'integrità di migliaia di server deve inviare ogni volta lo stesso payload delle dimensioni (uno snapshot completo dello stato corrente). Indipendentemente dal fatto che non ci siano server guasti, o che lo siano tutti, il sistema di controllo dello stato esegue un lavoro costante con modifiche rapide e di piccole dimensioni.

Ad esempio, se il sistema di controllo dello stato monitora 100.000 server, il carico su di esso è nominale al di sotto del tasso di errore normalmente basso del server. Tuttavia, se un evento importante rendesse la metà di questi server non integra, il sistema di controllo dello stato sarebbe sovraccarico nel tentativo di aggiornare i sistemi di notifica e comunicare lo stato con i client. Pertanto, il sistema di controllo dello stato dovrebbe ogni volta inviare lo snapshot completo dello stato corrente. 100.000 stati di integrità del server, ciascuno rappresentato da un bit, sarebbero solo un payload di 12,5 KB. Indipendentemente dal fatto che non ci siano server guasti, o che lo siano tutti, il sistema di controllo dello stato esegue un lavoro costante e le modifiche rapide e di grandi dimensioni non rappresentano una minaccia per la stabilità del sistema. Questo è in realtà il modo in cui Amazon Route 53 gestisce i controlli dell'integrità degli endpoint (come gli indirizzi IP) per stabilire come gli utenti finali vengono instradati verso di loro.

Livello di rischio associato se questa best practice non fosse adottata: Bassa

Guida all'implementazione

- Esegui un lavoro costante in modo che i sistemi non falliscano quando si verificano cambiamenti rapidi e significativi nel carico.
- Implementazione di dipendenze "loosely coupled". Le dipendenze come sistemi di accodamento, sistemi di streaming, flussi di lavoro e sistemi di bilanciamento del carico sono "loosely coupled" (con accoppiamento debole). L'accoppiamento debole aiuta a isolare il comportamento di un componente dagli altri componenti che dipendono da esso, aumentando la resilienza e l'agilità.
 - [The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee](#)
 - [AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small \(Chiudere i cicli e aprire le menti: come prendere il controllo dei sistemi, grandi e piccoli\), include lavoro costante \(ARC337\)](#)
- Per l'esempio di un sistema di controllo dell'integrità che monitora 100.000 server, progetta i carichi di lavoro in modo che le dimensioni dei payload rimangano costanti indipendentemente dal numero di successi o di fallimenti.

Risorse

Documenti correlati:

- [Amazon EC2: garantire l'idempotenza](#)
- [The Amazon Builders' Library: Difficoltà dei sistemi distribuiti](#)
- [The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee](#)

Video correlati:

- [AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge \(Introduzione alle architetture guidate dagli eventi e ad Amazon EventBridge\) \(MAD205\)](#)
- [AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small \(Chiudere i cicli e aprire le menti: come prendere il controllo dei sistemi, grandi e piccoli\), include lavoro costante \(ARC337\)](#)
- [AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small \(Chiudere i cicli e aprire le menti: come prendere il controllo dei sistemi, grandi e piccoli\), sono inclusi accoppiamento debole, lavoro costante e stabilità statica \(ARC337\)](#)
- [AWS re:Invent 2019: passare alle architetture basate sugli eventi \(SVS308\)](#)

REL04-BP04 Rendere tutte le risposte idempotenti

Un servizio idempotente promette il completamento di ogni richiesta esattamente una volta, in modo tale che effettuare più richieste identiche abbia lo stesso effetto di effettuare una singola richiesta. Un servizio idempotente semplifica ad un client l'implementazione di nuovi tentativi senza temere che una richiesta venga elaborata erroneamente più volte. Per eseguire questa operazione, i client possono inviare richieste API con un token di idempotenza: viene utilizzato lo stesso token ogni volta che si ripete la richiesta. Un'API del servizio idempotente utilizza il token per restituire una risposta identica a quella restituita la prima volta che la richiesta è stata completata.

In un sistema distribuito, è facile eseguire un'operazione al massimo una volta (il client effettua una sola richiesta) o almeno una volta (la richiesta continua finché il client non ottiene la conferma dell'esito positivo). Tuttavia, è difficile garantire che un'operazione sia idempotente, il che significa che viene eseguita esattamente una volta, in modo tale che effettuare più richieste identiche abbia lo stesso effetto di effettuare una singola richiesta. Utilizzando i token di idempotenza nelle API, i servizi

possono ricevere una richiesta di mutazione una o più volte senza creare record duplicati o effetti collaterali.

Livello di rischio associato se questa best practice non fosse adottata: Medium

Guida all'implementazione

- Rendi tutte le risposte idempotenti. Un servizio idempotente promette il completamento di ogni richiesta esattamente una volta, in modo tale che effettuare più richieste identiche abbia lo stesso effetto di effettuare una singola richiesta.
- I client possono inviare richieste API con un token di idempotenza: viene utilizzato lo stesso token ogni volta che si ripete la richiesta. Un'API del servizio idempotente utilizza il token per restituire una risposta identica a quella restituita la prima volta che la richiesta è stata completata.
 - [Amazon EC2: Ensuring Idempotency \(EC2: garantire l'idempotenza\)](#)

Risorse

Documenti correlati:

- [Amazon EC2: Ensuring Idempotency \(EC2: garantire l'idempotenza\)](#)
- [The Amazon Builders' Library: Difficoltà dei sistemi distribuiti](#)
- [The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee](#)

Video correlati:

- [AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge \(AWS New York Summit 2019: Introduzione alle architetture guidate dagli eventi e ad Amazon EventBridge\) \(MAD205\)](#)
- [AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small \(Chiudere i cicli e aprire le menti: come prendere il controllo dei sistemi, grandi e piccoli\) \(sono inclusi accoppiamento debole, lavoro costante e stabilità statica\) \(ARC337\)](#)
- [AWS re:Invent 2019: Moving to event-driven architectures \(Passare alle architetture basate sugli eventi\) \(SVS308\)](#)

Progettazione di interazioni in un sistema distribuito per mitigare o affrontare gli errori

I sistemi distribuiti si basano sulle reti di comunicazione per interconnettere i componenti (ad esempio server o servizi). Il carico di lavoro deve funzionare in modo affidabile nonostante la perdita o la latenza dei dati su queste reti. I componenti del sistema distribuito devono funzionare in modo da non influire negativamente su altri componenti o sul carico di lavoro. Queste best practice consentono ai carichi di lavoro di affrontare stress o guasti, recuperare più rapidamente e mitigare l'impatto di tali problemi. Il risultato è un miglioramento del tempo medio di ripristino (MTTR).

Queste best practice prevencono gli errori e migliorano il tempo medio tra errori (MTBF).

Best practice

- [REL05-BP01 Implementazione della normale riduzione delle prestazioni per trasformare le dipendenze forti applicabili in dipendenze deboli](#)
- [REL05-BP02 Richieste di limitazione \(della larghezza di banda della rete\)](#)
- [REL05-BP03 Controllo e limitazione delle chiamate di ripetizione](#)
- [REL05-BP04 Anticipazione degli errori e limitazione delle code](#)
- [REL05-BP05 Impostazione dei timeout dei client](#)
- [REL05-BP06 Utilizzo dei sistemi stateless laddove possibile](#)
- [REL05-BP07 Implementazione di leve di emergenza](#)

REL05-BP01 Implementazione della normale riduzione delle prestazioni per trasformare le dipendenze forti applicabili in dipendenze deboli

I componenti dell'applicazione devono continuare a svolgere la loro funzione principale anche se le dipendenze non sono disponibili. Potrebbero fornire dati leggermente obsoleti, dati alternativi o addirittura nessun dato. Ciò garantisce che la funzionalità complessiva del sistema sia ostacolata solo in minima parte da errori localizzati, garantendo al contempo il valore aziendale intrinseco.

Risultato desiderato: quando le dipendenze di un componente non sono integre, il componente stesso può comunque funzionare, anche se non in modo ottimale. Le modalità di errore dei componenti devono essere considerate come funzionamenti normali. I flussi di lavoro devono essere progettati in modo tale che questi errori non conducano a un fallimento completo o almeno a stati prevedibili e recuperabili.

Anti-pattern comuni:

- Mancata identificazione della funzionalità aziendale di base necessaria. Mancata verifica del funzionamento dei componenti anche in caso di errori di dipendenza.
- Mancata restituzione di dati sugli errori o quando solo una delle dipendenze non è disponibile e possono comunque essere restituiti risultati parziali.
- Creazione di uno stato incoerente quando una transazione fallisce parzialmente.
- Mancata disponibilità di alternative per accedere a un archivio di parametri centralizzato.
- Invalidare o svuotare lo stato locale a seguito di un aggiornamento non riuscito senza considerare le conseguenze di tale operazione.

Vantaggi dell'adozione di questa best practice: la normale riduzione delle prestazioni migliora la disponibilità del sistema nel suo complesso e conserva la funzionalità delle funzioni più importanti anche in caso di errori.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

L'implementazione di una normale riduzione delle prestazioni aiuta a ridurre al minimo l'impatto degli errori di dipendenza sul funzionamento dei componenti. Idealmente, un componente rileva gli errori nelle dipendenze e trova soluzioni alternative in modo da avere un impatto minimo sugli altri componenti o clienti.

Progettare per una normale riduzione delle prestazioni significa considerare le potenziali modalità di errore durante la progettazione delle dipendenze. Per ogni modalità di errore, disponi di un modo per fornire la maggior parte delle funzionalità o almeno quelle più critiche del componente a chiamanti o clienti. Queste considerazioni possono diventare requisiti aggiuntivi testabili e verificabili. Idealmente, un componente è in grado di svolgere la sua funzione principale in modo accettabile anche in caso di errore di una o più dipendenze.

Questa è una discussione di carattere tanto commerciale quanto tecnico. Tutti i requisiti aziendali sono importanti e devono essere soddisfatti, se possibile. Tuttavia, ha ancora senso chiedersi cosa dovrebbe succedere quando non tutti i requisiti possono essere soddisfatti. Un sistema può essere progettato per essere disponibile e coerente, ma nelle circostanze in cui è necessario eliminare un requisito, qual è quello più importante? Per l'elaborazione dei pagamenti, potrebbe essere la coerenza. Per un'applicazione in tempo reale, potrebbe essere la disponibilità. Per un sito Web rivolto ai clienti, la risposta può dipendere dalle aspettative dei clienti.

Il significato di ciò dipende dai requisiti del componente e da ciò che dovrebbe essere considerato la sua funzione principale. Ad esempio:

- un sito di e-commerce potrebbe visualizzare dati provenienti da più sistemi diversi, come consigli personalizzati, prodotti con il punteggio più alto e lo stato degli ordini dei clienti sulla pagina di destinazione. Quando in un sistema upstream si verifica un errore, ha comunque senso mostrare tutto il resto, invece di mostrare una pagina di errore a un cliente.
- Un componente che esegue la scrittura in batch può continuare a elaborare un batch se una delle singole operazioni fallisce. Dovrebbe essere semplice implementare un meccanismo di ripetizione dei tentativi. A tale scopo, è sufficiente restituire al chiamante informazioni su quali operazioni hanno avuto successo, quali e perché non sono riuscite, oppure inserendo le richieste non riuscite in una coda DLQ per implementare nuovi tentativi asincroni. Anche le informazioni sulle operazioni non riuscite devono essere registrate.
- Un sistema che elabora le transazioni deve verificare che vengano eseguiti tutti gli aggiornamenti o nessun aggiornamento. Per le transazioni distribuite, il modello Saga può essere utilizzato per ripristinare le operazioni precedenti nel caso in cui fallisca un'operazione successiva della stessa transazione. Qui, la funzione principale è mantenere la coerenza.
- I sistemi critici dal punto di vista temporale dovrebbero essere in grado di gestire le dipendenze che non rispondono in modo tempestivo. In questi casi, è possibile utilizzare lo schema dell'interruttore. Quando inizia a verificarsi il timeout delle risposte di una dipendenza, il sistema può passare a uno stato chiuso in cui non vengono effettuate chiamate aggiuntive.
- Un'applicazione può leggere i parametri da un archivio di parametri. Può essere utile creare immagini di container con un set di parametri predefinito e utilizzarli nel caso in cui l'archivio dei parametri non sia disponibile.

Si noti che i percorsi seguiti in caso di errore dei componenti devono essere testati e devono essere significativamente più semplici del percorso primario. In genere, [è consigliabile evitare il fallback](#).

Passaggi dell'implementazione

Identifica le dipendenze esterne e interne. Considera i tipi di errore che si possono verificare nelle dipendenze. Considera i modi per ridurre al minimo l'impatto negativo sui sistemi upstream e downstream e sui clienti durante questi errori.

Di seguito è riportato un elenco di dipendenze e di come ridurre normalmente le prestazioni quando si verifica un errore a livello di dipendenze:

1. Errore parziale delle dipendenze: un componente può effettuare più richieste ai sistemi downstream, sia come richieste multiple a un sistema sia come richiesta a più sistemi. A seconda del contesto aziendale, possono essere appropriate diverse modalità di gestione (per maggiori dettagli, consulta gli esempi precedenti nella Guida all'implementazione).
2. Un sistema downstream non è in grado di elaborare le richieste a causa del carico elevato: se le richieste a un sistema downstream hanno costantemente esito negativo, non ha senso continuare a riprovare. Ciò può creare un carico aggiuntivo su un sistema già sovraccarico e rendere più difficile il ripristino. Qui è possibile utilizzare lo schema dell'interruttore, che monitora le chiamate non riuscite a un sistema downstream. Se un numero elevato di chiamate ha esito negativo, interromperà l'invio di altre richieste al sistema downstream e solo occasionalmente lascerà passare le chiamate per verificare se il sistema downstream è nuovamente disponibile.
3. Un archivio di parametri non è disponibile: per trasformare un archivio di parametri, è possibile utilizzare la cache delle dipendenze a protezione debole o i valori predefiniti integri inclusi nelle immagini del container o del computer. Tieni presente che queste impostazioni predefinite devono essere costantemente aggiornate e incluse nelle suite di test.
4. Un servizio di monitoraggio o altra dipendenza non funzionale non è disponibile: se un componente non è in grado di inviare a intermittenza log, metriche o tracce a un servizio di monitoraggio centralizzato, spesso è meglio continuare a eseguire le funzioni aziendali come al solito. Non registrare o eseguire il push delle metriche in modo invisibile all'utente per un lungo periodo di tempo spesso non è una procedura accettabile. Inoltre, in alcuni casi d'uso potrebbero essere necessari dati di controllo completi per soddisfare i requisiti di conformità.
5. Un'istanza primaria di un database relazionale potrebbe non essere disponibile: Amazon Relational Database Service, come quasi tutti i database relazionali, può avere solo un'istanza di scrittura primaria. Questo crea un unico punto di errore per i carichi di lavoro di scrittura e rende più difficile il dimensionamento. Questo problema può essere parzialmente mitigato utilizzando una configurazione Multi-AZ per una disponibilità elevata o Amazon Aurora Serverless per un migliore dimensionamento. Per requisiti di disponibilità molto elevati, può essere logico non fare affatto affidamento sull'istanza di scrittura primaria. Per le query che si limitano a leggere, è possibile utilizzare repliche di lettura, che forniscono ridondanza e la possibilità di dimensionare non solo verticalmente, ma anche orizzontalmente. Le operazioni di scrittura possono essere memorizzate nel buffer, ad esempio in una coda Amazon Simple Queue Service, in modo che le richieste di scrittura dei clienti possano comunque essere accettate anche se l'istanza primaria non è temporaneamente disponibile.

Risorse

Documenti correlati:

- [Amazon API Gateway: throttling delle richieste API per migliorare la velocità di trasmissione effettiva](#)
- [CircuitBreaker \(riepilogo dal libro Circuit Breaker da "Release It!"\)](#)
- [Ripetizione dei tentativi in caso di errore e backoff esponenziale in AWS](#)
- [Michael Nygard "Release It! Design and Deploy Production-Ready Software"](#)
- [The Amazon Builders' Library: Evitare il fallback nei sistemi distribuiti](#)
- [The Amazon Builders' Library: Evitare insormontabili backlog di code](#)
- [The Amazon Builders' Library: Sfide e strategie del caching](#)
- [The Amazon Builders' Library: Timeout, nuovi tentativi e backoff con jitter](#)

Video correlati:

- [Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library \(Presentazione della libreria dei costruttori di Amazon\) \(DOP328\)](#)

Esempi correlati:

- [Corso Well-Architected: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability](#)

REL05-BP02 Richieste di limitazione (della larghezza di banda della rete)

Limita le richieste per mitigare l'esaurimento delle risorse dovuto ad aumenti imprevisti della domanda. Le richieste inferiori alla percentuale di limitazione (della larghezza di banda della rete) vengono elaborate mentre quelle che superano il limite definito vengono rifiutate con un messaggio che indica che la richiesta è stata limitata.

Risultato desiderato: i picchi di volume di grandi dimensioni dovuti a improvvisi aumenti del traffico dei clienti, attacchi di flooding o tempeste di ripetizioni dei tentativi sono mitigati dalla limitazione (della larghezza di banda della rete) delle richieste, che consente ai carichi di lavoro di continuare la normale elaborazione del volume di richieste supportato.

Anti-pattern comuni:

- Le accelerazioni degli endpoint API non sono implementate o vengono implementate in base ai valori predefiniti senza considerare i volumi previsti.
- Gli endpoint delle API non sono sottoposti a test di carico né i limiti relativi alla limitazione (della larghezza di banda della rete) vengono testati.
- Limitazione delle tariffe delle richieste senza considerare le dimensioni o la complessità delle richieste.
- Verifica delle percentuali massime di richieste o delle dimensioni massime delle richieste, senza però testarle congiuntamente.
- Le risorse non vengono fornite entro gli stessi limiti stabiliti durante i test.
- I piani di utilizzo non sono stati configurati o considerati per gli utenti di API Application to Application (A2A).
- Gli utenti di code con dimensionamento orizzontale non hanno configurato le impostazioni di simultaneità massima.
- La limitazione della velocità per indirizzo IP non è stata implementata.

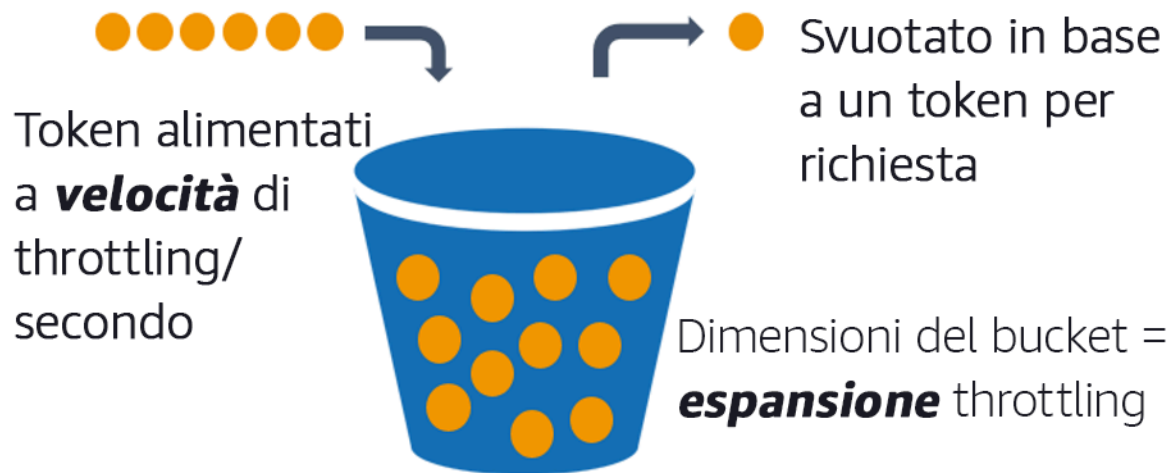
Vantaggi dell'adozione di questa best practice: i carichi di lavoro che stabiliscono limiti di accelerazione sono in grado di funzionare normalmente ed elaborare correttamente il caricamento delle richieste accettate in presenza di picchi di volume imprevisti. I picchi improvvisi o prolungati di richieste alle API e alle code vengono limitati e non esauriscono le risorse di elaborazione delle richieste. I limiti tariffari vincolano i richiedenti in modo che elevati volumi di traffico provenienti da un utente di un indirizzo IP o API specifico non esauriscano le risorse e influiscano sugli altri utenti.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

I servizi devono essere progettati per elaborare una capacità nota di richieste; tale capacità può essere stabilita mediante test di carico. Se le percentuali di arrivo delle richieste superano i limiti, la risposta appropriata segnala che una richiesta è stata limitata. Ciò consente all'utente di gestire l'errore e riprovare in un secondo momento.

Quando il servizio richiede un'implementazione della limitazione (della larghezza di banda della rete), prendi in considerazione l'implementazione dell'algoritmo token bucket, in cui un token conta come una richiesta. I token vengono alimentati a una specifica velocità di throttling al secondo e svuotati in modo asincrono in base a un token per richiesta.



Algoritmo token bucket.

[Amazon API Gateway](#) implementa l'algoritmo token bucket in base ai limiti dell'account e della regione e può essere configurato per cliente con piani di utilizzo. Inoltre, [Amazon Simple Queue Service \(Amazon SQS\)](#) e [Amazon Kinesis](#) possono memorizzare le richieste nel buffer per livellare la frequenza delle richieste e consentire percentuali di limitazione più elevati per le richieste che possono essere soddisfatte. Infine, puoi implementare la limitazione della velocità con [AWS WAF](#) per limitare utenti di API specifici che generano carichi insolitamente elevati.

Passaggi dell'implementazione

Puoi configurare API Gateway con limiti di limitazione (della larghezza di banda della rete) per le tue API e restituire errori 429 - Troppe richieste in caso di superamento dei limiti. Puoi utilizzare AWS WAF con gli endpoint API Gateway e AWS AppSync per abilitare la limitazione della velocità per indirizzo IP. Inoltre, laddove il sistema può tollerare l'elaborazione asincrona, è possibile inserire i messaggi in una coda o in un flusso per velocizzare le risposte ai client del servizio, il che consente di aumentare le velocità.

Con l'elaborazione asincrona, una volta configurato Amazon SQS come origine degli eventi per AWS Lambda, è possibile [configurare la simultaneità massima](#) per evitare che percentuali elevate di eventi consumino la quota di esecuzione simultanea disponibile dell'account necessaria per altri servizi nel carico di lavoro o nell'account.

Sebbene API Gateway fornisca un'implementazione gestita dell'algoritmo token bucket, nei casi in cui non sia possibile utilizzare API Gateway, puoi sfruttare le implementazioni open source specifiche del

linguaggio (consulta gli esempi correlati nella sezione Risorse) dell'algoritmo token bucket per i tuoi servizi.

- Analizza e configura [i valori di limitazione \(della larghezza di banda della rete\) API Gateway](#) a livello di account per regione, API per fase e chiave API per livelli del piano di utilizzo.
- Applica le [regole di limitazione \(della larghezza di banda della rete\) AWS WAF](#) sugli endpoint API Gateway e AWS AppSync come prevenzione degli attacchi flood e per bloccare gli IP pericolosi. Le regole di limitazione (della larghezza di banda della rete) possono anche essere configurate su chiavi API AWS AppSync per gli utenti A2A.
- Valuta se hai bisogno di più controllo sulla limitazione della larghezza di banda della rete rispetto al controllo sulla limitazione della velocità per le API AWS AppSync e, in tal caso, configura un API Gateway davanti all'endpoint AWS AppSync.
- Quando le code Amazon SQS sono impostate come trigger per gli utenti della coda Lambda, imposta [la simultaneità massima](#) su un valore che elabora in misura sufficiente a soddisfare gli obiettivi dei livelli di servizio ma non consuma i limiti di simultaneità che influiscono su altre funzioni Lambda. Valuta la possibilità di impostare la simultaneità riservata su altre funzioni Lambda nello stesso account e nella stessa regione quando utilizzi le code con Lambda.
- Utilizza API Gateway con integrazioni di servizi native per Amazon SQS o Kinesis per memorizzare le richieste nel buffer.
- Se non puoi utilizzare API Gateway, consulta le librerie specifiche della lingua per implementare l'algoritmo token bucket per il tuo carico di lavoro. Controlla la sezione degli esempi e cerca una libreria adatta.
- Verifica i limiti che intendi impostare o che prevedi di incrementare e documenta i limiti testati.
- Non aumentare i limiti oltre i valori stabiliti durante i test. Quando si aumenta un limite, verifica che le risorse assegnate siano equivalenti o superiori a quelle degli scenari di test prima di applicare l'aumento.

Risorse

Best practice correlate:

- [REL04-BP03 Esecuzione di un lavoro costante](#)
- [REL05-BP03 Controllo e limitazione delle chiamate di ripetizione](#)

Documenti correlati:

- [Amazon API Gateway: throttling delle richieste API per migliorare la velocità di trasmissione effettiva](#)
- [AWS WAF: istruzione delle regole basate sulla frequenza](#)
- [Introduzione alla simultaneità massima di AWS Lambda in caso di utilizzo Amazon SQS come origine degli eventi](#)
- [AWS Lambda: simultaneità massima](#)

Esempi correlati:

- [Le tre regole AWS WAF più importanti basate sulla velocità](#)
- [Java Bucket4j](#)
- [Algoritmo token bucket per Python](#)
- [Algoritmo token bucket a livello di nodo](#)
- [Limitazione della velocità di threading del sistema .NET](#)

Video correlati:

- [Implementazione delle best practice di sicurezza dell'API GraphQL con AWS AppSync](#)

Strumenti correlati:

- [Amazon API Gateway](#)
- [AWS AppSync](#)
- [Amazon SQS](#)
- [Amazon Kinesis](#)
- [AWS WAF](#)

REL05-BP03 Controllo e limitazione delle chiamate di ripetizione

Utilizza il backoff esponenziale per rieseguire le richieste a intervalli progressivamente più lunghi tra i singoli nuovi tentativi. Introduci il jitter tra i tentativi per randomizzare gli intervalli di ripetizione. Limita il numero massimo di tentativi.

Risultato desiderato: I componenti tipici di un sistema software distribuito includono server, sistemi di bilanciamento del carico, database e server DNS. Durante il normale funzionamento, questi

componenti possono rispondere alle richieste con errori temporanei o limitati e anche errori che sarebbero persistenti indipendentemente dai nuovi tentativi. Quando i client effettuano richieste ai servizi, le richieste consumano risorse tra cui memoria, thread, connessioni, porte o altre risorse limitate. Controllare e limitare i nuovi tentativi è una strategia per liberare risorse e ridurre al minimo il consumo in modo che i componenti del sistema sottoposti a stress non vengano sovraccaricati.

Quando vanno in timeout o ricevono risposte di errore, le richieste client devono decidere se eseguire o meno nuovi tentativi. Se vengono eseguiti nuovi tentativi, questi verranno eseguiti con un backoff esponenziale con jitter e un numero massimo di tentativi. Di conseguenza, i servizi e i processi back-end riducono il carico e i tempi di riparazione automatica, con un conseguente ripristino più rapido e una corretta gestione delle richieste.

Anti-pattern comuni:

- Implementazione di nuovi tentativi senza aggiungere il backoff esponenziale, il jitter e il numero massimo di tentativi. Il backoff e il jitter aiutano a evitare picchi di traffico artificiali dovuti a tentativi involontariamente coordinati a intervalli standard.
- Implementazione di nuovi tentativi senza testarne gli effetti o assunzione che i nuovi tentativi siano già integrati in un SDK senza testare gli scenari di ripetizione dei tentativi.
- La mancata comprensione dei codici di errore pubblicati nelle dipendenze porta a ritentare tutti gli errori, compresi quelli la cui causa è chiara e indica la mancanza di autorizzazione, un errore di configurazione o un'altra condizione che prevedibilmente non si risolverà senza un intervento manuale.
- Mancata gestione delle best practice di osservabilità, compresi il monitoraggio e la segnalazione di ripetuti guasti del servizio, in modo che i problemi sottostanti siano resi noti e possano essere risolti.
- Sviluppo di meccanismi di ripetizione personalizzati quando le funzionalità di ripetizione dei tentativi integrate o di terze parti sono sufficienti.
- Riprovare su più livelli dello stack di applicazioni in modo da accrescere in modo significativo i nuovi tentativi e pertanto da consumare ulteriormente le risorse in una tempesta di ripetizioni dei tentativi. Assicurati di comprendere in che modo questi errori influiscono sulla tua applicazione e sulle dipendenze su cui fai affidamento, quindi implementa i nuovi tentativi a un solo livello.
- Riesecuzione delle chiamate dei servizi non idempotenti, con effetti collaterali imprevisti come risultati duplicati.

Vantaggi dell'adozione di questa best practice: i nuovi tentativi aiutano i client a ottenere i risultati desiderati quando le richieste non riescono, ma consumano più tempo del server per ottenere le risposte corrette desiderate. Quando gli errori sono rari o transitori, i nuovi tentativi funzionano correttamente. Quando gli errori sono causati da un sovraccarico di risorse, i nuovi tentativi possono peggiorare le cose. L'aggiunta di un backoff esponenziale con jitter ai tentativi dei client consente ai server di recuperare risorse quando gli errori sono causati dal sovraccarico delle risorse. Il jitter evita l'allineamento delle richieste in picchi e il backoff riduce l'aumento del carico causato dall'aggiunta di nuovi tentativi al normale carico delle richieste. Infine, è importante configurare un numero massimo di tentativi o il tempo trascorso per evitare la creazione di backlog che producono errori metastabili.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

Controlla e limita le chiamate riproposte. Utilizza il backoff esponenziale per eseguire nuovi tentativi dopo intervalli progressivamente più lunghi. Introduci il jitter per randomizzare gli intervalli di ripetizione e limitare il numero massimo di tentativi.

Alcuni AWS SDK implementano i nuovi tentativi e il backoff esponenziale per impostazione predefinita. Usa queste implementazioni AWS integrate laddove applicabile nel tuo carico di lavoro. Implementa una logica simile nel tuo carico di lavoro quando chiami servizi idempotenti e i cui tentativi migliorano la disponibilità dei client. Potrai decidere quali sono i timeout e quando cessare i tentativi in base al tuo caso d'uso. Crea ed esegui scenari di test per quei casi d'uso relativi ai nuovi tentativi.

Passaggi dell'implementazione

- Determina il livello ottimale nello stack di applicazioni per implementare nuovi tentativi per i servizi su cui si basa l'applicazione.
- Presta attenzione agli SDK esistenti che implementano strategie collaudate di ripetizione dei tentativi con backoff esponenziale e jitter per la lingua prescelta, e preferisci queste soluzioni anziché scrivere implementazioni personalizzate.
- Verifica che [i servizi siano idempotenti](#) prima di implementare nuovi tentativi. Una volta implementati i nuovi tentativi, assicurati che siano testati e che vengano regolarmente eseguiti in produzione.
- Quando chiami le API del servizio AWS, utilizza gli [AWS SDK](#) e [AWS CLI](#) e analizza le opzioni di configurazione dei nuovi tentativi. Determina se le impostazioni predefinite sono adatte al tuo caso d'uso, esegui i test e regola i valori secondo necessità.

Risorse

Best practice correlate:

- [REL04-BP04 Rendere tutte le risposte idempotenti](#)
- [REL05-BP02 Richieste di limitazione \(della larghezza di banda della rete\)](#)
- [REL05-BP04 Anticipazione degli errori e limitazione delle code](#)
- [REL05-BP05 Impostazione dei timeout dei client](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)

Documenti correlati:

- [Ripetizione dei tentativi in caso di errore e backoff esponenziale in AWS](#)
- [The Amazon Builders' Library: Timeout, nuovi tentativi e backoff con jitter](#)
- [Exponential Backoff and Jitter \(Jitter e backoff esponenziale\)](#)
- [Rendere sicuri i tentativi con API idempotenti](#)

Esempi correlati:

- [Spring Retry](#)
- [Resilience4j Retry](#)

Video correlati:

- [Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library \(Presentazione della libreria dei costruttori di Amazon\) \(DOP328\)](#)

Strumenti correlati:

- [AWS SDKs and Tools: Retry behavior \(AWS SDK e strumenti: comportamento dei tentativi\)](#)
- [AWS Command Line Interface: tentativi AWS CLI](#)

REL05-BP04 Anticipazione degli errori e limitazione delle code

Se un servizio non è in grado di rispondere correttamente a una richiesta, anticipa l'errore (fail fast). Ciò consente il rilascio delle risorse associate a una richiesta e permette al servizio di recuperare le risorse se queste sono in esaurimento. L'anticipazione degli errori (fail fast) è un modello di progettazione software consolidato che può essere usato per creare carichi di lavoro altamente affidabili nel cloud. Anche l'accodamento è un modello di integrazione aziendale consolidato che può semplificare il carico e consentire ai client di rilasciare risorse quando l'elaborazione asincrona può essere tollerata. Quando un servizio è in grado di rispondere correttamente in condizioni normali ma fallisce quando la frequenza delle richieste è troppo alta, utilizza una coda per memorizzare le richieste nel buffer. Tuttavia, non consentire la creazione di backlog di code lunghe che possono comportare l'elaborazione di richieste obsolete già dismesse dal client.

Risultato desiderato: Quando i sistemi rilevano conflitti a livello di risorse, timeout, eccezioni o errori che rendono irraggiungibili gli obiettivi dei livelli di servizio, le strategie di anticipazione degli errori (fail fast) consentono un ripristino più rapido del sistema. I sistemi che devono assorbire i picchi di traffico e sono in grado di gestire l'elaborazione asincrona possono migliorare l'affidabilità consentendo ai client di rilasciare rapidamente le richieste utilizzando le code per archiviare le richieste nei servizi di back-end. Quando le richieste vengono memorizzate nei buffer delle code, vengono implementate strategie di gestione delle code per evitare backlog ingestibili.

Anti-pattern comuni:

- Implementazione delle code di messaggi ma non la configurazione delle code DLQ o degli allarmi nei volumi DLQ per rilevare quando un sistema è in errore.
- Mancata misurazione dell'età dei messaggi in una coda, misurazione della latenza per capire quando gli utenti della coda sono in ritardo o generano errori che causano un nuovo tentativo.
- Mancata cancellazione dei messaggi nel backlog da una coda quando non è più necessario elaborare questi messaggi se l'azienda non lo richiede più.
- La configurazione delle code First in First Out (FIFO) quando le code Last In First Out (LIFO) soddisferebbe meglio le esigenze dei client, ad esempio quando non sono richiesti ordini rigorosi e l'elaborazione dei backlog sta ritardando tutte le richieste nuove e urgenti, con conseguente violazione dei livelli di servizio per tutti i client.
- Esposizione delle code interne ai client, invece dell'esposizione delle API che gestiscono l'acquisizione del lavoro e l'inserimento delle richieste in code interne.

- Combinazione di un numero eccessivo di tipi di richieste di lavoro in un'unica coda; ciò può aggravare le condizioni dei backlog in seguito alla distribuzione delle richieste di risorse tra i tipi di richiesta.
- Elaborazione di richieste complesse e semplici nella stessa coda, nonostante siano necessari monitoraggio, timeout e allocazioni di risorse diversi.
- Mancata convalida degli input o utilizzo di asserzioni per implementare meccanismi di anticipazione degli errori (fail fast) nel software che generano eccezioni a componenti di livello superiore in grado di gestire normalmente gli errori.
- Mancata rimozione delle risorse in errore dall'instradamento delle richieste, soprattutto quando gli errori generano risultati sia positivi che negativi dovuti ad arresti anomali e riavvii, errori intermittenti a livello di dipendenze, capacità ridotta o perdita di pacchetti di rete.

Vantaggi dell'adozione di questa best practice: I sistemi con anticipazione degli errori sono più facili da sottoporre al debug e alla correzione degli errori e spesso presentano problemi di codifica e configurazione prima che le versioni vengano pubblicate in produzione. I sistemi che incorporano strategie di accodamento efficaci forniscono maggiore resilienza e affidabilità in caso di picchi di traffico e di condizioni intermittenti di errore del sistema.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

Le strategie di anticipazione degli errori possono essere codificate in soluzioni software e configurate nell'infrastruttura. Oltre all'anticipazione degli errori (fail fast), le code sono una tecnica semplice ma affidabile di definizione dell'architettura che consente il caricamento senza problemi di componenti disaccoppiati del sistema. [Amazon CloudWatch](#) fornisce funzionalità per il monitoraggio e la segnalazione di guasti. Una volta accertato il malfunzionamento di un sistema, è possibile ricorrere a strategie di mitigazione, ad esempio per evitare problemi dovuti a risorse danneggiate. Quando i sistemi implementano le code con [Amazon SQS](#) e altre tecnologie di accodamento, per semplificare il caricamento, devono valutare come gestire i backlog e gli errori di utilizzo dei messaggi.

Passaggi dell'implementazione

- Implementa asserzioni programmatiche o metriche specifiche nel tuo software e utilizzale per avvisare esplicitamente in caso di problemi a livello di sistema. Amazon CloudWatch ti aiuta a creare metriche e allarmi in base al modello di log delle applicazioni e alla strumentazione SDK.

- Usa le metriche CloudWatch e gli allarmi per eseguire il failover per le risorse danneggiate responsabili dell'incremento della latenza dell'elaborazione o che ripetutamente non riescono a elaborare le richieste.
- Utilizza l'elaborazione asincrona. A tale scopo, progetta API in grado di accettare le richieste e aggiungere richieste alle code interne mediante Amazon SQS e, quindi, rispondere al client che genera il messaggio con un messaggio di successo, in modo che il client possa rilasciare risorse e passare ad altre attività mentre gli utenti nella coda di back-end elaborano le richieste.
- Misura e monitora la latenza di elaborazione delle code generando una metrica CloudWatch ogni volta che escludi un messaggio da una coda confrontandolo con il timestamp del messaggio.
- Quando gli errori impediscono la corretta elaborazione dei messaggi o il traffico aumenta a livelli tali da impedirne l'elaborazione in base agli accordi sul livello di servizio, escludi il traffico obsoleto o in eccesso indirizzandolo a una coda per il traffico eccedente. Ciò consente l'elaborazione prioritaria del nuovo processo e del processo più vecchio quando si rende disponibile nuova capacità. Questa tecnica è un'approssimazione dell'elaborazione LIFO e consente la normale elaborazione del sistema per tutti i nuovi processi.
- Usa le code DLQ o le code di reindirizzamento per spostare i messaggi che non possono essere elaborati dal backlog in una posizione che può essere ricercata e risolta in un secondo momento.
- Riprova o, se possibile, elimina i vecchi messaggi confrontandoli con il timestamp del messaggio ed eliminando i messaggi che non sono più rilevanti per il client richiedente.

Risorse

Best practice correlate:

- [REL04-BP02 Implementazione di dipendenze "loosely coupled"](#)
- [REL05-BP02 Richieste di limitazione \(della larghezza di banda della rete\)](#)
- [REL05-BP03 Controllo e limitazione delle chiamate di ripetizione](#)
- [REL06-BP02 Definizione e calcolo dei parametri \(aggregazione\)](#)
- [REL06-BP07 Monitoraggio del tracciamento end-to-end delle richieste attraverso il sistema](#)

Documenti correlati:

- [Evitare backlog di coda insormontabili](#)
- [Anticipazione degli errori \(fail fast\)](#)

- [Come posso prevenire un aumento del backlog dei messaggi nella mia coda Amazon SQS?](#)
- [Elastic Load Balancing: spostamento zonale](#)
- [Sistema di controllo Amazon Route 53 per il ripristino di applicazioni: controllo dell'instradamento per il failover del traffico](#)

Esempi correlati:

- [Modelli di integrazione aziendale: canale DLQ](#)

Video correlati:

- [AWS re:Invent 2022 - Operating highly available Multi-AZ applications \(Esecuzione di applicazioni multi-AZ a disponibilità elevata\)](#)

Strumenti correlati:

- [Amazon SQS](#)
- [Amazon MQ](#)
- [AWS IoT Core](#)
- [Amazon CloudWatch](#)

REL05-BP05 Impostazione dei timeout dei client

Imposta i timeout in modo appropriato per connessioni e richieste, verificali sistematicamente e non fare affidamento sui valori predefiniti perché non fanno riferimento alle specifiche del carico di lavoro.

Risultato desiderato: I timeout dei client devono considerare il costo per client, server e carico di lavoro associato all'attesa di richieste il cui completamento richiede una quantità di tempo anomala. Poiché non è possibile conoscere la causa esatta di un timeout, i client devono fare riferimento ai servizi per sviluppare ipotesi sulle cause probabili e sui timeout appropriati.

Il timeout delle connessioni client si verifica in base ai valori configurati. Dopo aver rilevato un timeout, i client decidono di riprovare o aprire un [interruttore](#). Questi modelli evitano la generazione di richieste che potrebbero aggravare una condizione di errore sottostante.

Anti-pattern comuni:

- Non essere a conoscenza dei timeout di sistema o dei timeout predefiniti.
- Non essere a conoscenza dei normali tempi di completamento delle richieste.
- Non essere a conoscenza delle possibili cause dei tempi anomali necessari per il completamento delle richieste o dei costi in termini di prestazioni di client, servizio o carico di lavoro associati all'attesa di tali completamenti.
- Non essere consapevoli della probabilità che una rete danneggiata causi un errore di esecuzione della richiesta solo al raggiungimento del timeout, nonché dei costi in termini di prestazioni del client e del carico di lavoro derivanti dalla mancata adozione di un timeout più breve.
- Non testare gli scenari di timeout sia per le connessioni che per le richieste.
- Impostazione di timeout troppo elevati, che può comportare lunghi tempi di attesa e aumentare l'utilizzo delle risorse.
- Impostazione di timeout troppo bassi, con conseguenti errori artificiali.
- Mancata verifica degli schemi per gestire gli errori di timeout per chiamate remote come interruttori e nuovi tentativi.
- Non considerare il monitoraggio delle percentuali di errore delle chiamate dei servizi, degli obiettivi del livello di servizio per la latenza e dei valori anomali della latenza. Queste metriche possono fornire informazioni sui timeout restrittivi o permissivi.

Vantaggi dell'adozione di questa best practice: I timeout delle chiamate remote sono configurati e i sistemi sono progettati per gestirli correttamente, in modo da preservare le risorse quando le chiamate remote rispondono in modo eccessivamente lento e gli errori di timeout vengono gestiti correttamente dai client di servizio.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

Imposta sia un timeout di connessione che un timeout della richiesta su qualsiasi chiamata della dipendenza del servizio e, generalmente, su qualsiasi chiamata tra i processi. Molti framework offrono funzionalità di timeout integrate, ma è necessario prestare attenzione perché alcuni sono caratterizzati da valori predefiniti infiniti o superiori a quelli accettabili per gli obiettivi dei tuoi servizi. Un valore troppo elevato riduce l'utilità del timeout perché le risorse continuano a essere consumate mentre il client attende che si verifichi il timeout. Un valore troppo basso può generare un aumento del traffico sul back-end e una maggiore latenza perché vengono ritentate troppe richieste. In alcuni casi, questo può portare a interruzioni vere e proprie perché tutte le richieste vengono ritentate.

Considera quanto segue per determinare le strategie di timeout:

- L'elaborazione delle richieste può richiedere più tempo del normale a causa del loro contenuto, di problemi nel servizio di destinazione o di un errore nella partizione della rete.
- Le richieste con contenuti troppo costosi potrebbero consumare risorse server e client non necessarie. In questo caso, forzare il timeout di queste richieste e non eseguire nuovi tentativi possono preservare le risorse. I servizi dovrebbero, inoltre, proteggersi da contenuti eccessivamente costosi con limitazioni e timeout lato server.
- Per le richieste con tempi di elaborazione eccessivamente lunghi a causa di un'interruzione del servizio è possibile forzare il timeout e, quindi, eseguire un nuovo tentativo. È necessario considerare i costi del servizio per la richiesta e il nuovo tentativo, ma se la causa è un problema localizzato, è probabile che un nuovo tentativo non sia costoso e riduca il consumo di risorse del client. Il timeout può anche liberare risorse del server a seconda della natura del problema.
- Per le richieste il cui completamento richiede troppo tempo o per risposte non distribuite dalla rete è possibile forzare il timeout e, quindi, eseguire un nuovo tentativo. Poiché la richiesta o la risposta non è stata distribuita, viene comunque restituito un errore indipendentemente dalla durata del timeout. Il timeout in questo caso non rilascerà le risorse del server, ma le risorse del client, con il conseguente miglioramento delle prestazioni del carico di lavoro.

Sfrutta modelli di progettazione consolidati come i nuovi tentativi e interruttori per gestire normalmente i timeout e supportare l'approccio all'anticipazione degli errori (fail fast). [AWS SDK](#) e la [AWS CLI](#) consentono la configurazione dei timeout per connessioni e richieste dei nuovi tentativi con backoff esponenziale e jitter. [Le funzioni AWS Lambda](#) supportano la configurazione dei timeout e con [AWS Step Functions](#) puoi creare interruttori a uso limitato di codice che sfruttano le integrazioni predefinite con i servizi e gli SDK AWS. [AWS App Mesh](#) Envoy fornisce funzionalità di tipo timeout e interruttore.

Passaggi dell'implementazione

- Configura i timeout per le chiamate remote dei servizi e sfrutta le funzionalità di timeout integrate o le librerie di timeout open source.
- Quando il carico di lavoro esegue chiamate con un SDK AWS, consulta la documentazione per la configurazione del timeout specifica della lingua.
 - [Python](#)
 - [PHP](#)
 - [.NET](#)

- [Ruby](#)
- [Java](#)
- [Go](#)
- [Node.js](#)
- [C++](#)
- Quando usi SDK AWS o comandi AWS CLI nel carico di lavoro, configura i valori di timeout predefiniti impostando i valori di configurazione AWS [predefiniti](#) per `connectTimeoutInMillis` e `tlsNegotiationTimeoutInMillis`.
- Applica le [opzioni della riga di comando](#) `cli-connect-timeout` e `cli-read-timeout` per controllare i comandi AWS CLI occasionali nei servizi AWS.
- Monitora le chiamate remote dei servizi per i timeout e imposta gli allarmi sugli errori persistenti in modo da poter gestire in modo proattivo gli scenari di errore.
- Implementa [le metriche CloudWatch](#) e [il rilevamento delle anomalie CloudWatch](#) per le percentuali di errore nelle chiamate, gli obiettivi dei livelli di servizio per la latenza e i valori anomali della latenza per ottenere informazioni sulla gestione dei timeout eccessivamente restrittivi o permissivi.
- Configura i timeout per [le funzioni Lambda](#).
- I client API Gateway devono implementare nuovi tentativi specifici durante la gestione dei timeout. API Gateway supporta un [timeout di integrazione da 50 millisecondi a 29 secondi](#) per le integrazioni downstream e non effettua nuovi tentativi quando l'integrazione richiede il timeout.
- Implementa lo schema basato sull' [interruttore](#) per evitare di effettuare chiamate remote quando si è verificato il timeout. Apri l'interruttore per evitare chiamate non riuscite e chiudi l'interruttore quando le chiamate rispondono normalmente.
- Per i carichi di lavoro basati su container, verifica le funzioni [App Mesh Envoy](#) per usare i timeout e gli interruttori integrati.
- Utilizza AWS Step Functions per creare interruttori a uso limitato di codice per le chiamate remote dei servizi, in particolare quando vengono richiamati gli SDK AWS nativi e le integrazioni Step Functions supportate per semplificare il carico di lavoro.

Risorse

Best practice correlate:

- [REL05-BP03 Controllo e limitazione delle chiamate di ripetizione](#)
- [REL05-BP04 Anticipazione degli errori e limitazione delle code](#)

- [REL06-BP07 Monitoraggio del tracciamento end-to-end delle richieste attraverso il sistema](#)

Documenti correlati:

- [AWS SDK: Retries and Timeouts \(SDK AWS: nuovi tentativi e timeout\)](#)
- [The Amazon Builders' Library: Timeout, nuovi tentativi e backoff con jitter](#)
- [Quote Amazon API Gateway e note importanti](#)
- [AWS Command Line Interface: opzioni della riga di comando](#)
- [AWS SDK for Java 2.x: configurazione dei timeout delle API](#)
- [AWS Botocore mediante l'oggetto config e informazioni di riferimento sulla configurazione](#)
- [AWS SDK for .NET: nuovi tentativi e timeout](#)
- [AWS Lambda: configurazione delle opzioni della funzione Lambda](#)

Esempi correlati:

- [Utilizzo dello schema dell'interruttore con AWS Step Functions e Amazon DynamoDB](#)
- [Martin Fowler: CircuitBreaker](#)

Strumenti correlati:

- [AWS SDK](#)
- [AWS Lambda](#)
- [Amazon SQS](#)
- [AWS Step Functions](#)
- [AWS Command Line Interface](#)

REL05-BP06 Utilizzo dei sistemi stateless laddove possibile

I sistemi non devono richiedere lo stato né eseguire l'offload dello stato in modo tale che, tra diverse richieste client, non vi sia alcuna dipendenza dai dati archiviati localmente su disco o in memoria. I server possono così essere sostituiti a piacimento senza compromettere la disponibilità.

Quando gli utenti o i servizi interagiscono con un'applicazione, spesso eseguono una serie di interazioni che formano una sessione. Una sessione è un dato univoco per gli utenti che persistono

tra le richieste mentre utilizzano l'applicazione. Un'applicazione stateless è un'applicazione che non richiede la conoscenza delle interazioni precedenti e non memorizza le informazioni sulla sessione.

Una volta progettata per essere stateless, puoi utilizzare servizi di elaborazione serverless, come AWS Lambda o AWS Fargate.

Oltre alla sostituzione dei server, un altro vantaggio delle applicazioni stateless è la possibilità di scalare orizzontalmente, perché qualsiasi risorsa di calcolo disponibile (ad esempio istanze EC2 e funzioni AWS Lambda) può soddisfare ogni richiesta.

Vantaggi dell'adozione di questa best practice: i sistemi progettati per essere stateless sono più adattabili al dimensionamento orizzontale, rendendo possibile l'aggiunta o la rimozione di capacità in base alle fluttuazioni del traffico e della domanda. Sono inoltre intrinsecamente resistenti ai guasti e offrono flessibilità e agilità allo sviluppo delle applicazioni.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

Trasforma le applicazioni in stateless. Le applicazioni stateless consentono il dimensionamento orizzontale e sono tolleranti ai guasti di un singolo nodo. Analizza e individua i componenti della tua applicazione che mantengono lo stato dell'architettura. Questo processo ti aiuta a valutare il potenziale impatto della transizione a una progettazione stateless. Un'architettura stateless separa i dati degli utenti ed esegue l'offload dei dati della sessione, offrendo la flessibilità necessaria per scalare ogni componente in modo indipendente al fine di soddisfare le diverse richieste del carico di lavoro e ottimizzare l'utilizzo delle risorse.

Passaggi dell'implementazione

- Individua e comprendi i componenti stateful dell'applicazione.
- Suddividi i dati, separando e gestendo i dati dell'utente dalla logica dell'applicazione principale.
 - [Amazon Cognito](#) è in grado di separare i dati dell'utente dal codice dell'applicazione utilizzando funzionalità quali [pool di identità](#), [pool di utenti](#) e [Amazon Cognito Sync](#).
 - Puoi utilizzare [AWS Secrets Manager](#) per separare i dati dell'utente archiviando i segreti in un luogo sicuro e centralizzato. Il codice dell'applicazione pertanto non dovrà più memorizzare i segreti, rendendolo più sicuro.
 - Prendi in considerazione l'utilizzo di [Amazon S3](#) per archiviare dati non strutturati di grandi dimensioni, come immagini e documenti. L'applicazione può recuperare questi dati quando richiesto, eliminando la necessità di archivarli in memoria.

- Usa [Amazon DynamoDB](#) per memorizzare informazioni come i profili utente. L'applicazione può eseguire query su questi dati pressoché in tempo reale.
- Trasferisci i dati della sessione in un database, una cache o in file esterni.
- [Amazon ElastiCache](#), Amazon DynamoDB, [Amazon Elastic File System \(Amazon EFS\)](#) e [Amazon MemoryDB for Redis](#) sono esempi di servizi AWS che puoi utilizzare per eseguire l'offload dei dati della sessione.
- Progetta un'architettura stateless dopo aver identificato lo stato e i dati dell'utente che devono essere mantenuti con la tua soluzione di archiviazione preferita.

Risorse

Best practice correlate:

- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)

Documenti correlati:

- [Amazon Builders' Library: Evitare il fallback nei sistemi distribuiti](#)
- [Amazon Builders' Library: Evitare insormontabili backlog di code](#)
- [Amazon Builders' Library: Sfide e strategie del caching](#)
- [Best practice su AWS Livello Web senza stato](#)

REL05-BP07 Implementazione di leve di emergenza

Le leve di emergenza sono processi rapidi che possono mitigare l'impatto sulla disponibilità sul carico di lavoro.

Le leve di emergenza disabilitano, limitano o modificano il comportamento di componenti o dipendenze mediante meccanismi noti e testati. Ciò può ridurre i danni causati al carico di lavoro dall'esaurimento delle risorse dovuto ad aumenti imprevisti della domanda e l'impatto dei guasti nei componenti non critici all'interno del carico di lavoro.

Risultato desiderato: implementando le leve di emergenza, è possibile stabilire processi validi noti per garantire la disponibilità dei componenti critici nel carico di lavoro. Il carico di lavoro dovrebbe diminuire gradualmente e continuare a svolgere le sue funzioni aziendali critiche durante l'attivazione di una leva di emergenza. Per ulteriori informazioni sulla parziale riduzione delle prestazioni,

consulta [REL05-BP01 Implementazione della normale riduzione delle prestazioni per trasformare le dipendenze forti applicabili in dipendenze deboli](#).

Anti-pattern comuni:

- L'errore a livello di dipendenze non critiche influisce sulla disponibilità del carico di lavoro principale.
- Mancato test o mancata verifica del comportamento dei componenti critici durante il deterioramento delle prestazioni dei componenti non critici.
- Mancata definizione di criteri chiari e deterministici per l'attivazione o la disattivazione di una leva di emergenza.

Vantaggi dell'adozione di questa best practice: l'implementazione delle leve di emergenza può migliorare la disponibilità dei componenti critici del carico di lavoro fornendo ai risolutori processi consolidati per rispondere a picchi di domanda imprevisti o errori a livello di dipendenze non critiche.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

- Identifica i componenti critici del tuo carico di lavoro.
- Progetta e definisci l'architettura dei componenti critici del tuo carico di lavoro in modo che sia in grado di sostenere i guasti dei componenti non critici.
- Esegui i test per convalidare il comportamento dei componenti critici in caso di guasti dei componenti non critici.
- Definisci e monitora le metriche o i trigger pertinenti per avviare le procedure relative alle leve di emergenza.
- Definisci le procedure (manuali o automatiche) che includono la leva di emergenza.

Passaggi dell'implementazione

- Identifica i componenti business-critical nel tuo carico di lavoro.
 - Ogni componente tecnico del carico di lavoro deve essere mappato alla funzione aziendale pertinente e classificato come critico o non critico. Per esempi di funzionalità critiche e non critiche in Amazon, consulta [Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second](#) (informazioni in lingua inglese).

- Si tratta di una decisione sia tecnica che aziendale e varia in base all'organizzazione e al carico di lavoro.
- Progetta e definisci l'architettura dei componenti critici del tuo carico di lavoro in modo che sia in grado di sostenere i guasti dei componenti non critici.
- Durante l'analisi delle dipendenze, valuta tutte le potenziali modalità di guasto e verifica che i meccanismi basati su leve di emergenza forniscano le funzionalità critiche ai componenti a valle.
- Esegui i test per convalidare il comportamento dei componenti critici durante l'attivazione delle leve di emergenza.
- Evita il comportamento bimodale. Per maggiori dettagli, consulta [REL11-BP05 Utilizzo della stabilità statica per evitare un comportamento bimodale](#).
- Definisci, monitora e attiva gli avvisi per le metriche pertinenti per avviare la procedura relative alla leva di emergenza.
- L'individuazione delle metriche da monitorare dipende dal carico di lavoro. Alcuni esempi di metrica sono la latenza o il numero di richieste non riuscite nei confronti di una dipendenza.
- Definisci le procedure (manuali o automatiche) che includono la leva di emergenza.
- Ciò può includere meccanismi come la [riduzione del carico](#), le [richieste di limitazione della larghezza di banda della rete \(throttling\)](#) o l'implementazione di una [parziale riduzione delle prestazioni](#).

Risorse

Best practice correlate:

- [REL05-BP01 Implementazione della normale riduzione delle prestazioni per trasformare le dipendenze forti applicabili in dipendenze deboli](#)
- [REL05-BP02 Richieste di limitazione \(della larghezza di banda della rete\)](#)
- [REL11-BP05 Utilizzo della stabilità statica per evitare un comportamento bimodale](#)

Documenti correlati:

- [Automazione di implementazioni pratiche e sicure](#)
- [Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second](#) (informazioni in lingua inglese)

Video correlati:

- [AWS re:Invent 2020: Reliability, consistency, and confidence through immutability](#)

Gestione delle modifiche

Le modifiche apportate al carico di lavoro o al relativo ambiente devono essere previste e gestite affinché il carico di lavoro funzioni in modo affidabile. Le modifiche includono quelle imposte al carico di lavoro, ad esempio i picchi di domanda, nonché quelle dall'interno quali le distribuzioni delle caratteristiche e le patch di sicurezza.

Nelle sezioni seguenti vengono illustrate le best practice per la gestione delle modifiche.

Argomenti

- [Monitoraggio delle risorse del carico di lavoro](#)
- [Progettazione di un carico di lavoro in grado di adattarsi ai cambiamenti della domanda](#)
- [Implementazione della modifica](#)

Monitoraggio delle risorse del carico di lavoro

I log e i parametri sono strumenti molto efficaci per ottenere informazioni sullo stato del tuo carico di lavoro. È possibile configurare il carico di lavoro in modo da monitorare i log e i parametri e inviare notifiche quando vengono superate le soglie o si verificano eventi significativi. Il monitoraggio consente al carico di lavoro di riconoscere quando vengono superate le soglie di prestazioni basse o si verificano errori, in modo che possa essere ripristinato automaticamente in risposta.

Il monitoraggio è essenziale per accertarti di soddisfare i requisiti di disponibilità. Il monitoraggio deve rilevare efficacemente gli errori. La modalità di errore peggiore è l'errore "silenzioso", in cui la funzionalità non è più attiva, ma non c'è modo di rilevarla se non indirettamente. I tuoi clienti se ne accorgono prima di te. L'avviso in caso di problemi è uno dei principali motivi per cui esegui il monitoraggio. I tuoi avvisi devono essere disaccoppiati dal tuo sistema il più possibile. Se l'interruzione del servizio elimina la funzionalità di avviso, avrai un periodo di interruzione più lungo.

In AWS, dotiamo le nostre applicazioni di strumenti su più livelli. Registriamo latenza, tassi di errore e disponibilità per ogni richiesta, per tutte le dipendenze e per le operazioni chiave all'interno del processo. Registriamo anche parametri di operazioni di successo. Questo ci consente di vedere i problemi imminenti prima che si verifichino. Non consideriamo solo la latenza media. Ci concentriamo ancora di più sui valori anomali di latenza, ad esempio il 99,9° e il 99,99° percentile. Questo perché se una richiesta su 1.000 o 10.000 è lenta, l'esperienza è comunque scarsa. Inoltre, anche se la media può essere accettabile, se una richiesta su 100 provoca una latenza estrema, ciò diventerà un problema man mano che il traffico cresce.

Il monitoraggio su AWS si compone di quattro fasi distinte:

1. Generazione - monitora tutti i componenti per il carico di lavoro
2. Aggregazione - definisci e calcola i parametri
3. Elaborazione in tempo reale e allarmi - invia notifiche e automatizza le risposte
4. Archiviazione e analisi

Best practice

- [REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro \(generazione\)](#)
- [REL06-BP02 Definizione e calcolo dei parametri \(aggregazione\)](#)
- [REL06-BP03 Invio di notifiche \(elaborazione e avvisi in tempo reale\)](#)
- [REL06-BP04 Automatizzazione delle risposte \(elaborazione e avvisi in tempo reale\)](#)
- [REL06-BP05 Analisi](#)
- [REL06-BP06 Esecuzione di revisioni periodiche](#)
- [REL06-BP07 Monitoraggio del tracciamento end-to-end delle richieste attraverso il sistema](#)

REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro (generazione)

monitora i componenti del carico di lavoro con Amazon CloudWatch o con strumenti di terze parti. Monitora i servizi AWS con il pannello di controllo AWS Health.

Occorre monitorare tutti i componenti del carico di lavoro, inclusi front-end, logica aziendale e livelli di storage. Definisci i parametri chiave e come estrarli dai registri, se necessario, e imposta soglie per l'attivazione degli eventi di allarme corrispondenti. Assicurati che i parametri siano pertinenti agli indicatori chiave di prestazione (KPI) del tuo carico di lavoro e utilizza i parametri e i registri per identificare i primi segnali di degrado del servizio. Ad esempio, un parametro legato ai risultati aziendali, come il numero di ordini elaborati con successo al minuto, può indicare problemi di carico di lavoro più rapidamente di un parametro tecnico, come l'utilizzo della CPU. Utilizza il pannello di controllo AWS Health per una visualizzazione personalizzata delle prestazioni e della disponibilità dei servizi AWS sottostanti alle risorse AWS.

Il monitoraggio nel cloud offre nuove opportunità. La maggior parte dei provider cloud ha sviluppato hook personalizzabili e può fornire approfondimenti per aiutarti a monitorare più livelli del carico di

lavoro. I servizi AWS come Amazon CloudWatch applicano algoritmi statistici e di apprendimento automatico per analizzare continuamente i parametri di sistemi e applicazioni, determinare le normali linee di base e far emergere le anomalie con un intervento minimo da parte dell'utente. Gli algoritmi di rilevamento delle anomalie tengono conto della stagionalità e delle variazioni di tendenza dei parametri.

AWS mette a disposizione una grande quantità di informazioni di monitoraggio e di registro che possono essere utilizzate per definire parametri specifici per i carichi di lavoro, processi di variazione della domanda e per l'adozione di tecniche di apprendimento automatico indipendentemente dalle competenze di ML.

Inoltre, monitora tutti gli endpoint esterni per avere la certezza che siano indipendenti dall'implementazione di base. Questo monitoraggio attivo può essere effettuato con transazioni sintetiche (talvolta indicate come canary utente, ma da non confondere con le implementazioni canary) che eseguono periodicamente una serie di attività comuni che corrispondono alle azioni eseguite dai client del carico di lavoro. Mantieni queste attività di breve durata e assicurati di non sovraccaricare il carico di lavoro durante il test. Amazon CloudWatch Synthetics ti consente di [creare canary sintetici](#) per monitorare gli endpoint e le API. Puoi anche combinare i nodi client sintetici Canary con la console AWS X-Ray per individuare quali Canary sintetiche stanno riscontrando problemi con errori, guasti o velocità di throttling per l'intervallo di tempo selezionato.

Risultato desiderato:

raccogliere e utilizzare i parametri critici di tutti i componenti del carico di lavoro per garantire l'affidabilità del carico di lavoro e un'esperienza utente ottimale. Rilevare che un carico di lavoro non sta raggiungendo i risultati aziendali consente di dichiarare rapidamente un disastro e di riprendersi da un incidente.

Anti-pattern comuni:

- Solo monitoraggio delle interfacce esterne per il carico di lavoro.
- Non generare parametri specifici per il carico di lavoro e affidati solo ai parametri forniti dai servizi AWS utilizzati dal carico di lavoro.
- Utilizzare solo parametri tecnici nel carico di lavoro e non monitorare i parametri relativi agli indicatori chiave di prestazione (KPI) non tecnici a cui il carico di lavoro contribuisce.
- Affidarsi al traffico di produzione e a semplici controlli di integrità per monitorare e valutare lo stato del carico di lavoro.

Vantaggi dell'adozione di questa best practice: il monitoraggio a tutti i livelli del carico di lavoro consente di prevedere e risolvere più rapidamente i problemi dei componenti che costituiscono il carico di lavoro.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

1. Abilitazione della registrazione ove disponibile. I dati di monitoraggio devono essere ottenuti da tutti i componenti dei carichi di lavoro. Attiva ulteriori registri, come i registri di accesso S3, e abilita il carico di lavoro per registrare i dati specifici del carico di lavoro. Raccogli i parametri per le medie di CPU, I/O di rete e I/O su disco da servizi come Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling ed Amazon EMR. Consulta [Servizi AWS che pubblicano parametri CloudWatch](#) Servizi AWS che pubblicano parametri su CloudWatch.
2. Esamina tutti i parametri predefiniti ed esplora eventuali lacune nella raccolta dei dati. Tutti i servizi generano parametri predefiniti. La raccolta di parametri predefiniti consente di comprendere meglio le dipendenze tra i componenti del carico di lavoro e il modo in cui l'affidabilità e le prestazioni dei componenti influiscono sul carico di lavoro. Puoi anche creare e [pubblicare parametri propri](#) affinché CloudWatch utilizzi la AWS CLI o un'API. Questo
3. valuta tutti i parametri per decidere quelli a cui inviare avvisi per ogni servizio AWS nel carico di lavoro. Puoi scegliere di selezionare un sottoinsieme di parametri che hanno un impatto importante sull'affidabilità del carico di lavoro. La focalizzazione su soglie e parametri critici consente di affinare il numero di avvisi [informativi](#) e può contribuire a ridurre al minimo i falsi positivi.
4. Definisci gli avvisi e il processo di recupero del carico di lavoro dopo l'attivazione dell'avviso. La definizione degli avvisi consente di notificare, intensificare e seguire rapidamente le fasi necessarie per il ripristino da un incidente e il rispetto dell'obiettivo di tempo di ripristino (RTO) prescritto. Puoi utilizzare [avvisi Amazon CloudWatch](#) per invocare flussi di lavoro automatici e avviare procedure di ripristino in base a soglie definite.
5. Esplora l'uso di transazioni sintetiche per raccogliere dati rilevanti sullo stato dei carichi di lavoro. Il monitoraggio sintetico segue gli stessi percorsi ed esegue le stesse azioni di un cliente, il che consente di verificare continuamente l'esperienza del cliente anche quando non c'è traffico di clienti sui carichi di lavoro. Utilizzando [le transazioni sintetiche](#), puoi individuare i problemi prima dei clienti.

Risorse

Best practice correlate:

- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)

Documenti correlati:

- [Getting started with your AWS Health Dashboard – Your account health \(Nozioni di base su AWS HealthDashboard: stato del tuo account\)](#)
- [Servizi AWS che pubblicano parametri CloudWatch](#)
- [Log di accesso per Network Load Balancer](#)
- [Log di accesso per Application Load Balancer](#)
- [Accesso a Amazon CloudWatch Logs per AWS Lambda](#)
- [Registrazione delle richieste con registrazione dell'accesso al server Amazon S3](#)
- [Abilita i log di accesso per Classic Load Balancer](#)
- [Esportazione di dati di registro in Amazon S3](#)
- [Installazione dell'agente CloudWatch su un'istanza Amazon EC2](#)
- [Pubblicazione di parametri personalizzati](#)
- [Utilizzo dei pannelli di controllo Amazon CloudWatch](#)
- [Utilizzare i parametri Amazon CloudWatch](#)
- [Utilizzo di Canary \(Amazon CloudWatch Synthetics\)](#)
- [Cosa sono i Amazon CloudWatch Logs?](#)

Guide per l'utente:

- [Creazione di un trail](#)
- [Monitoraggio dei parametri di memoria e del disco per le istanze Amazon EC2 Linux](#)
- [Utilizzo di CloudWatch Logs con istanze di container](#)
- [Log di flusso VPC](#)
- [Che cos'è Amazon DevOps Guru?](#)
- [Che cos'è AWS X-Ray?](#)

Blog correlati:

- [Effettuare il debug con Amazon CloudWatch Synthetics e AWS X-Ray](#)

Esempi e workshop correlati:

- [AWS Well-Architected Labs: Operational Excellence - Dependency Monitoring \(Laboratori ben strutturati AWS: Eccellenza operativa - Monitoraggio delle dipendenze\)](#)
- [The Amazon Builders' Library: Dotazione dei sistemi distribuiti per la visibilità operativa](#)
- [Workshop sull'osservabilità](#)

REL06-BP02 Definizione e calcolo dei parametri (aggregazione)

Archivia i dati di registro e applica i filtri, laddove necessari, per calcolare i parametri, ad esempio i conteggi di un evento di registro specifico o la latenza calcolata dai timestamp del registro eventi.

Amazon CloudWatch e Amazon S3 fungono da principali livelli di aggregazione e storage. Per alcuni servizi, come AWS Auto Scaling e Elastic Load Balancing, i parametri predefiniti vengono forniti per impostazione predefinita per il carico della CPU o la latenza media delle richieste in un cluster o in un'istanza. Per i servizi di streaming, come i registri di flusso VPC e AWS CloudTrail, i dati degli eventi vengono inoltrati a CloudWatch Logs ed è necessario definire e applicare filtri di parametri per estrarre i parametri dai dati dell'evento. In questo modo vengono forniti dati di serie temporali, che possono fungere da input per gli allarmi CloudWatch definiti dall'utente per attivare gli avvisi.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

- **Aggregazione:** definisci e calcola i parametri. Archivia i dati di log e applica filtri, se necessario, per calcolare i parametri, ad esempio i conteggi di un evento di log specifico o la latenza calcolata dai timestamp degli eventi di log
 - I filtri dei parametri definiscono i termini e i modelli da ricercare nei dati di registro inviati a CloudWatch Logs. CloudWatch Logs utilizza questi filtri di parametri per trasformare i dati di registro in parametri CloudWatch numerici che è possibile rappresentare su un grafico o un avviso.
 - [Ricerca e filtraggio dei dati di log](#)
 - Utilizza una terza parte affidabile per aggregare i registri.
 - Segui le istruzioni che ti vengono fornite dalle terze parti. La maggior parte dei prodotti di terze parti si integra con CloudWatch e Amazon S3.
 - Alcuni servizi AWS possono pubblicare registri direttamente in Amazon S3. Se il requisito principale per i registri è l'archiviazione in Amazon S3, si può facilmente fare in modo che

il servizio che produce i registri li invii direttamente a Amazon S3, senza dover creare un'infrastruttura aggiuntiva.

- [Invio di registri direttamente a Amazon S3](#)

Risorse

Documenti correlati:

- [Query di esempio di Amazon CloudWatch Logs Insights](#)
- [Effettuare il debug con Amazon CloudWatch Synthetics e AWS X-Ray](#)
- [One Observability Workshop](#)
- [Ricerca e filtraggio dei dati di log](#)
- [Invio di registri direttamente a Amazon S3](#)
- [The Amazon Builders' Library: Dotazione dei sistemi distribuiti per la visibilità operativa](#)

REL06-BP03 Invio di notifiche (elaborazione e avvisi in tempo reale)

Quando le organizzazioni rilevano potenziali problemi, inviano notifiche e avvisi in tempo reale ai team e ai sistemi appropriati per rispondere rapidamente ed efficacemente alle difficoltà.

Risultato desiderato: è possibile rispondere rapidamente agli eventi operativi attraverso la configurazione di allarmi pertinenti in base ai parametri del servizio e dell'applicazione. Quando la soglia degli allarmi viene superata, i team e i sistemi appropriati vengono informati in modo che possano risolvere i problemi sottostanti.

Anti-pattern comuni:

- Configuri gli allarmi con una soglia eccessivamente alta, con conseguente mancato invio di notifiche importanti.
- Configuri gli allarmi con una soglia troppo bassa, con il risultato che gli avvisi importanti non vengono presi in considerazione a causa del numero eccessivo di notifiche generate.
- Non aggiorni gli allarmi e la relativa soglia quando cambia l'utilizzo.
- Per gli allarmi gestiti meglio tramite le azioni automatizzate, l'invio della notifica ai team anziché l'attivazione dell'azione automatizzata comporta la generazione di un numero eccessivo di notifiche.

Vantaggi dell'adozione di questa best practice: l'invio di notifiche e avvisi in tempo reale ai team e ai sistemi appropriati consente di individuare tempestivamente i problemi e di rispondere rapidamente agli incidenti operativi.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

I carichi di lavoro devono essere dotati di sistemi di elaborazione e allarme in tempo reale per migliorare l'identificazione dei problemi che possono influire sulla disponibilità dell'applicazione e fungere da trigger per la risposta automatizzata. Le organizzazioni possono eseguire un sistema di elaborazione e allarme in tempo reale creando avvisi con parametri definiti in modo da ricevere le notifiche ogni volta che si verificano eventi significativi o un parametro supera una determinata soglia.

[Amazon CloudWatch](#) ti permette di creare [allarmi](#) composti e di parametri utilizzando gli allarmi CloudWatch basati su soglie statiche, rilevamento di anomalie e altri criteri. Per maggiori dettagli sui tipi di allarmi che puoi configurare utilizzando CloudWatch, consulta la [sezione allarmi della documentazione CloudWatch](#).

Puoi creare per i tuoi team visualizzazioni personalizzate dei parametri e degli avvisi delle risorse AWS utilizzando le [dashboard CloudWatch](#). Le home page personalizzabili nella console di CloudWatch consentono di monitorare le risorse di più regioni in un'unica visualizzazione.

Gli allarmi possono eseguire una o più azioni, come inviare una notifica a un [argomento Amazon SNS](#), eseguendo un'azione su [Amazon EC2](#) o un'azione su [Amazon EC2 Auto Scaling](#) oppure [creando un OpsItem](#) o [a](#) in AWS Systems Manager.

Amazon CloudWatch utilizza [Amazon SNS](#) per inviare le notifiche quando l'allarme cambia stato, con la distribuzione dei messaggi degli editori (produttori) agli abbonati (consumatori). Per maggiori dettagli sull'impostazione delle notifiche Amazon SNS, consulta [Configurazione di Amazon SNS](#).

CloudWatch invia [EventBridge della sicurezza](#) ogni volta che un allarme CloudWatch viene creato, aggiornato, eliminato o cambia stato. Puoi usare EventBridge con questi eventi per creare le regole che eseguono le azioni, come avvisare ogni volta che lo stato di un allarme cambia o attivare automaticamente gli eventi nel tuo account tramite [l'automazione Systems Manager](#).

Quando si usa EventBridge rispetto ad Amazon SNS?

EventBridge e Amazon SNS possono entrambi essere utilizzati per sviluppare applicazioni basate su eventi e la scelta dipende dalle tue esigenze specifiche.

Amazon EventBridge è consigliato quando desideri creare un'applicazione che reagisca agli eventi delle tue applicazioni, delle applicazioni SaaS e dei servizi AWS. EventBridge è l'unico servizio basato su eventi che si integra direttamente con i partner SaaS di terze parti. EventBridge inoltre acquisisce automaticamente eventi da oltre 200 servizi AWS senza richiedere agli sviluppatori di creare risorse negli account.

EventBridge utilizza una struttura definita basata su JSON per gli eventi e consente di creare regole applicate all'intero corpo dell'evento per selezionare gli eventi da inoltrare alle [destinazioni](#). EventBridge attualmente supporta oltre 20 servizi AWS come destinazioni, tra cui [AWS Lambda](#), [Amazon SQS](#), Amazon SNS, [Amazon Kinesis Data Streamse](#) [Amazon Data Firehose](#).

Amazon SNS è consigliato per le applicazioni che richiedono un fan-out elevato (migliaia o milioni di endpoint). Di solito i clienti utilizzano Amazon SNS come destinazione della regola per filtrare gli eventi di cui hanno bisogno e sottoporli al fan-out su più endpoint.

I messaggi non sono strutturati e possono essere in qualsiasi formato. Amazon SNS supporta l'inoltro dei messaggi a sei diversi tipi di destinazioni, tra cui Lambda, Amazon SQS, endpoint HTTP/S, SMS, push mobile ed e-mail. La latenza tipica di Amazon SNS [è inferiore a 30 millisecondi](#). Un'ampia gamma di servizi AWS invia i messaggi Amazon SNS definendo la configurazione appropriata (più di 30, inclusi Amazon EC2, [Amazon S3](#)e [Amazon RDS](#)).

Passaggi dell'implementazione

1. Crea un allarme usando gli [avvisi Amazon CloudWatch](#).
 - a. Un allarme di parametri monitora un singolo parametro CloudWatch o un'espressione dipendente dai parametri CloudWatch. L'allarme avvia una o più azioni in base al valore del parametro o dell'espressione rispetto a una soglia, per un determinato numero di intervalli di tempo. L'azione può consistere nell'inviare una notifica a un [argomento Amazon SNS](#), eseguendo un'azione su [Amazon EC2](#) o un'azione su [Amazon EC2 Auto Scaling](#) oppure [creando un OpsItem](#) o [a](#) in AWS Systems Manager.
 - b. Un allarme composito è costituito da un'espressione di regola che considera le condizioni di altri allarmi che hai creato. L'allarme composito entra in stato di allarme solo se tutte le condizioni della regola sono soddisfatte. Gli allarmi specificati nell'espressione di regola di un allarme composito possono includere allarmi di parametri e allarmi compositi aggiuntivi. Gli allarmi compositi possono inviare notifiche Amazon SNS quando il loro stato cambia e possono creare Systems Manager [OpsItems](#) o [incidenti](#) quando entrano nello stato di allarme, ma non possono eseguire azioni Amazon EC2 o Auto Scaling.

2. Configura [le notifiche Amazon SNS](#). Quando si crea un allarme CloudWatch, è possibile includere un argomento Amazon SNS per inviare una notifica quando l'allarme cambia stato.
3. [Crea regole in EventBridge](#) che corrisponde agli allarmi CloudWatch specificati. Ogni regola supporta più destinazioni, incluse le funzioni Lambda. Ad esempio, è possibile definire un allarme che si attiva quando lo spazio disponibile su disco si sta esaurendo e che esegue una funzione Lambda tramite una regola EventBridge per ripulire lo spazio. Per maggiori dettagli sulle destinazioni EventBridge, consulta [Destinazioni EventBridge](#).

Risorse

Best practice Well-Architected correlate:

- [REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro \(generazione\)](#)
- [REL06-BP02 Definizione e calcolo dei parametri \(aggregazione\)](#)
- [REL12-BP01 Utilizzo dei playbook per analizzare gli errori](#)

Documenti correlati:

- [Amazon CloudWatch](#)
- [CloudWatch Logs insights](#)
- [Utilizzo degli allarmi di Amazon CloudWatch](#)
- [Utilizzo dei pannelli di controllo Amazon CloudWatch](#)
- [Utilizzare i parametri Amazon CloudWatch](#)
- [Setting up Amazon SNS notifications](#)
- [il rilevamento delle anomalie CloudWatch](#)
- [Protezione dei dati CloudWatch Logs](#)
- [Amazon EventBridge](#)
- [Amazon Simple Notification Service](#)

Video correlati:

- [Video sull'osservabilità di reinvent 2022](#)
- [AWS re:Invent 2022 - Observability best practices at Amazon](#)

Esempi correlati:

- [One Observability Workshop](#)
- [Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms](#)

REL06-BP04 Automatizzazione delle risposte (elaborazione e avvisi in tempo reale)

utilizza l'automazione per agire quando viene rilevato un evento; ad esempio, per sostituire i componenti guasti.

L'elaborazione automatizzata in tempo reale degli allarmi è implementata in modo che i sistemi possano effettuare azioni correttive rapide e tentare di prevenire guasti o danni al servizio quando vengono attivati gli allarmi. Le risposte automatiche agli allarmi potrebbero includere la sostituzione dei componenti guasti, la regolazione della capacità di calcolo, il reindirizzamento del traffico verso host integri, zone di disponibilità o altre regioni e la notifica agli operatori.

Risultato desiderato: vengono identificati gli allarmi in tempo reale e viene impostata l'elaborazione automatizzata degli allarmi per richiamare le azioni appropriate per mantenere gli obiettivi dei livelli di servizio e gli accordi sul livello di servizio (SLA). L'automazione può interessare un ambito che va dalle attività di autoriparazione dei singoli componenti al failover dell'intero sito.

Anti-pattern comuni:

- Non disporre di un inventario o un catalogo dettagliato dei principali allarmi in tempo reale.
- Nessuna risposta automatica in caso di allarmi critici (ad esempio, quando le risorse di calcolo stanno per esaurirsi, viene implementato il dimensionamento automatico).
- Azioni di risposta agli allarmi contraddittorie.
- Nessuna procedura operativa standard (SOP) da seguire per gli operatori quando ricevono notifiche di avviso.
- Non monitorare le modifiche apportate alla configurazione, poiché le modifiche della configurazione non rilevate possono causare tempi di inattività per i carichi di lavoro.
- Non avere una strategia per annullare le modifiche involontarie alla configurazione.

Vantaggi dell'adozione di questa best practice: l'automazione dell'elaborazione degli allarmi può migliorare la resilienza del sistema. Il sistema implementa automaticamente azioni correttive,

riducendo le attività manuali che possono comportare interventi umani soggetti a errori. L'operatività del carico di lavoro soddisfa gli obiettivi di disponibilità e riduce le interruzioni del servizio.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

Per gestire in modo efficiente gli avvisi e automatizzarne la risposta, classifica gli avvisi in base alla loro criticità e al loro impatto, documenta le procedure di risposta e pianifica le risposte prima di classificare le attività.

Identifica le attività che richiedono azioni specifiche (spesso dettagliate nei runbook) ed esamina tutti i runbook e i playbook per determinare quali attività possono essere automatizzate. Se è possibile definire delle azioni, significa che esse spesso possono essere automatizzate. Se le azioni non possono essere automatizzate, documenta le fasi manuali in una procedura operativa standard (SOP) e forma gli operatori su tali procedure. Continua ad analizzare dettagliatamente i processi manuali alla ricerca di opportunità di automazione in cui puoi stabilire e mantenere un piano per automatizzare le risposte agli avvisi.

Passaggi dell'implementazione

1. Crea un inventario degli allarmi: per ottenere un elenco di tutti gli allarmi, nella [AWS CLI](#) puoi utilizzare il comando [Amazon CloudWatch describe-alarms](#). A seconda del numero di allarmi configurati, potrebbe essere necessario utilizzare la paginazione per recuperare un sottoinsieme di allarmi per ogni chiamata o, in alternativa, è possibile utilizzare AWS SDK per recuperare gli allarmi mediante una [chiamata API](#).
2. Documenta tutte le azioni degli allarmi: aggiorna un runbook con tutti gli allarmi e le relative azioni, indipendentemente dal fatto che siano manuali o automatiche. [AWS Systems Manager](#) fornisce runbook predefiniti. Per ulteriori informazioni sui runbook, consulta [Working with runbooks](#). Per informazioni dettagliate su come visualizzare il contenuto del runbook, consulta [View runbook content](#).
3. Configura e gestisci le azioni associate agli allarmi: per tutti gli allarmi che richiedono un'azione, specifica l'[azione automatizzata mediante CloudWatch SDK](#). Ad esempio, puoi modificare automaticamente lo stato delle tue istanze Amazon EC2 in base a un allarme CloudWatch creando e abilitando o disabilitando le azioni associate a un allarme.

Puoi anche utilizzare [Amazon EventBridge](#) per rispondere automaticamente agli eventi di sistema, come problemi di disponibilità delle applicazioni o modifiche delle risorse. Puoi creare regole per indicare quali eventi ti interessano e le azioni da eseguire quando un evento soddisfa una regola.

Le azioni che possono essere avviate automaticamente includono il richiamo di una funzione [AWS Lambda](#), il richiamo della funzionalità [Amazon EC2 Run Command](#), l'inoltro dell'evento a [Amazon Kinesis Data Streams](#) e la visualizzazione del comando [Automate Amazon EC2 mediante EventBridge](#).

4. Procedure operative standard (SOP): in base ai componenti dell'applicazione, [AWS Resilience Hub](#) consiglia più [modelli SOP](#). È possibile utilizzare queste SOP per documentare tutti i processi che un operatore deve seguire nel caso in cui venga generato un avviso. Puoi anche [creare una SOP](#) basata su raccomandazioni Resilience Hub, laddove sia necessaria un'applicazione Resilience Hub con una policy di resilienza associata, nonché una valutazione cronologica della resilienza rispetto a tale applicazione. Le raccomandazioni per la SOP sono prodotte dalla valutazione della resilienza.

Resilience Hub in combinazione con Systems Manager consente di automatizzare le fasi delle SOP fornendo una serie di [documenti SSM](#) che è possibile utilizzare come base per tali SOP. Ad esempio, Resilience Hub può consigliare una SOP per aggiungere spazio su disco in base a un documento SSM di automazione esistente.

5. Esegui azioni automatizzate utilizzando Amazon DevOps Guru: puoi utilizzare [Amazon DevOps Guru](#) per monitorare automaticamente le risorse dell'applicazione per rilevare comportamenti anomali e fornire raccomandazioni mirate per accelerare i tempi di identificazione e riparazione dei problemi. Con DevOps Guru, puoi monitorare flussi di dati operativi quasi in tempo reale da più origini, tra cui metriche Amazon CloudWatch, [AWS Config](#), [AWS CloudFormation](#) e [AWS X-Ray](#). È inoltre possibile utilizzare DevOps Guru per creare automaticamente [OpsItems](#) in OpsCenter e inviare eventi a [EventBridge per un'automazione aggiuntiva](#).

Risorse

Best practice correlate:

- [REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro \(generazione\)](#)
- [REL06-BP02 Definizione e calcolo dei parametri \(aggregazione\)](#)
- [REL06-BP03 Invio di notifiche \(elaborazione e avvisi in tempo reale\)](#)
- [REL08-BP01 Utilizzo di runbook per attività standard come l'implementazione](#)

Documenti correlati:

- [AWS Systems Manager Automation](#)

- [Creating an EventBridge Rule That Triggers on an Event from an AWS Resource](#)
- [One Observability Workshop](#)
- [The Amazon Builders' Library: Dotazione dei sistemi distribuiti per la visibilità operativa](#)
- [What is Amazon DevOps Guru?](#)
- [Gestione dei documenti di automazione \(playbook\)](#)

Video correlati:

- [AWS re:Invent 2022 - Best practice di visibilità in Amazon](#)
- [AWS re:Invent 2020: Automate anything with AWS Systems Manager](#)
- [Introduction to AWS Resilience Hub](#)
- [Create Custom Ticket Systems for Amazon DevOps Guru Notifications](#)
- [Enable Multi-Account Insight Aggregation with Amazon DevOps Guru](#)

Esempi correlati:

- [Workshop sull'affidabilità](#)
- [Workshop su Amazon CloudWatch e Systems Manager](#)

REL06-BP05 Analisi

raccogli i file di log e le cronologie dei parametri e analizzali per ottenere informazioni più ampie sulle tendenze e sui carichi di lavoro.

Amazon CloudWatch Logs Insights supporta un [linguaggio di query semplice ma potente](#) che puoi utilizzare per analizzare i dati di log. Amazon CloudWatch Logs supporta anche le sottoscrizioni che consentono ai dati di fluire in modo ottimale verso Amazon S3, dove puoi utilizzare o Amazon Athena per eseguire query sui dati. Supporta, inoltre, le query su un'ampia gamma di formati. Consulta [SerDe e formati di dati supportati](#) nella Guida per l'utente Amazon Athena per ulteriori informazioni. Per l'analisi di enormi set di file di log, puoi eseguire un cluster Amazon EMR per effettuare analisi con capacità nell'ordine dei petabyte.

Esistono numerosi strumenti forniti da Partner AWS e terze parti che consentono aggregazione, elaborazione, archiviazione e analisi. Questi strumenti includono New Relic, Splunk, Loggly,

Logstash, CloudHealth e Nagios. Tuttavia, la generazione esterna di log di sistema e applicazioni è univoca per ciascun provider di servizi cloud e spesso per ciascun servizio.

Una parte spesso trascurata del processo di monitoraggio è la gestione dei dati. È necessario determinare i requisiti di conservazione per il monitoraggio dei dati, quindi applicare le policy del ciclo di vita di conseguenza. Amazon S3 supporta la gestione del ciclo di vita a livello di bucket S3. Questa gestione del ciclo di vita può essere applicata in modo diverso ai diversi percorsi nel bucket. Verso la fine del ciclo di vita è possibile trasferire i dati su Amazon S3 Glacier per l'archiviazione a lungo termine fino alla scadenza, al termine del periodo di conservazione. La classe di storage S3 Intelligent-Tiering è progettata per ottimizzare i costi trasferendo automaticamente i dati nel livello di accesso più conveniente, senza impatto sulle prestazioni o sovraccarico operativo.

Livello di rischio associato se questa best practice non fosse adottata: Medium

Guida all'implementazione

- Gli approfondimenti CloudWatch Logs consentono di cercare e analizzare in modo interattivo i dati di registro in Amazon CloudWatch Logs.
 - [Analisi dei dati di registro con gli approfondimenti CloudWatch Logs](#)
 - [Query di esempio di Amazon CloudWatch Logs Insights](#)
- Utilizza Amazon CloudWatch Logs per inviare registri a Amazon S3 dove puoi utilizzare Amazon Athena per le query dei dati.
 - [Come faccio ad analizzare i miei registri di accesso al server Amazon S3 utilizzando Athena?](#)
 - Crea una policy del ciclo di vita di S3 per il bucket dei log di accesso al server. Configura la policy del ciclo di vita per rimuovere periodicamente i file di log. In questo modo si riduce la quantità di dati che Athena deve analizzare per ogni query.
 - [Come faccio a creare una policy del ciclo di vita per un bucket S3?](#)

Risorse

Documenti correlati:

- [Query di esempio di Amazon CloudWatch Logs Insights](#)
- [Analisi dei dati di registro con gli approfondimenti CloudWatch Logs](#)
- [Effettuare il debug con Amazon CloudWatch Synthetics e AWS X-Ray](#)
- [Come faccio a creare una policy del ciclo di vita per un bucket S3?](#)

- [Come faccio ad analizzare i miei registri di accesso al server Amazon S3 utilizzando Athena?](#)
- [One Observability Workshop](#)
- [The Amazon Builders' Library: Dotazione dei sistemi distribuiti per la visibilità operativa](#)

REL06-BP06 Esecuzione di revisioni periodiche

Esegui verifiche frequenti delle modalità di implementazione del monitoraggio del carico di lavoro e aggiornalo in base a eventi e modifiche significativi.

Il monitoraggio efficace è basato su parametri aziendali chiave. Assicurati che questi parametri siano presenti nel carico di lavoro man mano che le priorità aziendali cambiano.

L'audit del monitoraggio consente di sapere quando un'applicazione sta raggiungendo gli obiettivi di disponibilità. L'analisi delle cause principali richiede la capacità di scoprire cosa è successo in caso di errori. AWS consente di monitorare lo stato dei tuoi servizi durante un incidente:

- Amazon CloudWatch Logs: è possibile archiviare i log in questo servizio e controllarne i contenuti.
- Amazon CloudWatch Logs Insights: è un servizio completamente gestito che consente di eseguire analisi di registri di grandi dimensioni in pochi secondi. Offre query e visualizzazioni rapide e interattive.
- AWS Config: è possibile vedere quale infrastruttura AWS era in uso in momenti differenti.
- AWS CloudTrail: è possibile vedere quali API AWS sono state richiamate, a che ora e da quale principale.

In AWS, conduciamo meeting settimanali per [esaminare le prestazioni operative](#) e condividere quanto appreso tra i team. Dato l'elevato numero di team presenti in AWS, abbiamo creato [La ruota](#) per scegliere casualmente un carico di lavoro da esaminare. Stabilire una cadenza regolare per le revisioni delle prestazioni operative e la condivisione delle conoscenze migliora la capacità di ottenere prestazioni più elevate dai team operativi.

Anti-pattern comuni:

- Raccolta dei soli parametri predefiniti.
- Impostazione di una strategia di monitoraggio senza alcuna revisione.
- Nessuna discussione sul monitoraggio quando vengono distribuite modifiche importanti.

Vantaggi dell'adozione di questa best practice: la verifica periodica del monitoraggio consente di prevedere potenziali problemi, invece di rispondere alle notifiche quando un problema previsto si verifica effettivamente.

Livello di rischio associato se questa best practice non fosse adottata: Medium

Guida all'implementazione

- Crea più pannelli di controllo per il carico di lavoro. È necessario disporre di un pannello di controllo di primo livello contenente i parametri aziendali chiave, nonché i parametri tecnici che hai identificato come i più rilevanti per lo stato previsto del carico di lavoro al variare dell'utilizzo. È inoltre importante disporre di pannelli di controllo per vari livelli di applicazione e dipendenze che è possibile ispezionare.
 - [Utilizzo dei pannelli di controllo Amazon CloudWatch](#)
- Pianifica ed effettua revisioni periodiche dei pannelli di controllo del carico di lavoro. Effettua un'ispezione regolare dei pannelli di controllo. La frequenza può essere diversa a seconda di quanto l'ispezione sia approfondita.
 - Ispeziona l'andamento nei parametri. Confronta i valori dei parametri con i valori storici per vedere se ci sono tendenze che potrebbero suggerire l'esame di un particolare aspetto. Riportiamo alcuni esempi: aumento della latenza, riduzione della funzione aziendale primaria e aumento delle risposte all'errore.
 - Identificazione di outlier/anomalie nei parametri. Le medie o mediane possono nascondere outlier e anomalie. Osserva i valori più alti e più bassi nell'intervallo di tempo e analizza le cause dei risultati estremi. Man mano che continui a eliminare tali cause, la riduzione del numero di valori estremi ti consente di continuare a migliorare la coerenza delle prestazioni del carico di lavoro.
 - Ricerca di bruschi cambiamenti nel comportamento. Un cambiamento repentino della quantità o della direzione di un parametro può indicare un cambiamento nell'applicazione o fattori esterni che potrebbero richiedere l'aggiunta di ulteriori parametri da monitorare.

Risorse

Documenti correlati:

- [Query di esempio di Amazon CloudWatch Logs Insights](#)
- [Effettuare il debug con Amazon CloudWatch Synthetics e AWS X-Ray](#)

- [One Observability Workshop](#)
- [The Amazon Builders' Library: Dotazione dei sistemi distribuiti per la visibilità operativa](#)
- [Utilizzo dei pannelli di controllo Amazon CloudWatch](#)

REL06-BP07 Monitoraggio del tracciamento end-to-end delle richieste attraverso il sistema

Tieni traccia delle richieste durante l'elaborazione dei componenti del servizio in modo che i team del prodotto possano analizzare i problemi, semplificarne il debug e migliorare le prestazioni.

Risultato desiderato: I carichi di lavoro con tracciabilità completa di tutti i componenti sono caratterizzati da processi di debug più semplici e ciò migliora il [tempo medio di risoluzione](#) (MTTR) degli errori e la latenza grazie alla semplificazione dell'individuazione delle cause principali. La tracciabilità end-to-end riduce il tempo necessario per individuare i componenti interessati e approfondire in dettaglio le cause principali degli errori o della latenza.

Anti-pattern comuni:

- Il tracciamento viene utilizzato per alcuni componenti ma non per tutti. Ad esempio, senza il tracciamento AWS Lambda, i team potrebbero non avere una chiara comprensione della latenza causata dagli avviamenti a freddo in un periodo di picco del carico di lavoro.
- I canary Synthetics o le metriche RUM (Real-User Monitoring) non sono configurati con il tracciamento. Senza canary o metriche RUM, la telemetria delle interazioni dei clienti viene omessa dall'analisi dei tracciamenti e ciò rende incompleto il profilo delle prestazioni.
- I carichi di lavoro ibridi includono strumenti di tracciamento nativi del cloud e di terze parti, ma non sono state prese misure specifiche per selezionare e integrare completamente un'unica soluzione di tracciamento. In base alla soluzione di tracciamento scelta, gli SDK di tracciamento nativi del cloud devono essere utilizzati per instrumentare i componenti non nativi del cloud oppure è necessario configurare strumenti di terze parti per acquisire i dati telemetrici delle tracce nativi del cloud.

Vantaggi dell'adozione di questa best practice: Quando vengono avvisati della presenza di problemi, i team di sviluppo possono visualizzare un quadro completo delle interazioni tra i componenti del sistema, inclusa la correlazione componente per componente con registrazione, prestazioni e guasti. Poiché il tracciamento semplifica l'identificazione visiva delle cause principali, viene dedicato meno tempo all'individuazione di tali cause. I team che hanno una visione dettagliata delle interazioni tra

i componenti prendono decisioni migliori e più rapide durante la fase di risoluzione dei problemi. Le decisioni, ad esempio quando attivare il failover del ripristino di emergenza o dove implementare in modo più efficace le strategie di riparazione automatica, possono essere migliorate analizzando le tracce dei sistemi; ciò ottimizza in ultima analisi la soddisfazione dei clienti nei confronti dei servizi.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

I team che gestiscono le applicazioni distribuite possono utilizzare strumenti di tracciamento per definire un identificatore di correlazione, raccogliere le tracce delle richieste e creare mappe di servizio dei componenti connessi. Tutti i componenti dell'applicazione devono essere inclusi nelle tracce delle richieste, inclusi client di servizio, gateway middleware e router di eventi, componenti di elaborazione e archiviazione, tra cui gli archivi e i database dei valori chiave. Includi canary Synthetics o metriche RUM (Real-User Monitoring) nella configurazione del tracciamento end-to-end per misurare le interazioni e la latenza dei client remoti in modo da poter valutare con precisione le prestazioni dei tuoi sistemi rispetto agli accordi sul livello di servizio (SLA) e agli obiettivi corrispondenti.

Puoi utilizzare [AWS X-Ray](#) e i servizi di strumentazione di [Monitoraggio delle applicazioni Amazon CloudWatch](#) per avere una visione completa delle richieste man mano che vengono inviate all'applicazione. X-Ray raccoglie la telemetria delle applicazioni e consente di visualizzare e filtrare i dati corrispondenti tra payload, funzioni, tracce, servizi e API. L'acquisizione dei dati telemetrici può essere attivata per i componenti di sistema senza codice o a uso limitato di codice. Monitoraggio delle applicazioni CloudWatch include ServiceLens per integrare le tracce con metriche, log e allarmi. La funzionalità Monitoraggio delle applicazioni CloudWatch include anche elementi Synthetics per monitorare gli endpoint e le API, oltre alle metriche RUM (Real-User Monitoring) per instrumentare i client delle applicazioni Web.

Passaggi dell'implementazione

- Utilizza AWS X-Ray su tutti i servizi nativi supportati come [Amazon S3, AWS Lambda e Amazon API Gateway](#). Questi servizi AWS consentono a X-Ray di attivare opzioni di configurazione utilizzando l'infrastruttura come codice, AWS SDK o la AWS Management Console.
- Esegui l'strumentazione delle applicazioni [AWS Distro per Open Telemetry e X-Ray](#) o degli agenti di raccolta di terze parti.
- Consulta la [Guida per gli sviluppatori AWS X-Ray](#) per l'implementazione di linguaggi di programmazione specifici. Queste sezioni della documentazione descrivono come instrumentare

le richieste HTTP, le query SQL e altri processi specifici del linguaggio di programmazione delle applicazioni.

- Usa il tracciamento X-Ray per [i canary Synthetics di Amazon CloudWatch](#) e le metriche [RUM Amazon CloudWatch](#) per analizzare il percorso delle richieste dal client dell'utente finale attraverso l'infrastruttura AWS downstream.
- Configura le metriche CloudWatch e gli allarmi in base allo stato delle risorse e alla telemetria dei canary in modo che i team siano avvisati tempestivamente in merito ai problemi e possano, quindi, analizzare in dettaglio le tracce e le mappe dei servizi con ServiceLens.
- Abilita l'integrazione X-Ray per gli strumenti di tracciamento di terze parti come [Datadog](#), [New Relico](#) [Dynatrace](#) se utilizzi strumenti di terze parti per la tua soluzione di tracciamento principale.

Risorse

Best practice correlate:

- [REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro \(generazione\)](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)

Documenti correlati:

- [Che cos'è AWS X-Ray?](#)
- [Amazon CloudWatch: monitoraggio delle applicazioni](#)
- [Effettuare il debug con Amazon CloudWatch Synthetics e AWS X-Ray](#)
- [The Amazon Builders' Library: Dotazione dei sistemi distribuiti per la visibilità operativa](#)
- [Integrazione AWS X-Ray con altri servizi AWS](#)
- [AWS Distro per OpenTelemetry e AWS X-Ray](#)
- [Amazon CloudWatch: utilizzo del monitoraggio sintetico](#)
- [Amazon CloudWatch: utilizzo di CloudWatch RUM](#)
- [Installare i canary Amazon CloudWatch Synthetics e gli allarmi Amazon CloudWatch](#)
- [Oltre la disponibilità: comprendere e migliorare la resilienza dei sistemi distribuiti su AWS](#)

Esempi correlati:

- [One Observability Workshop](#)

Video correlati:

- [AWS re:Invent 2022 - How to monitor applications across multiple accounts \(Come monitorare le applicazioni su più account\)](#)
- [Come monitorare le tue applicazioni AWS](#)

Strumenti correlati:

- [AWS X-Ray](#)
- [Amazon CloudWatch](#)
- [Amazon Route 53](#)

Progettazione di un carico di lavoro in grado di adattarsi ai cambiamenti della domanda

Un carico di lavoro scalabile fornisce elasticità per aggiungere o rimuovere risorse automaticamente, in modo che vi sia una stretta corrispondenza con la domanda attuale in un dato momento.

Best practice

- [REL07-BP01 Utilizzo dell'automazione per l'acquisizione o il dimensionamento delle risorse](#)
- [REL07-BP02 Ottenimento di risorse quando viene rilevata la compromissione di un carico di lavoro](#)
- [REL07-BP03 Ottenimento di risorse dopo aver rilevato che sono necessarie più risorse per un carico di lavoro](#)
- [REL07-BP04 Esecuzione di un test di carico sul carico di lavoro](#)

REL07-BP01 Utilizzo dell'automazione per l'acquisizione o il dimensionamento delle risorse

Quando sostituisci risorse danneggiate o esegui il dimensionamento del carico di lavoro, puoi automatizzare il processo utilizzando servizi AWS gestiti, come Amazon S3 e AWS Auto Scaling. Puoi anche utilizzare strumenti di terze parti e SDK AWS per automatizzare il dimensionamento.

I servizi gestiti AWS includono Amazon S3, Amazon CloudFront, AWS Auto Scaling, AWS Lambda, Amazon DynamoDB, AWS Fargate e Amazon Route 53.

AWS Auto Scaling consente di rilevare e sostituire le istanze danneggiate. Inoltre, permette di creare piani di dimensionamento per le risorse, tra cui istanze e parchi istanze [Amazon EC2](#) , attività [Amazon ECS](#) , tabelle e indici [Amazon DynamoDB](#) e repliche di [Amazon Aurora](#) .

Durante il dimensionamento di istanze EC2, assicurati di utilizzare più zone di disponibilità (preferibilmente almeno tre) e di aggiungere o rimuovere capacità per mantenere il bilanciamento tra queste zone. Anche le attività ECS o i pod Kubernetes (quando si utilizza Amazon Elastic Kubernetes Service) devono essere distribuiti su più zone di disponibilità.

Quando utilizzi AWS Lambda, le istanze subiscono un dimensionamento automatico. Ogni volta che viene ricevuta una notifica di evento per la funzione, AWS Lambda individua rapidamente la capacità libera all'interno del parco istanze di calcolo ed esegue il codice fino alla simultaneità allocata. Devi assicurarti che la simultaneità necessaria sia configurata sulla Lambda specifica e nelle tue Service Quotas.

Amazon S3 ricalibra automaticamente le risorse per gestire elevati tassi di richiesta. Ad esempio, l'applicazione può ottenere almeno 3.500 richieste PUT/COPY/POST/DELETE o 5.500 richieste GET /HEAD al secondo per prefisso in un bucket. Non ci sono limiti al numero di prefissi in un bucket. Puoi aumentare le prestazioni di lettura o scrittura parallelizzando le letture. Ad esempio, se crei 10 prefissi in un bucket Amazon S3 per parallelizzare le letture, potresti dimensionare le prestazioni di lettura a 55.000 richieste al secondo.

Configura e utilizza Amazon CloudFront o una rete di distribuzione di contenuti (CDN) attendibile. Una CDN può fornire tempi di risposta più rapidi agli utenti finali e può servire le richieste di contenuti dalla cache, riducendo così la necessità di dimensionare il carico di lavoro.

Anti-pattern comuni:

- Implementare gruppi Auto Scaling per la correzione automatica, ma senza elasticità.
- Utilizzare l'auto scaling per rispondere a grandi aumenti di traffico.
- Distribuire applicazioni altamente stateful, eliminando l'opzione di elasticità.

Vantaggi dell'adozione di questa best practice: L'automazione elimina il potenziale di errori manuali nella distribuzione e nella disattivazione delle risorse. L'automazione elimina il rischio di superamento dei costi e di rifiuto del servizio a causa della risposta lenta alle esigenze di distribuzione o disattivazione.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

- Configura e utilizza AWS Auto Scaling. In questo modo è possibile monitorare le applicazioni e regolare automaticamente la capacità per mantenere prestazioni stabili e prevedibili al minor costo possibile. Grazie ad AWS Auto Scaling, puoi configurare il dimensionamento delle applicazioni per più risorse in vari servizi.
 - [Che cos'è AWS Auto Scaling?](#)
 - Configura il dimensionamento automatico su serie di istanze Spot e istanze Amazon EC2, attività Amazon ECS, indici e tabelle Amazon DynamoDB, repliche Amazon Aurora e applicazioni Marketplace AWS, come applicabile.
 - [Gestione automatica della capacità di throughput con DynamoDB Auto Scaling](#)
 - Utilizza le operazioni delle API di servizi per specificare gli avvisi, le policy di ridimensionamento e i tempi di riscaldamento e raffreddamento.
 - Utilizza Elastic Load Balancing. I sistemi di bilanciamento del carico possono distribuire il carico in base al percorso o alla connettività di rete.
 - [Che cos'è Elastic Load Balancing?](#)
 - Application Load Balancers può distribuire il carico per percorso.
 - [What is an Application Load Balancer? \(Che cos'è un Application Load Balancer?\)](#)
 - Configura un Application Load Balancer per distribuire il traffico su diversi carichi di lavoro in base a un percorso nello stesso nome di dominio.
 - Gli Application Load Balancers possono essere utilizzati per distribuire i carichi in modo da gestire la domanda attraverso l'integrazione con AWS Auto Scaling.
 - [Uso di un sistema di bilanciamento del carico con un gruppo Auto Scaling](#)
 - I Network Load Balancer possono distribuire il carico in base alla connessione.
 - [Che cos'è un Network Load Balancer?](#)
 - Configura un Network Load Balancer per distribuire il traffico su diversi carichi di lavoro tramite TCP o per disporre di un set costante di indirizzi IP per il carico di lavoro.
 - I Network Load Balancer possono essere utilizzati per distribuire i carichi in modo da gestire la domanda attraverso l'integrazione con AWS Auto Scaling.
 - Uso di un provider DNS altamente disponibile I nomi DNS consentono agli utenti di accedere ai carichi di lavoro utilizzando nomi anziché indirizzi IP e distribuire queste informazioni in un ambito definito, solitamente a livello globale per gli utenti del carico di lavoro.
 - [Utilizza Amazon Route 53 o un provider DNS affidabile.](#)

- [Che cos'è Amazon Route 53?](#)
- Utilizza Route 53 per gestire le distribuzioni CloudFront e i load balancer.
 - Individua i domini e i sottodomini da gestire.
 - Crea set di record appropriati utilizzando record ALIAS o CNAME.
 - [Uso dei record](#)
- Utilizza la rete globale AWS per ottimizzare il percorso dagli utenti alle applicazioni. AWS Global Accelerator monitora costantemente l'integrità degli endpoint delle applicazioni e reindirizza il traffico verso endpoint integri in meno di 30 secondi.
 - AWS Global Accelerator è un servizio che migliora la disponibilità e le prestazioni delle applicazioni con utenti locali o globali, fornendo indirizzi IP statici che fungono da punto di ingresso fisso agli endpoint delle applicazioni in una o più regioni AWS, ad esempio Application Load Balancers, Network Load Balancer o istanze Amazon EC2.
 - [Che cos'è AWS Global Accelerator?](#)
- Configura e utilizza Amazon CloudFront o una rete di distribuzione di contenuti (CDN) attendibile. Una rete di distribuzione di contenuti (CDN) può fornire tempi di risposta più rapidi agli utenti finali e soddisfare richieste di contenuti che possono causare un dimensionamento non necessario dei carichi di lavoro.
 - [Che cos'è Amazon CloudFront?](#)
 - Configura le distribuzioni di Amazon CloudFront per i carichi di lavoro oppure utilizza una CDN di terze parti.
 - Puoi limitare l'accesso ai tuoi carichi di lavoro in modo che siano accessibili solo da CloudFront utilizzando gli intervalli di indirizzi IP per CloudFront nelle policy di accesso o nei gruppi di sicurezza degli endpoint.

Risorse

Documenti correlati:

- [Partner APN: partner per la creazione di soluzioni di elaborazione automatizzate](#)
- [AWS Auto Scaling: come funzionano i piani di dimensionamento](#)
- [Marketplace AWS: prodotti che possono essere utilizzati con Auto Scaling](#)
- [Gestione automatica della capacità di throughput con DynamoDB Auto Scaling](#)
- [Uso di un sistema di bilanciamento del carico con un gruppo Auto Scaling](#)

- [Che cos'è AWS Global Accelerator?](#)
- [Che cos'è Amazon EC2 Auto Scaling?](#)
- [Che cos'è AWS Auto Scaling?](#)
- [Che cos'è Amazon CloudFront?](#)
- [Che cos'è Amazon Route 53?](#)
- [Che cos'è Elastic Load Balancing?](#)
- [Che cos'è un Network Load Balancer?](#)
- [What is an Application Load Balancer? \(Che cos'è un Application Load Balancer?\)](#)
- [Uso dei record](#)

REL07-BP02 Ottenimento di risorse quando viene rilevata la compromissione di un carico di lavoro

All'occorrenza, ridimensiona le risorse in modo reattivo se la disponibilità è influenzata per ripristinare la disponibilità del carico di lavoro.

Devi prima configurare i controlli dello stato e i criteri su questi controlli per indicare quando la disponibilità è influenzata dalla mancanza di risorse. Quindi invita il personale appropriato a dimensionare manualmente la risorsa o attivare l'automazione per dimensionarla automaticamente.

Il dimensionamento può essere regolato manualmente in base al carico di lavoro, ad esempio modificando il numero di istanze EC2 in un gruppo Auto Scaling o modificando la velocità di trasmissione effettiva di una tabella DynamoDB tramite la AWS Management Console o la AWS CLI. Tuttavia, l'automazione deve essere utilizzata ogni volta che è possibile (consulta [Utilizzo dell'automazione per l'acquisizione o il dimensionamento delle risorse](#)).

Risultato desiderato: le attività di dimensionamento (automatico o manuale) vengono avviate per ripristinare la disponibilità al rilevamento di un guasto o di un'esperienza degradata del cliente.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

Implementa l'osservabilità e il monitoraggio su tutti i componenti del carico di lavoro, per monitorare l'esperienza del cliente e rilevare i guasti. Definisci le procedure (manuali o automatiche) che dimensionano le risorse richieste. Per ulteriori informazioni, consulta [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#).

Passaggi dell'implementazione

- Definisci le procedure (manuali o automatiche) che dimensionano le risorse richieste.
 - Le procedure di dimensionamento dipendono da come sono progettati i diversi componenti del carico di lavoro.
 - Le procedure di dimensionamento variano anche a seconda della tecnologia sottostante utilizzata.
 - I componenti che utilizzano AWS Auto Scaling possono impiegare piani di dimensionamento per configurare una serie di istruzioni per dimensionare le risorse. Se utilizzi AWS CloudFormation o aggiungi tag alle risorse AWS, puoi impostare piani di dimensionamento per diversi set di risorse, per ogni applicazione. Auto Scaling fornisce raccomandazioni per strategie di dimensionamento personalizzate per ogni risorsa. Dopo aver creato il piano, Auto Scaling combina i metodi di dimensionamento dinamico e predittivo per supportare la tua strategia di dimensionamento. Per maggiori dettagli, consulta [Come funzionano i piani di dimensionamento](#).
 - Amazon EC2 Auto Scaling garantisce che sia disponibile il numero corretto di istanze Amazon EC2 per gestire il carico dell'applicazione. È possibile creare raccolte di istanze EC2, denominate gruppi Auto Scaling. Puoi specificare il numero minimo e massimo di istanze in ogni gruppo Auto Scaling; Amazon EC2 Auto Scaling garantisce che il gruppo non superi mai questi limiti. Per maggiori dettagli, vedi [What is Amazon EC2 Auto Scaling?](#)
 - Il dimensionamento automatico Amazon DynamoDB utilizza il servizio Application Auto Scaling per regolare dinamicamente la capacità effettiva di trasmissione allocata per tuo conto, in risposta ai modelli di traffico effettivi. Ciò consente a una tabella o a un indice secondario globale di aumentare la capacità di lettura e scrittura assegnata per gestire aumenti di traffico improvvisi, senza limitazione della larghezza di banda della rete. Per maggiori dettagli, consulta [Gestione automatica della capacità effettiva di trasmissione con il dimensionamento automatico di DynamoDB](#).

Risorse

Best practice correlate:

- [REL07-BP01 Utilizzo dell'automazione per l'acquisizione o il dimensionamento delle risorse](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)

Documenti correlati:

- [AWS Auto Scaling: Come funzionano i piani di dimensionamento](#)
- [Gestione automatica della capacità effettiva di trasmissione con il dimensionamento automatico di DynamoDB](#)
- [What is Amazon EC2 Auto Scaling?](#)

REL07-BP03 Ottenimento di risorse dopo aver rilevato che sono necessarie più risorse per un carico di lavoro

Dimensiona le risorse in modo proattivo per soddisfare la domanda ed evitare l'impatto sulla disponibilità.

Molti servizi AWS dimensionano automaticamente le risorse per soddisfare la domanda. Se si utilizzano istanze Amazon EC2 o cluster Amazon ECS, puoi configurare la scalabilità automatica di tali istanze in base ai parametri di utilizzo corrispondenti alla richiesta del carico di lavoro. Per Amazon EC2, è possibile impiegare l'utilizzo medio della CPU, il conteggio delle richieste del sistema di bilanciamento del carico o la larghezza di banda di rete per aumentare (o ridurre) le istanze EC2. Per Amazon ECS, è possibile impiegare l'utilizzo medio della CPU, il conteggio delle richieste del load balancer e l'utilizzo della memoria per aumentare orizzontalmente (o ridurre orizzontalmente) le attività ECS. Utilizzando il dimensionamento automatico di destinazione su AWS, l'autoscaler si comporta come un termostato domestico, aggiungendo o rimuovendo risorse per mantenere il valore di destinazione (ad esempio, il 70% di utilizzo della CPU) specificato.

AWS Auto Scaling può anche eseguire l' [Auto Scaling predittivo](#), che utilizza il machine learning per analizzare il carico di lavoro cronologico di ciascuna risorsa e prevede regolarmente il carico futuro per i due giorni successivi.

La legge di Little aiuta a calcolare il numero di istanze di calcolo (istanze EC2, funzioni Lambda simultanee, ecc.) necessarie.

$$L = \lambda W$$

L = numero di istanze (o simultaneità media nel sistema)

λ = velocità media alla quale arrivano le richieste (richieste/sec)

W = tempo medio trascorso da ogni richiesta nel sistema (sec)

Ad esempio, a 100 rps, se ogni richiesta impiega 0,5 secondi per l'elaborazione, avrai bisogno di 50 istanze per tenere il passo con la domanda.

Livello di rischio associato se questa best practice non fosse adottata: Medium

Guida all'implementazione

- Ottieni risorse dopo aver rilevato che sono necessarie più risorse per un carico di lavoro. Dimensiona le risorse in modo proattivo per soddisfare la domanda ed evitare l'impatto sulla disponibilità.
- Valuta quante risorse di calcolo sono necessarie (simultaneità di calcolo) per gestire un determinato tasso di richiesta
 - [Telling Stories About Little's Law](#)
- Quando disponi di un modello cronologico per l'utilizzo, imposta il dimensionamento programmato per il dimensionamento automatico Amazon EC2.
 - [Dimensionamento programmato per Amazon EC2 Auto Scaling](#)
- Utilizza il dimensionamento predittivo di AWS.
 - [Dimensionamento predittivo per EC2, alimentato dal machine learning](#)

Risorse

Documenti correlati:

- [AWS Auto Scaling: come funzionano i piani di dimensionamento](#)
- [Marketplace AWS: prodotti che possono essere utilizzati con Auto Scaling](#)
- [Gestione automatica della capacità di throughput con DynamoDB Auto Scaling](#)
- [Dimensionamento predittivo per EC2, alimentato dal machine learning](#)
- [Dimensionamento programmato per Amazon EC2 Auto Scaling](#)
- [Telling Stories About Little's Law](#)
- [Che cos'è Amazon EC2 Auto Scaling?](#)

REL07-BP04 Esecuzione di un test di carico sul carico di lavoro

Adotta un metodo di test del carico per misurare se l'attività di dimensionamento soddisfa i requisiti del carico di lavoro.

È importante eseguire test di carico prolungati. I test di carico devono rilevare il punto di rottura e testare le prestazioni del carico di lavoro. AWS consente di creare facilmente ambienti di test

temporanei che riproducono la scala del carico di lavoro di produzione. Nel cloud, puoi creare un ambiente di test su scala produttiva on demand, completare i test e disattivare le risorse. Poiché paghi per l'ambiente di test solo quando è in esecuzione, puoi simulare un ambiente live a un costo notevolmente inferiore rispetto ai test in locale.

I test di carico in produzione dovrebbero anche essere considerati come parte dei game day in cui il sistema di produzione viene messo alla prova, durante le ore di utilizzo inferiore del cliente, con tutto il personale a disposizione per interpretare i risultati e risolvere eventuali problemi che si presentano.

Anti-pattern comuni:

- Eseguire test di carico su distribuzioni che non presentano la stessa configurazione della tua produzione.
- Eseguire test di carico solo su singole parti del carico di lavoro e non sulla sua interezza.
- Eseguire test di carico con un sottoinsieme di richieste e non con un set rappresentativo delle richieste effettive.
- Eseguire test di carico su un fattore di sicurezza di poco superiore al carico previsto.

Vantaggi dell'adozione di questa best practice: Saprai quali sono i componenti dell'architettura che non funzionano sotto carico e potrai identificare per tempo i parametri che indicano l'avvicinamento al carico in questione, così da affrontare il problema e prevenire l'impatto dell'esito negativo.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

- Esegui test di carico per identificare quali aspetti del carico di lavoro indicano la necessità di aggiungere o rimuovere capacità. Il test di carico deve avere un traffico rappresentativo simile a quello che ricevi nella produzione. Aumenta il carico mentre osservi i parametri implementati per stabilire quale di questi indica quando è necessario aggiungere o rimuovere risorse.
- [Distributed Load Testing on AWS \(Test di carico distribuito su AWS\): simula migliaia di utenti connessi](#)
 - Identifica la combinazione di richieste. Potresti avere diverse combinazioni di richieste, quindi dovresti esaminare vari intervalli di tempo per identificare la combinazione di traffico.
 - Implementa un driver di caricamento. Puoi utilizzare codice personalizzato, software open source o software commerciale per implementare un driver di carico.

- Esegui un test di carico iniziale con una capacità ridotta. Puoi vedere alcuni effetti immediati applicando il carico su una capacità inferiore, possibilmente pari a un'istanza o a un container.
- Esegui un test di carico con una capacità maggiore. Gli effetti saranno diversi su un carico distribuito, quindi è necessario eseguire il test in condizioni quanto più simili possibili all'ambiente del prodotto.

Risorse

Documenti correlati:

- [Distributed Load Testing on AWS \(Test di carico distribuito su AWS\): simula migliaia di utenti connessi](#)
- [Load testing applications](#)

Video correlati:

- [AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing](#)

Implementazione della modifica

Le modifiche controllate sono necessarie per distribuire nuove funzionalità e per garantire che i carichi di lavoro e l'ambiente operativo eseguano software noti con patch corrette. Se invece non sono controllate, risulta difficile prevederne l'effetto o risolvere eventuali problemi che causano.

Best practice

- [REL08-BP01 Utilizzo di runbook per attività standard come l'implementazione](#)
- [REL08-BP02 Esecuzione di test funzionali come parte integrante dell'implementazione](#)
- [REL08-BP03 Esecuzione di test di resilienza come parte integrante dell'implementazione](#)
- [REL08-BP04 Esecuzione dell'implementazione utilizzando un'infrastruttura immutabile](#)
- [REL08-BP05 Implementazione delle modifiche tramite automazione](#)

REL08-BP01 Utilizzo di runbook per attività standard come l'implementazione

I runbook sono le procedure predefinite per ottenere risultati specifici. Utilizza i runbook per eseguire attività standard, o manualmente o automaticamente. Alcuni esempi includono l'implementazione di un carico di lavoro, l'applicazione di patch a un carico di lavoro o la realizzazione di modifiche DNS.

Ad esempio, metti in atto processi per [garantire la sicurezza del rollback durante le distribuzioni](#). Garantire la possibilità di eseguire il rollback di una distribuzione senza interruzioni per i clienti è fondamentale per rendere un servizio affidabile.

Per le procedure di runbook, inizia da un processo manuale valido ed efficace, implementalo nel codice e attivalo per l'esecuzione automatica, se necessario.

Anche per carichi di lavoro sofisticati e altamente automatizzati, i runbook rimangono utili per [eseguire game day](#) o soddisfare rigorosi requisiti di reportistica e audit.

Tieni presente che i playbook vengono utilizzati in risposta a incidenti specifici e i runbook vengono utilizzati per ottenere risultati specifici. Spesso, i runbook sono per attività di routine, mentre i playbook vengono utilizzati per rispondere a eventi non di routine.

Anti-pattern comuni:

- Eseguire modifiche impreviste alla configurazione nella produzione.
- Ignorare le fasi del piano per velocizzare l'implementazione, compromettendone la riuscita.
- Apportare modifiche senza testarne l'annullamento.

Vantaggi dell'adozione di questa best practice: Una pianificazione efficace aumenta la capacità di eseguire correttamente la modifica, perché sei a conoscenza di tutti i sistemi interessati. Convalidare la modifica negli ambienti di test aumenta la sicurezza.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

- Abilita risposte coerenti e tempestive agli eventi noti documentando le procedure nei runbook.
- [Framework AWS Well-Architected – Concetti – Runbook](#)

- Uso del principio di infrastruttura come codice per definire l'infrastruttura Utilizzando AWS CloudFormation o una terza parte affidabile per definire la tua infrastruttura, puoi utilizzare un software per il controllo delle versioni per gestire le versioni e tenere traccia delle modifiche.
- Utilizza AWS CloudFormation o un provider di terze parti affidabile per definire l'infrastruttura.
 - [Che cos'è AWS CloudFormation?](#)
- Crea modelli unici e disaccoppiati, utilizzando solidi principi di progettazione del software.
 - Stabilisci le autorizzazioni, i modelli e le parti responsabili dell'implementazione
 - [Controllo degli accessi con AWS Identity and Access Management](#)
 - Utilizza un controllo sorgente come AWS CodeCommit o uno strumento di terze parti affidabili per il controllo delle versioni.
 - [Che cos'è AWS CodeCommit?](#)

Risorse

Documenti correlati:

- [Partner APN: partner per la creazione di soluzioni di distribuzione automatizzate](#)
- [Marketplace AWS: prodotti per l'automazione delle distribuzioni](#)
- [Framework AWS Well-Architected – Concetti – Runbook](#)
- [Che cos'è AWS CloudFormation?](#)
- [Che cos'è AWS CodeCommit?](#)

Esempi correlati:

- [Automating operations with Playbooks and Runbooks \(Automazione delle operazioni con Playbook e Runbook\)](#)

REL08-BP02 Esecuzione di test funzionali come parte integrante dell'implementazione

I test funzionali vengono eseguiti come parte integrante dell'implementazione automatizzata. Se non vengono soddisfatti i criteri di esito positivo, la pipeline viene arrestata o ripresa dall'inizio. Questi test vengono eseguiti in un ambiente di pre-produzione, gestito per fasi prima della produzione nella pipeline. Idealmente, questa operazione viene eseguita come parte di una pipeline di implementazione.

Risultato desiderato: utilizzi l'automazione per eseguire test funzionali e i dati dei test associati riducono la durata e il costo dei test e migliorano l'accuratezza dei risultati. Integri i test funzionali come parte del processo di implementazione per automatizzare le pipeline di rilascio per l'esecuzione di aggiornamenti rapidi e affidabili di applicazioni e infrastrutture.

Anti-pattern comuni:

- I test vengono eseguiti manualmente al di fuori della pipeline di implementazione.
- Non esegui le fasi di test nell'automazione tramite flussi di lavoro manuali di emergenza.
- Non segui i piani e i processi di test stabiliti a favore di tempistiche accelerate.

Vantaggi dell'adozione di questa best practice: i test funzionali verificano che il sistema funzioni secondo i requisiti specificati. Vengono utilizzati per verificare in modo coerente il funzionamento previsto di componenti quali interfacce utente, API, database e codice sorgente. Quando esamini questi componenti del sistema, i test funzionali verificano che ciascuna funzionalità si comporti come previsto, tutelando sia le esigenze degli utenti sia l'integrità del software. Integra i test funzionali come parte dell'implementazione regolare e utilizza l'automazione per implementare tutte le modifiche, riducendo i potenziali errori umani.

Livello di rischio associato se questa best practice non fosse adottata: elevato

Guida all'implementazione

Esegui test funzionali come parte integrante dell'implementazione. I test funzionali vengono eseguiti come parte integrante dell'implementazione automatizzata. Se non vengono soddisfatti i criteri di esito positivo, la pipeline viene arrestata o ripristinata. AWS CodePipeline fornisce una pipeline di distribuzione continua per i test automatizzati, che consente ai tester di automatizzare l'intero processo di test e implementazione. Si integra con i servizi AWS come AWS CodeBuild e AWS CodeDeploy per automatizzare le fasi di creazione, test e implementazione del ciclo di vita di sviluppo del software.

Passaggi dell'implementazione

- Configura la pipeline: configura le fasi di origine, creazione, test e implementazione utilizzando la console AWS CodePipeline o AWS Command Line Interface (CLI).
- Definisci l'origine: con AWS CodePipeline puoi recuperare automaticamente il codice sorgente da sistemi di controllo delle versioni come GitHub, AWS CodeCommit o Bitbucket e assicurarti che per i test venga sempre utilizzato il codice più recente.

- Automatizza build e test: AWS CodeBuild può creare e testare automaticamente il codice e generare i report di test. Supporta i framework di test più diffusi come JUnit, NUnit e TestNG.
- Implementa il codice: una volta che il codice è stato creato e testato, AWS CodeDeploy può implementarlo nell'ambiente di test, incluse le istanze Amazon EC2, le funzioni AWS Lambda o i server on-premises.
- Monitora le pipeline: con AWS CodePipeline puoi monitorare l'avanzamento della pipeline e lo stato di ogni fase. Puoi anche utilizzare i controlli di qualità per bloccare la pipeline in base allo stato di esecuzione dei test. Puoi anche ricevere notifiche per qualsiasi errore che si verifica durante l'esecuzione o il completamento della pipeline.

Risorse

Documenti correlati:

- [Use AWS CodePipeline with AWS CodeBuild to test code and run builds](#)
- [Logging and monitoring in AWS CodeBuild](#)
- [Indicators for functional testing](#)

REL08-BP03 Esecuzione di test di resilienza come parte integrante dell'implementazione

Integra i test di resilienza introducendo consapevolmente errori nel sistema per misurarne la capacità in caso di scenari destabilizzanti. I test di resilienza, diversamente dai test funzionali e dagli unit test che di solito sono integrati nei cicli di implementazione, si concentrano sull'identificazione di errori imprevisti nel sistema. Puoi iniziare l'integrazione dei test di resilienza nella fase di pre-produzione, ma stabilisci l'obiettivo di implementare questi test in produzione durante le [giornate di gioco](#).

Risultato desiderato: i test di resilienza favoriscono la fiducia nella capacità del sistema di reggere al danneggiamento in produzione. Gli esperimenti identificano i punti di debolezza che potrebbero causare errori, consentendoti di migliorare il sistema per mitigare automaticamente ed efficacemente errori e danneggiamento.

Anti-pattern comuni:

- Mancanza di osservabilità e monitoraggio nei processi di implementazione.
- Dipendenza dagli esseri umani per risolvere gli errori del sistema.

- Meccanismi di analisi di scarsa qualità.
- Supporto per i problemi noti del sistema e mancanza di sperimentazione per identificare eventuali incognite.
- Identificazione degli errori, ma nessuna risoluzione.
- Nessuna documentazione degli esiti e dei runbook.

Vantaggi dell'adozione di questa best practice: i test di resilienza integrati nelle implementazioni aiutano a identificare problemi non noti del sistema che altrimenti passano inosservati, causando tempi di inattività in produzione. L'identificazione di queste incognite nel sistema ti consente di documentare gli esiti, integrare i test nel processo CI/CD e creare runbook che semplificano la mitigazione attraverso meccanismi efficienti e ripetibili.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

I moduli di test di resilienza più comuni che possono essere integrati nelle implementazioni del sistema sono il ripristino di emergenza e l'ingegneria del caos.

- Includi gli aggiornamenti ai piani di ripristino di emergenza e alle procedure operative standard (SOP) con qualsiasi implementazione significativa.
- Integra i test di affidabilità nelle pipeline di implementazione automatizzate. Servizi come [AWS Resilience Hub](#) possono essere [integrati nella pipeline CI/CD](#) per stabilire valutazioni continue della resilienza che vengono eseguite automaticamente come parte di ogni implementazione.
- Definisci le applicazioni in AWS Resilience Hub. Le valutazioni della resilienza generano frammenti di codice che consentono di creare procedure di ripristino come i documenti AWS Systems Manager per le applicazioni e forniscono un elenco di controlli e allarmi Amazon CloudWatch consigliati.
- Una volta aggiornati i piani di ripristino di emergenza e le SOP, completa i test di ripristino di emergenza per verificarne l'efficacia. I test di ripristino di emergenza consentono di determinare se è possibile ripristinare il sistema dopo un evento e tornare alle normali operazioni. Puoi simulare varie strategie di ripristino di emergenza e determinare se la pianificazione è sufficiente a soddisfare i requisiti di operatività. Le strategie di ripristino di emergenza più comuni includono backup e ripristino, pilot light, cold standby, warm standby, standby a caldo e attivo-attivo e si differenziano tutte per costi e complessità. Prima dei test di ripristino di emergenza, ti consigliamo di definire l'obiettivo del tempo di ripristino (RTO) e l'obiettivo del punto di ripristino (RPO) per

semplificare la scelta della strategia da simulare. AWS offre strumenti di ripristino di emergenza come [AWS Elastic Disaster Recovery](#) per aiutarti a iniziare la pianificazione e i test.

- Gli esperimenti di ingegneria del caos introducono interruzioni nel sistema, come interruzioni di rete ed errori del servizio. Simulando con gli errori controllati, puoi scoprire le vulnerabilità del sistema contenendo al contempo l'impatto degli errori inseriti. Proprio come le altre strategie, esegui le simulazioni controllate degli errori in ambienti non di produzione utilizzando servizi come [AWS Fault Injection Service](#) per acquisire sicurezza prima dell'implementazione in produzione.

Risorse

Documenti correlati:

- [Experiment with failure using resilience testing to build recovery preparedness](#)
- [Continually assessing application resilience with AWS Resilience Hub and AWS CodePipeline](#)
- [Disaster recovery \(DR\) architecture on AWS, part 1: Strategies for recovery in the cloud](#)
- [Verify the resilience of your workloads using Chaos Engineering](#)
- [Principles of Chaos Engineering](#)
- [Chaos Engineering Workshop](#)

Video correlati:

- [AWS re:Invent 2020: Testing Resilience using Chaos Engineering](#)
- [Improve Application Resilience with AWS Fault Injection Service](#)
- [Prepare & Protect Your Applications From Disruption With AWS Resilience Hub](#)

REL08-BP04 Esecuzione dell'implementazione utilizzando un'infrastruttura immutabile

L'infrastruttura immutabile è un modello che richiede che non vengano applicati aggiornamenti, patch di sicurezza o modifiche di configurazione sui carichi di lavoro di produzione. Quando è necessaria una modifica, l'architettura viene costruita su una nuova infrastruttura e distribuita alla produzione.

Segui una strategia di implementazione dell'infrastruttura immutabile per aumentare l'affidabilità, la coerenza e la riproducibilità nelle implementazioni dei carichi di lavoro.

Risultato desiderato: con un'infrastruttura immutabile, non sono consentite [modifiche locali \(in-place\)](#) per l'esecuzione delle risorse dell'infrastruttura all'interno di un carico di lavoro. Invece, quando è necessaria una modifica, un nuovo set di risorse infrastrutturali aggiornate contenente tutte le modifiche necessarie viene implementato in parallelo alle risorse esistenti. Questa implementazione viene convalidata automaticamente e, in caso di successo, il traffico viene gradualmente trasferito al nuovo set di risorse.

Questa strategia di implementazione si applica, ad esempio, agli aggiornamenti software, alle patch di sicurezza, alle modifiche apportate all'infrastruttura, agli aggiornamenti della configurazione e agli aggiornamenti delle applicazioni.

Anti-pattern comuni:

- Implementazione locale (in-place) di modifiche alle risorse dell'infrastruttura in esecuzione.

Vantaggi dell'adozione di questa best practice:

- Maggiore coerenza tra ambienti: poiché non vi sono differenze nelle risorse dell'infrastruttura tra ambienti, la coerenza aumenta e i test risultano semplificati.
- Riduzione delle deviazioni di configurazione: sostituendo le risorse dell'infrastruttura con una configurazione nota e controllata dalla versione, l'infrastruttura viene reimpostata a uno stato noto, testato e attendibile, evitando in questo modo deviazioni di configurazione.
- Implementazioni atomiche affidabili: le implementazioni vengono completate correttamente o, in caso contrario, non generano alcun cambiamento, aumentando così la coerenza e l'affidabilità nel processo di implementazione.
- Implementazioni semplificate: le implementazioni sono semplificate perché non devono supportare gli aggiornamenti. Gli aggiornamenti sono solo nuove distribuzioni.
- Implementazioni più sicure con processi di rollback e ripristino rapidi: le implementazioni sono più sicure perché la versione funzionante precedente non viene modificata. Puoi eseguire il rollback se vengono rilevati errori.
- Potenziamento del profilo di sicurezza: non consentendo modifiche all'infrastruttura, i meccanismi di accesso remoto (come SSH) possono essere disabilitati. Questo riduce il vettore di attacco, migliorando il profilo di sicurezza dell'organizzazione.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

Automazione

Quando si definisce una strategia di implementazione immutabile dell'infrastruttura, si consiglia di utilizzare l'[automazione](#) il più possibile per aumentare la riproducibilità e ridurre al minimo i potenziali errori umani. Per maggiori dettagli, consulta [REL08-BP05 Implementazione delle modifiche tramite automazione](#) e [Automazione di implementazioni pratiche e sicure](#).

Con il modello [Infrastructure as code \(IaC\)](#), le fasi di provisioning, orchestrazione e implementazione dell'infrastruttura sono definite in modo programmatico, descrittivo e dichiarativo e archiviate in un sistema di controllo del codice sorgente. L'utilizzo del modello Infrastructure as code (IaC) semplifica l'automazione dell'implementazione dell'infrastruttura e aiuta a raggiungere l'immutabilità dell'infrastruttura.

Schemi di implementazione

Quando è richiesta una modifica del carico di lavoro, la strategia di implementazione immutabile dell'infrastruttura impone l'implementazione di un nuovo set di risorse dell'infrastruttura, comprese tutte le modifiche necessarie. È importante che questo nuovo set di risorse si basi su un modello di implementazione che riduca al minimo l'impatto sugli utenti. Esistono due strategie principali per questa implementazione:

[Implementazione Canary](#): pratica di indirizzare un piccolo numero di clienti alla nuova versione, in genere in esecuzione su una singola istanza di servizio (la release Canary). Quindi analizzerai in modo approfondito le modifiche di comportamento o gli errori generati. Puoi rimuovere il traffico dalla release Canary in caso di problemi critici e reindirizzare gli utenti alla versione precedente. Se l'implementazione viene completata correttamente, puoi continuare l'implementazione alla velocità desiderata, monitorando le modifiche alla ricerca di errori, fino al completamento dell'implementazione. AWS CodeDeploy può essere configurato con una [configurazione di implementazione](#) che abilita un'implementazione Canary.

[Implementazione blu/verde](#): simile all'implementazione Canary, tranne per il fatto che viene implementato in parallelo un intero parco istanze dell'applicazione. Puoi alternare le distribuzioni tra i due stack (blue e green). Ancora una volta, puoi inviare il traffico alla nuova versione e tornare alla versione precedente in caso di problemi con la distribuzione. Generalmente, tutto il traffico viene trasferito contemporaneamente, tuttavia puoi anche utilizzare frazioni del traffico verso ciascuna versione per comporre l'adozione della nuova versione utilizzando le funzionalità di indirizzamento DNS ponderato di Amazon Route 53. AWS CodeDeploy e [AWS Elastic Beanstalk](#) possono essere configurati con una configurazione di distribuzione che abilita un'implementazione blu/verde.

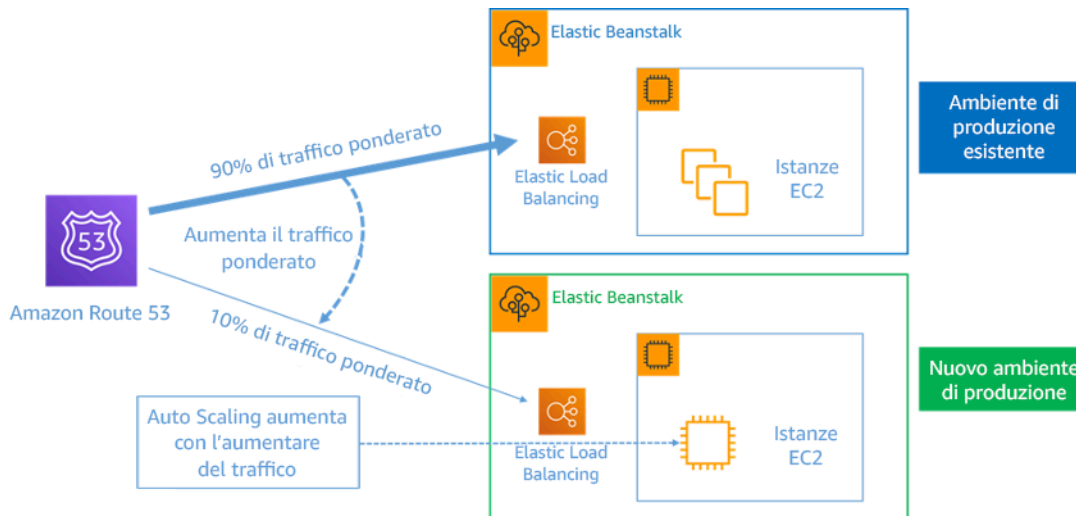


Figura 8. Implementazione blu/verde con AWS Elastic Beanstalk e Amazon Route 53

Rilevamento della deviazione

La deviazione è definita come qualsiasi modifica che fa sì che una risorsa dell'infrastruttura abbia uno stato o una configurazione diversi dal previsto. Qualsiasi tipo di modifica non gestita della configurazione è contraria al concetto di infrastruttura immutabile e tale modifica dovrebbe essere individuata e corretta per implementare con successo l'infrastruttura immutabile.

Passaggi dell'implementazione

- Non autorizzare la modifica locale (in-place) delle risorse dell'infrastruttura in esecuzione.
- Puoi usare [AWS Identity and Access Management \(IAM\)](#) per specificare chi o cosa può accedere a servizi e risorse in AWS, gestire centralmente le autorizzazioni a elevata granularità e analizzare l'accesso per perfezionare le autorizzazioni in AWS.
- Automatizza l'implementazione delle risorse dell'infrastruttura per aumentare la riproducibilità e ridurre al minimo i potenziali errori umani.
 - Come descritto nel whitepaper [Introduzione a DevOps in AWS](#), l'automazione è un elemento fondamentale per i servizi AWS ed è supportata internamente in tutti i servizi, le funzionalità e le offerte.
 - La [preparazione preliminare](#) delle Amazon Machine Image (AMI) può velocizzare i tempi di avvio. [EC2 Image Builder](#) è un servizio AWS completamente gestito che consente di automatizzare la creazione, la manutenzione, la convalida, la condivisione e l'implementazione di AMI personalizzate, sicure e aggiornate per Linux o Windows.
- Alcuni dei servizi che supportano l'automazione sono:

- [AWS Elastic Beanstalk](#) è un servizio per implementare e dimensionare rapidamente applicazioni Web sviluppate con Java, .NET, PHP, Node.js, Python, Ruby, Go e Docker su server tradizionali come Apache, NGINX, Passenger e IIS.
- [AWS Proton](#) aiuta i team della piattaforma a connettere e coordinare tutti i diversi strumenti di cui i team di sviluppo hanno bisogno per il provisioning dell'infrastruttura, le implementazioni del codice, il monitoraggio e gli aggiornamenti. AWS Proton abilita il provisioning e l'implementazione basati sul modello IaC di applicazioni serverless e basate su container.
- L'utilizzo del modello Infrastructure as code (IaC) semplifica l'automazione dell'implementazione dell'infrastruttura e aiuta a raggiungere l'immutabilità dell'infrastruttura. AWS fornisce servizi che consentono la creazione, l'implementazione e la manutenzione dell'infrastruttura in modo programmatico, descrittivo e dichiarativo.
- [AWS CloudFormation](#) aiuta gli sviluppatori a creare risorse AWS in modo ordinato e prevedibile. Le risorse sono scritte in file di testo utilizzando il formato JSON o YAML. I modelli richiedono una sintassi e una struttura specifiche che dipendono dai tipi di risorse create e gestite. Crea le tue risorse in formato JSON o YAML con qualsiasi editor di codice, ad esempio AWS Cloud9, e le inserisci in un sistema di controllo delle versioni. A questo punto, CloudFormation crea i servizi specificati in modo sicuro e ripetibile.
- [AWS Serverless Application Model \(AWS SAM\)](#) è un framework open source che puoi utilizzare per creare applicazioni serverless in AWS. AWS SAM si integra con altri servizi AWS ed è un'estensione di AWS CloudFormation.
- [AWS Cloud Development Kit \(AWS CDK\)](#) è un framework di sviluppo di software open source per modellare ed effettuare il provisioning delle risorse delle applicazioni cloud utilizzando linguaggi di programmazione noti. È possibile utilizzare AWS CDK per modellare l'infrastruttura dell'applicazione mediante TypeScript, Python, Java e .NET. AWS CDK utilizza AWS CloudFormation in background per fornire risorse in modo sicuro e ripetibile.
- [AWS Cloud Control API](#) introduce un set comune di API Create, Read, Update, Delete and List (CRUDL) per aiutare gli sviluppatori a gestire la propria infrastruttura cloud in modo semplice e coerente. Le API Cloud Control API comuni consentono agli sviluppatori di gestire in modo uniforme il ciclo di vita di AWS e i servizi di terze parti.
- Applica modelli di implementazione che riducano al minimo l'impatto sugli utenti.
 - Implementazioni canary:
 - [Configura un'implementazione di una release canary API Gateway](#)
 - [Crea una pipeline con implementazioni canary per Amazon ECS mediante AWS App Mesh](#)

- Implementazioni blu/verdi: il whitepaper relativo alle [implementazioni blu/verdi in AWS](#) descrive [esempi di tecniche](#) per applicare le strategie di implementazione blu/verde.
- Rileva le deviazioni a livello di configurazione o stato. Per maggiori dettagli, consulta [Rilevamento di modifiche non gestite della configurazione di stack e risorse](#).

Risorse

Best practice correlate:

- [REL08-BP05 Implementazione delle modifiche tramite automazione](#)

Documenti correlati:

- [Automazione di implementazioni pratiche e sicure](#)
- [Utilizzo di AWS CloudFormation per creare un'infrastruttura immutabile presso Nubank](#)
- [Scrittura dell'infrastruttura come codice](#)
- [Implementazione di un allarme per rilevare automaticamente la deviazione negli stack AWS CloudFormation](#)

Video correlati:

- [AWS re:Invent 2020: Reliability, consistency, and confidence through immutability](#)

REL08-BP05 Implementazione delle modifiche tramite automazione

Le distribuzioni e l'applicazione di patch sono automatizzate per eliminare l'impatto negativo.

Apportare modifiche ai sistemi produttivi è una delle maggiori aree di rischio per molte organizzazioni. Riteniamo che le implementazioni siano un problema prioritario da risolvere insieme ai problemi aziendali affrontati dal software. Oggi, ciò significa l'uso dell'automazione ovunque sia pratica nelle operazioni, inclusi test e implementazione di modifiche, aggiunta o rimozione di capacità e migrazione dei dati.

Risultato desiderato: integri la sicurezza dell'implementazione automatizzata nel processo di rilascio con test approfonditi di pre-produzione, rollback automatici e implementazioni di produzione scaglionate. Questa automazione riduce al minimo il potenziale impatto sulla produzione causato

da implementazioni non riuscite e gli sviluppatori non devono più monitorare attivamente le implementazioni in produzione.

Anti-pattern comuni:

- Esegui le modifiche manualmente.
- Non esegui le fasi nell'automazione tramite flussi di lavoro manuali di emergenza.
- Non segui i piani e i processi stabiliti a favore di tempistiche accelerate.
- Esegui implementazioni successive rapide senza attendere i tempi di incorporamento.

Vantaggi dell'adozione di questa best practice: quando utilizzi l'automazione per implementare tutte le modifiche, elimini il rischio di errori umani e fornisci la possibilità di eseguire i test prima di modificare la produzione. L'esecuzione di questo processo prima del passaggio in produzione verifica che i piani siano completi. Inoltre, il rollback automatico del processo di rilascio può identificare i problemi di produzione e riportare il carico di lavoro allo stato operativo precedente.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

Automatizzazione della pipeline di distribuzione Le pipeline di implementazione permettono di richiamare test automatici, rilevare le anomalie e interrompere la pipeline a una determinata fase prima dell'implementazione in produzione o eseguire automaticamente il ripristino di una modifica. Parte integrante è l'adozione della cultura basata su [integrazione continua e distribuzione/implementazione continua](#) (CI/CD), in cui un commit o una modifica del codice passa attraverso vari controlli automatizzati dalle fasi di creazione e test all'implementazione negli ambienti di produzione.

Anche se la prassi comune suggerisce di includere le persone nelle procedure operative più difficili, suggeriamo di automatizzare le procedure più difficili proprio per questo motivo.

Passaggi dell'implementazione

Per automatizzare le implementazioni ed eliminare le operazioni manuali, segui questi passaggi:

- Configura un repository di codice per archiviare il codice in modo sicuro: usa [AWS CodeCommit](#) per creare un repository sicuro basato su Git.
- Configura un servizio di integrazione continua per compilare il codice sorgente, eseguire i test e creare gli artefatti di implementazione: per impostare un progetto di compilazione per questo scopo, consulta [Getting started with AWS CodeBuild using the console](#).

- Configura un servizio di implementazione che automatizzi le implementazioni delle applicazioni e gestisca la complessità degli aggiornamenti delle applicazioni senza dipendere da implementazioni manuali soggette a errori: [AWS CodeDeploy](#) automatizza le implementazioni del software su una varietà di servizi di calcolo, come Amazon EC2, [AWS Fargate](#), [AWS Lambda](#) e i server locali. Per configurare questi passaggi, consulta [Getting started with CodeDeploy](#).
- Configura un servizio di distribuzione continua che automatizzi le pipeline di rilascio per eseguire aggiornamenti più rapidi e affidabili di applicazioni e infrastrutture: prendi in considerazione [AWS CodePipeline](#) per automatizzare le pipeline di rilascio. Per maggiori dettagli, consulta [CodePipeline tutorials](#).

Risorse

Best practice correlate:

- [OPS05-BP04 Utilizzo di sistemi di gestione della compilazione e implementazione](#)
- [OPS05-BP10 Automazione completa dell'integrazione e dell'implementazione](#)
- [OPS06-BP02 Implementazioni dei test](#)
- [OPS06-BP04 Automazione dei test e del rollback](#)


Documenti correlati:

- [Continuous Delivery of Nested AWS CloudFormation Stacks Using AWS CodePipeline](#)
- [Complete CI/CD with AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline](#)
- [APN Partner: partners that can help you create automated deployment solutions](#)
- [Marketplace AWS: products that can be used to automate your deployments](#)
- [Automate chat messages with webhooks.](#)
- [Amazon Builders' Library: Garantire la sicurezza del rollback durante le distribuzioni](#)
- [Amazon Builders' Library: Più velocità con una consegna continua](#)
- [What is AWS CodePipeline?](#)
- [What is CodeDeploy?](#)
- [AWS Systems Manager Patch Manager](#)
- [What is Amazon SES?](#)
- [What is Amazon Simple Notification Service?](#)

Video correlati:

- [AWS Summit 2019: CI/CD on AWS](#)

Gestione degli errori

 Con il passare del tempo, non sono da escludere eventuali guasti: dai router ai dischi rigidi, dai sistemi operativi alle unità di memoria che danneggiano i pacchetti TCP, nonché errori di natura temporanea o permanente. Questo è un dato di fatto, indipendentemente dal fatto che venga utilizzando hardware di alta qualità o componenti a basso costo - [Werner Vogels, CTO - Amazon.com](#)

I guasti dei componenti hardware di basso livello vengono risolti ogni giorno in un data center in locale. Nel cloud, tuttavia, devi essere protetto dalla maggior parte di questi tipi di guasti. Ad esempio, i volumi Amazon EBS sono collocati in una zona di disponibilità specifica in cui vengono automaticamente replicati per proteggerti dai guasti di un singolo componente. Tutti i volumi EBS sono progettati per garantire il 99,999% della disponibilità. Gli oggetti Amazon S3 vengono archiviati in almeno tre zone di disponibilità e garantiscono un'affidabilità pari al 99,999999999% per quanto riguarda la resistenza degli oggetti in un determinato anno. Indipendentemente dal provider di servizi cloud, è possibile che si verifichino guasti che influiscono sul tuo carico di lavoro. Pertanto, è necessario adottare misure per implementare la resilienza se è necessario che il tuo carico di lavoro sia affidabile.

Un prerequisito per l'applicazione delle linee guida qui discusse è la necessità di accertarsi che le persone incaricate della progettazione, dell'implementazione e della gestione dei tuoi carichi di lavoro, siano consapevoli degli obiettivi aziendali e dei requisiti per raggiungerli. Queste persone devono essere informate e addestrate per questi requisiti di affidabilità.

Nelle sezioni seguenti vengono illustrate le best practice per la gestione dei guasti, così da evitarne l'impatto sul tuo carico di lavoro.

Argomenti

- [Eseguire il backup dei dati](#)
- [Utilizzo dell'isolamento dei guasti per proteggere il carico di lavoro](#)
- [Progettazione di un carico di lavoro resistente agli errori dei componenti](#)
- [Test dell'affidabilità](#)
- [Pianificazione per il disaster recovery \(DR\)](#)

Eseguire il backup dei dati

Esegui il backup dei dati, delle applicazioni e della configurazione per soddisfare i requisiti relativi agli obiettivi del tempo di ripristino (RTO) e agli obiettivi del punto di ripristino (RPO).

Best practice

- [REL09-BP01 Identificazione e backup di tutti i dati che richiedono un backup o riproduzione dei dati dalle origini](#)
- [REL09-BP02 Protezione e crittografia dei backup](#)
- [REL09-BP03 Esecuzione del backup dei dati in automatico](#)
- [REL09-BP04 Ripristino periodico dei dati per verificare l'integrità e i processi di backup:](#)

REL09-BP01 Identificazione e backup di tutti i dati che richiedono un backup o riproduzione dei dati dalle origini

Scopri e usa le funzionalità di backup dei servizi per i dati e delle risorse utilizzati dal carico di lavoro. La maggior parte dei servizi offre funzionalità per eseguire il backup dei dati del carico di lavoro.

Risultato desiderato: capacità di identificare e classificare le origini dati in base alla criticità. Quindi, stabilisci una strategia per il recupero dei dati in base all'RPO. Questa strategia prevede il backup di queste origini dei dati o la possibilità di riprodurre i dati da altre origini. In caso di perdita di dati, la strategia implementata consente il recupero o la riproduzione dei dati entro i termini RPO e RTO definiti.

Fase di maturità del cloud: di base

Anti-pattern comuni:

- Mancata conoscenza di tutte le origini dei dati per il carico di lavoro e della loro criticità.
- Non si eseguono backup delle origini dei dati critiche.
- Esecuzione di backup solo di alcune origini dei dati senza utilizzare la criticità come criterio.
- Non esiste un RPO definito o la frequenza di backup non può soddisfare l'RPO.
- Nessuna valutazione della necessità di un backup o della possibilità di riprodurre i dati da altre origini.

Vantaggi dell'adozione di questa best practice: l'identificazione dei punti in cui sono necessari backup e l'implementazione di un meccanismo per la creazione di backup, o la riproduzione dei dati da un'origine esterna, migliorano la capacità di ripristinare e recuperare dati durante un'interruzione.

Livello di rischio associato alla mancata adozione di questa best practice: elevato

Guida all'implementazione

Tutti i data store AWS offrono funzionalità di backup. Servizi come Amazon RDS e Amazon DynamoDB supportano inoltre il backup automatico che consente il ripristino point-in-time (PITR), grazie al quale è possibile ripristinare un backup in qualsiasi momento fino a cinque minuti o meno rispetto all'ora corrente. Molti servizi AWS permettono di copiare backup in un'altra Regione AWS. AWS Backup è uno strumento che permette di centralizzare e automatizzare la protezione dei dati tra vari servizi AWS. [AWS Elastic Disaster Recovery](#) permette di copiare carichi di lavoro server completi e mantenere una protezione continua dei dati on-premise, tra zone di disponibilità o tra regioni con un obiettivo del punto di ripristino (RPO) misurato in secondi.

Amazon S3 può essere utilizzato come destinazione di backup per le origini dei dati gestite dal cliente e gestite da AWS. I servizi AWS come Amazon EBS, Amazon RDS e Amazon DynamoDB hanno funzionalità incorporate per creare i backup. È anche possibile utilizzare software di backup di terze parti.

È possibile eseguire il backup di dati on-premise nel Cloud AWS usando [AWS Storage Gateway](#) o [AWS DataSync](#). È possibile usare bucket Amazon S3 per archiviare questi dati in AWS. Amazon S3 offre più livelli di archiviazione, come [Amazon S3 Glacier o Deep Archive S3 Glacier](#), per ridurre i costi di archiviazione dei dati.

Potresti essere in grado di soddisfare le esigenze di recupero dei dati riproducendo i dati da altre origini. Ad esempio, potresti usare [nodi di replica Amazon ElastiCache](#) o [repliche di lettura Amazon RDS](#) per riprodurre i dati in caso di perdita del nodo primario. Nei casi in cui origini come questa possono essere usate per soddisfare l'[obiettivo del punto di ripristino \(RPO\) e l'obiettivo del tempo di ripristino \(RTO\)](#), un backup può non essere necessario. Come esempio aggiuntivo, se usi Amazon EMR, il backup del datastore HDFS può non essere necessario, purché sia possibile [riprodurre i dati in Amazon EMR da Amazon S3](#).

Quando scegli una strategia di backup, devi considerare il tempo necessario per il ripristino dei dati. Il tempo necessario per il ripristino dei dati dipende dal tipo di backup (nel caso di una strategia di backup) o dalla complessità del meccanismo di riproduzione dei dati. Questo tempo deve rientrare nell'RTO per il carico di lavoro.

Passaggi dell'implementazione

1. Identifica tutte le origini dati per il carico di lavoro. I dati possono essere archiviati in diverse risorse, come [database](#), [volumi](#), [file system](#), [sistemi di registrazione](#) e [risorse di archiviazione di oggetti](#). Consulta la sezione Risorse per trovare i documenti correlati su diversi servizi AWS che offrono l'archiviazione di dati e sulle funzionalità offerte da questi servizi.
2. Classifica le origini dati in base alla criticità. I diversi set di dati avranno diversi livelli di criticità per un carico di lavoro e quindi diversi requisiti di resilienza. Ad esempio, alcuni dati possono essere critici e richiedere un RPO prossimo allo zero, mentre altri dati possono essere meno critici e tollerare un RPO più elevato e una certa perdita di dati. Allo stesso modo, anche i diversi set di dati possono avere requisiti RTO diversi.
3. Usa AWS o servizi di terze parti per creare backup dei dati. [AWS Backup](#) è un servizio gestito che permette la creazione di backup di origini dati diverse in AWS. [AWS Elastic Disaster Recovery](#) gestisce la replica automatica dei dati in una Regione AWS con tempi inferiori al secondo. La maggior parte dei servizi AWS include anche funzionalità native per la creazione di backup. Marketplace AWS ha molte soluzioni che offrono anche queste funzionalità. Consulta la sezione Risorse di seguito per informazioni su come creare backup di dati da diversi servizi AWS.
4. Per i dati non sottoposti a backup, definisci un meccanismo di riproduzione dei dati. Puoi decidere di non eseguire il backup di dati riproducibili da altre origini per vari motivi. Potrebbe essere più conveniente riprodurre i dati dalle origini, quando necessario, piuttosto che creare un backup, dato che l'archiviazione dei backup può comportare dei costi. Un altro esempio è quello in cui il ripristino da un backup richiede più tempo rispetto alla riproduzione dei dati dalle origini, con conseguente violazione dell'RTO. In queste situazioni, è necessario considerare i compromessi e stabilire un processo ben definito per la riproduzione dei dati da queste origini quando è necessario il ripristino dei dati. Ad esempio, se hai caricato dati da Amazon S3 su un data warehouse (come Amazon Redshift) o su un cluster MapReduce (come Amazon EMR) per compiere analisi, ottieni un esempio pratico di riproduzione dati da oltre origini. Finché i risultati di queste analisi vengono archiviati o sono riproducibili, non subirai una perdita di dati a causa di un guasto nel data warehouse o nel cluster MapReduce. Altri esempi che possono essere riprodotti dalle origini includono le cache (ad esempio Amazon ElastiCache) o le repliche di lettura RDS.
5. Definisci una cadenza per il backup dei dati. La creazione di backup delle origini dei dati è un processo periodico e la frequenza deve dipendere dall'RPO.

Livello di impegno per il piano di implementazione: moderato.

Risorse

Best practice correlate:

[REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati](#)

[REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino](#)

Documenti correlati:

- [Che cos'è AWS Backup?](#)
- [Che cos'è AWS DataSync?](#)
- [What is Volume Gateway? \(Che cos'è il Gateway di volumi?\)](#)
- [Partner APN: partner per il backup](#)
- [Marketplace AWS: prodotti che possono essere usati per il backup](#)
- [Snapshot Amazon EBS](#)
- [Backup in Amazon EFS](#)
- [Backup in Amazon FSx per Windows File Server](#)
- [Backup e ripristino di ElastiCache for Redis](#)
- [Creazione di shapshot di cluster di database in Neptune](#)
- [Creazione di uno snapshot DB](#)
- [Creazione di una regola EventBridge attivata in base a una pianificazione](#)
- [Replica tra regioni con Amazon S3](#)
- [AWS Backup da EFS a EFS](#)
- [Esportazione di dati di log in Amazon S3](#)
- [Gestione del ciclo di vita dell'applicazione](#)
- [Backup e ripristino on demand per DynamoDB](#)
- [Ripristino point-in-time \(PITR\) per DynamoDB](#)
- [Uso di snapshot di indici Amazon OpenSearch Service](#)
- [Che cos'è AWS Elastic Disaster Recovery?](#)

Video correlati:

- [AWS re:Invent 2021: Backup, ripristino di emergenza e protezione da ransomware con AWS](#)
- [Demo su AWS Backup: Backup tra account e tra regioni](#)

- [AWS re:Invent 2019: Approfondimento su AWS Backup, con Rackspace \(STG341\)](#)

Esempi correlati:

- [Well-Architected Lab: Implementazione della replica bidirezionale tra regioni per Amazon S3](#)
- [Well-Architected Lab: Esecuzione di test del backup e del ripristino di dati](#)
- [Well-Architected Lab: Backup e ripristino con failback per un carico di lavoro di analisi](#)
- [Well-Architected Lab: Ripristino di emergenza – Backup e ripristino](#)

REL09-BP02 Protezione e crittografia dei backup

Controlla e rileva l'accesso ai backup tramite l'autenticazione e l'autorizzazione. Previene e rileva se l'integrità dei dati dei backup è compromessa utilizzando la crittografia.

Anti-pattern comuni:

- Disporre di un accesso identico sia per i backup e l'automazione del ripristino sia per i dati.
- Non codificare i backup.

Vantaggi dell'adozione di questa best practice: la protezione dei backup previene la manomissione dei dati, mentre la crittografia dei dati impedisce l'accesso ai dati se questi vengono accidentalmente esposti.

Livello di rischio associato alla mancata adozione di questa best practice: elevato

Guida all'implementazione

Controlla e rileva l'accesso ai backup tramite l'autenticazione e l'autorizzazione, ad esempio con AWS Identity and Access Management (IAM). Previene e rileva se l'integrità dei dati dei backup è compromessa utilizzando la crittografia.

Amazon S3 supporta diversi metodi di crittografia dei dati archiviati. Utilizzando la crittografia lato server, Amazon S3 accetta anche dati non crittografati e li crittografa man mano che vengono memorizzati. Utilizzando la crittografia lato client, l'applicazione del carico di lavoro è responsabile della crittografia dei dati prima che vengano inviati a Amazon S3. Entrambi i metodi ti consentono di utilizzare AWS Key Management Service (AWS KMS) per creare ed archiviare la chiave di crittografia dei dati, oppure di utilizzarne una personalizzata (della quale sarai responsabile). Tramite AWS KMS

puoi impostare delle policy utilizzando IAM per regolare l'accesso alle chiavi dei dati, oltre che ai dati privi di crittografia.

Per Amazon RDS, se hai scelto di crittografare i database, anche i backup verranno crittografati. I backup di DynamoDB sono sempre crittografati. Quando usi AWS Elastic Disaster Recovery, vengono crittografati tutti i dati in transito e a riposo. Con Elastic Disaster Recovery i dati a riposo possono essere crittografati tramite la chiave di crittografia dei volumi della crittografia predefinita in Amazon EBS o una chiave gestita dal cliente personalizzata.

Passaggi dell'implementazione

1. Utilizzo della crittografia su ciascuno dei datastore. Se i dati di origine sono crittografati, lo sarà anche il backup.
 - [Usa la crittografia in Amazon RDS](#).. Puoi configurare la crittografia dei dati a riposo utilizzando AWS Key Management Service al momento della creazione di un'istanza RDS.
 - [Usa la crittografia su volumi Amazon EBS](#).. Puoi configurare la crittografia predefinita o specificare una chiave univoca al momento della creazione del volume.
 - Usa la [crittografia Amazon DynamoDB](#) necessaria. DynamoDB esegue la crittografia di tutti i dati a riposo. Puoi utilizzare una chiave AWS KMS di proprietà di AWS o una chiave KMS gestita da AWS specificando una chiave archiviata nel tuo account.
 - [Esegui la crittografia dei dati archiviati in Amazon EFS](#). Configura la crittografia al momento della creazione del file system.
 - Configura la crittografia nelle regioni di origine e di destinazione. Puoi configurare la crittografia dei dati a riposo in Amazon S3 utilizzando le chiavi archiviate in KMS, ma le chiavi sono specifiche per regione. Puoi specificare le chiavi di destinazione quando configuri la replica.
 - Scegli se usare la [crittografia Amazon EBS predefinita o personalizzata per Elastic Disaster Recovery](#). Questa opzione esegue la crittografia dei dati a riposo replicati nei dischi della sottorete dell'area di staging e nei dischi replicati.
2. Implementazione delle autorizzazioni con privilegi minimi per accedere ai backup. Segui le best practice per limitare l'accesso a backup, snapshot e repliche in linea con le [best practice per la sicurezza](#).

Risorse

Documenti correlati:

- [Marketplace AWS: prodotti che possono essere usati per il backup](#)

- [Crittografia in Amazon EBS](#)
- [Amazon S3: protezione dei dati tramite crittografia](#)
- [Configurazione aggiuntiva della replica tra regioni: replica di oggetti creati con la crittografia lato server \(SSE\) usando chiavi di crittografia archiviate in AWS KMS](#)
- [Crittografia dei dati a riposo in DynamoDB](#)
- [Crittografia di risorse Amazon RDS](#)
- [Crittografia di dati e metadati in Amazon EFS](#)
- [Crittografia per i backup in AWS](#)
- [Gestione di tabelle crittografate](#)
- [Principio della sicurezza: Framework AWS Well-Architected](#)
- [Che cos'è AWS Elastic Disaster Recovery?](#)

Esempi correlati:

- [Well-Architected Lab: Implementazione della replica bidirezionale tra regioni per Amazon S3](#)

REL09-BP03 Esecuzione del backup dei dati in automatico

Configura i backup in modo che vengano eseguiti automaticamente in base a una pianificazione periodica informata dall'Obiettivo del punto di ripristino (RPO) o dalle modifiche apportate al set di dati. I set di dati critici con bassi requisiti di perdita di dati devono essere sottoposti a backup automatico su base frequente, mentre i dati meno critici, per i quali è accettabile una certa perdita, possono essere sottoposti a backup meno frequenti.

Risultato desiderato: un processo di backup automatico che crea backup delle origini dati a una cadenza prestabilita.

Anti-pattern comuni:

- Eseguire i backup manualmente.
- Utilizzare risorse che dispongono di funzionalità di backup, ma non includere il backup nell'automazione.

Vantaggi dell'adozione di questa best practice: l'automazione dei backup verifica che i backup vengano eseguiti regolarmente in base all'RPO e invia avvisi in caso contrario.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

AWS Backup può essere utilizzato per creare backup automatici di varie origini dei dati AWS. Il backup delle istanze Amazon RDS può essere eseguito quasi ininterrottamente ogni cinque minuti e quello degli oggetti Amazon S3 quasi ininterrottamente ogni quindici minuti, consentendo il ripristino point-in-time (PITR) a un punto specifico della cronologia di backup. Per altre origini dei dati AWS, come volumi Amazon EBS, tabelle Amazon DynamoDB o file system Amazon FSx, AWS Backup può eseguire il backup automatico con una frequenza di un'ora. Questi servizi offrono anche funzionalità di backup native. I servizi AWS che offrono il backup automatico con ripristino point-in-time (PITR) includono [Amazon DynamoDB](#), [Amazon RDS](#) e [Amazon Keyspaces \(per Apache Cassandra\)](#). Questi servizi permettono il ripristino temporizzato in base a un momento specifico all'interno della cronologia dei backup. La maggior parte degli altri servizi di archiviazione di dati AWS offre la possibilità di programmare backup periodici, anche ogni ora.

Amazon RDS e Amazon DynamoDB offrono il backup continuo con ripristino point-in-time (PITR). Una volta abilitato, il controllo delle versioni in Amazon S3 è automatico. Puoi usare [Amazon Data Lifecycle Manager](#) per automatizzare la creazione, la copia e l'eliminazione di snapshot Amazon EBS. Può anche automatizzare la creazione, la copia, la rimozione e la cancellazione di Amazon Machine Images (AMI) con backup Amazon EBS e dei relativi snapshot Amazon EBS sottostanti.

AWS Elastic Disaster Recovery offre la replica a livello di blocco continua dall'ambiente di origine (on-premise o AWS) alla regione di ripristino di destinazione. Gli snapshot Amazon EBS point-in-time vengono creati e gestiti automaticamente dal servizio.

Per una visualizzazione centralizzata dell'automazione e della cronologia dei backup, AWS Backup fornisce una soluzione di backup completamente gestita basata su policy. Centralizza e automatizza il backup dei dati su più servizi AWS nel cloud e on-premise utilizzando AWS Storage Gateway.

Oltre a quella di controllo delle versioni, Amazon S3 offre tutte le funzioni di replica. L'intero bucket S3 può essere replicato automaticamente in un altro bucket in una Regione AWS diversa.

Passaggi dell'implementazione

1. Identifica le origini dati di cui al momento viene eseguito manualmente il backup. Per ulteriori informazioni, consulta [REL09-BP01 Identificazione e backup di tutti i dati che richiedono un backup o riproduzione dei dati dalle origini](#).
2. Determina l'RPO per il carico di lavoro. Per ulteriori informazioni, consulta [REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati](#).

3. Usa una soluzione di backup automatica o un servizio gestito. AWS Backup è un servizio completamente gestito che semplifica la [centralizzazione e l'automazione della protezione dei dati tra diversi servizi AWS, nel cloud e on-premise](#). Usando piani di backup in AWS Backup, crea regole che definiscano le risorse di cui eseguire il backup e la frequenza di creazione dei backup. Questa frequenza deve essere informata dall'RPO stabilito al punto 2. Per linee guida pratiche su come creare backup automatici con AWS Backup, consulta [Well-Architected Lab: Test del backup e del ripristino di dati](#). La maggior parte dei servizi AWS di archiviazione dei dati offre funzionalità di backup native. Ad esempio, RDS può essere sfruttato per backup automatici con ripristino point-in-time (PITR).
4. Per le origini dati non supportate da una soluzione di backup automatica o da un servizio gestito, come le code di messaggi o le origini dati on-premise, valuta se usare una soluzione di terze parti affidabile per creare backup automatici. In alternativa, puoi creare un'automazione utilizzando la AWS CLI o gli SDK. Puoi usare funzioni AWS Lambda o AWS Step Functions per definire la logica necessaria per la creazione di un backup di dati e Amazon EventBridge per eseguirla in base a una frequenza determinata dall'RPO.

Livello di impegno per il piano di implementazione: basso.

Risorse

Documenti correlati:

- [Partner APN: partner per il backup](#)
- [Marketplace AWS: prodotti che possono essere usati per il backup](#)
- [Creazione di una regola EventBridge attivata in base a una pianificazione](#)
- [Che cos'è AWS Backup?](#)
- [Che cos'è AWS Step Functions?](#)
- [Che cos'è AWS Elastic Disaster Recovery?](#)

Video correlati:

- [AWS re:Invent 2019: Approfondimento su AWS Backup, con Rackspace \(STG341\)](#)

Esempi correlati:

- [Well-Architected Lab: Esecuzione di test del backup e del ripristino di dati](#)

REL09-BP04 Ripristino periodico dei dati per verificare l'integrità e i processi di backup:

Verifica che l'implementazione del processo di backup soddisfi gli obiettivi del tempo di ripristino (RTO) e gli obiettivi del punto di ripristino (RPO) eseguendo un test del ripristino.

Risultato desiderato: ripristino periodico dei dati dai backup tramite meccanismi ben definiti per garantire che il ripristino sia possibile entro l'obiettivo del tempo di ripristino (RTO) stabilito per il carico di lavoro. Verifica che il ripristino da un backup porti a una risorsa che contiene i dati originali senza che questi siano danneggiati o inaccessibili e con una perdita di dati entro l'Obiettivo del punto di ripristino (RPO).

Anti-pattern comuni:

- Ripristino di un backup, ma senza eseguire query sui dati o recuperarli per verificare di poter usare il ripristino.
- Presupporre l'esistenza di un backup.
- Presupporre che il backup di un sistema sia pienamente operativo e che i dati possano essere recuperati da esso.
- Presupporre che il tempo di ripristino o di recupero dei dati da un backup rientri nell'RTO del carico di lavoro.
- Presupporre che i dati contenuti nel backup rientrino nell'RPO del carico di lavoro.
- Ripristino in base alle esigenze, senza usare un runbook o seguire una procedura automatica prestabilita.

Vantaggi dell'adozione di questa best practice: i test del ripristino dei backup permettono di verificare che i dati possano essere ripristinati senza timore che siano mancanti o danneggiati, che il ripristino e il recupero siano possibili in base all'RTO per il carico di lavoro e che un'eventuale perdita di dati rientri nell'RPO per il carico di lavoro.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

La verifica delle capacità di backup e ripristino aumenta la fiducia nella capacità di eseguire queste azioni durante un'interruzione. Ripristina periodicamente i backup in una nuova posizione ed esegui test per verificare l'integrità dei dati. Alcuni test comuni che devono essere eseguiti sono la verifica

che tutti i dati siano disponibili, non siano danneggiati e siano accessibili e che un'eventuale perdita di dati rientri nell'RPO per il carico di lavoro. Questi test possono anche aiutare a verificare se i meccanismi di ripristino sono sufficientemente veloci per soddisfare l'RTO del carico di lavoro.

Con AWS, puoi creare un ambiente di test e ripristinare i backup per valutare le funzionalità RTO e RPO ed eseguire test sul contenuto e l'integrità dei dati.

Inoltre, Amazon RDS e Amazon DynamoDB consentono il ripristino point-in-time (PITR). Utilizzando il backup continuo, puoi ripristinare il set di dati allo stato in cui si trovava in una data e un'ora specificate.

tutti i dati siano disponibili, non siano danneggiati, siano accessibili e che qualsiasi perdita di dati rientri nell'RPO del carico di lavoro. Questi test possono anche aiutare a verificare se i meccanismi di ripristino sono sufficientemente veloci per soddisfare l'RTO del carico di lavoro.

AWS Elastic Disaster Recovery offre snapshot di ripristino point-in-time (RPIT) continui di volumi Amazon EBS. Durante la replica dei server di origine, vengono registrati gli stati point-in-time nel corso del tempo in base alla policy configurata. Elastic Disaster Recovery permette di verificare l'integrità di questi snapshot avviando istanze per scopi di test ed esercitazione senza reindirizzare il traffico.

Passaggi dell'implementazione

1. Identifica le origini dati di cui viene attualmente eseguito il backup e le posizioni in cui vengono archiviati i backup. Per le linee guida di implementazione, consulta [REL09-BP01 Identificazione e backup di tutti i dati che richiedono un backup o riproduzione dei dati dalle origini](#).
2. Definisci i criteri per la convalida dei dati per ogni origine dati. Tipi di dati differenti avranno proprietà diverse che potrebbero richiedere meccanismi di convalida diversi. Considera il modo in cui potrebbero essere convalidati questi dati prima di poterli utilizzare in produzione. Alcuni modi comuni per convalidare i dati sono l'uso delle loro proprietà dei dati e del backup, come il tipo di dati, il formato, la somma di controllo, la dimensione o la combinazione di questi elementi con una logica di convalida personalizzata. Ad esempio, può trattarsi di un confronto dei valori di checksum tra la risorsa ripristinata e l'origine dei dati al momento della creazione del backup.
3. Definisci l'RTO e l'RPO per il ripristino dei dati in base alle criticità dei dati. Per le linee guida di implementazione, consulta [REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati](#).
4. Valuta la capacità di ripristino. Rivedi la strategia di backup e ripristino per capire se è in grado di soddisfare RTO e RPO e modifica la strategia se necessario. Usando la [Centrale di resilienza](#)

- [AWS](#), puoi eseguire una valutazione del carico di lavoro. La valutazione esamina la configurazione dell'applicazione rispetto alle policy sulla resilienza e indica se gli obiettivi RTO e RPO possono essere raggiunti.
5. Esegui un ripristino di test usando i processi attualmente definiti nell'ambiente di produzione per il ripristino dei dati. Questi processi dipendono dal modo in cui è stato eseguito il backup dell'origine dei dati iniziale, dal formato e dalla posizione di archiviazione del backup stesso o dalla riproduzione dei dati da altre fonti. Ad esempio, se usi un servizio gestito come [AWS Backup](#), [può trattarsi di un semplice ripristino del backup in una nuova risorsa](#). Se hai usato AWS Elastic Disaster Recovery, puoi [avviare un'esercitazione di ripristino](#).
 6. Convalida il ripristino dei dati dalla risorsa ripristinata in base ai criteri definiti in precedenza per la convalida dei dati. I dati ripristinati e recuperati contengono il record o la voce più recente al momento del backup? Questi dati rientrano nell'RPO per il carico di lavoro?
 7. Misura il tempo necessario per il ripristino e il recupero e confrontalo con l'RTO definito. Questo tempo deve rientrare nell'RTO per il carico di lavoro? Ad esempio, confronta i timestamp dell'inizio del processo di ripristino e del completamento della convalida del ripristino per calcolare la durata del processo. Tutte le chiamate API AWS includono un timestamp e queste informazioni sono disponibili in [AWS CloudTrail](#). Sebbene queste informazioni possano fornire dettagli sull'inizio del processo di ripristino, la logica di convalida dovrebbe registrare il timestamp finale del completamento della convalida. Se usi un processo automatico, servizi come [Amazon DynamoDB](#) possono essere utili per archiviare queste informazioni. Inoltre, molti servizi AWS offrono una cronologia degli eventi che fornisce informazioni con data e ora in cui si sono verificate determinate azioni. In AWS Backup le attività di backup e ripristino sono note come processi e tali processi possono contenere informazioni sul timestamp come parte dei metadati, che possono essere usate per misurare il tempo necessario per il ripristino e il recupero.
 8. Comunica agli stakeholder se la convalida dei dati non riesce o se il tempo necessario per il ripristino e il recupero supera l'RTO definito per il carico di lavoro. Nell'implementare l'automazione a questo scopo, [come in questo lab](#), puoi usare servizi come Amazon Simple Notification Service (Amazon SNS) per inviare notifiche push come e-mail o SMS agli stakeholder. [Questi messaggi possono anche essere pubblicati in applicazioni di messaggistica come Amazon Chime, Slack o Microsoft Teams](#) o usati per [creare attività come OpsItem usando OpsCenter di AWS Systems Manager](#).
 9. Automatizza questo processo in modo che venga eseguito periodicamente. Ad esempio, per automatizzare i processi di ripristino e recupero si possono utilizzare servizi come AWS Lambda o una State Machine in AWS Step Functions, mentre Amazon EventBridge può essere utilizzato per attivare periodicamente questo flusso di lavoro di automazione, come mostrato nel diagramma di

architettura sottostante. Per altre informazioni, consulta [Automazione della convalida del ripristino di dati con AWS Backup](#). Inoltre, [questo Well-Architected Lab](#) permette di acquisire esperienza pratica su un modo per implementare l'automazione per diverse fasi presentate qui.

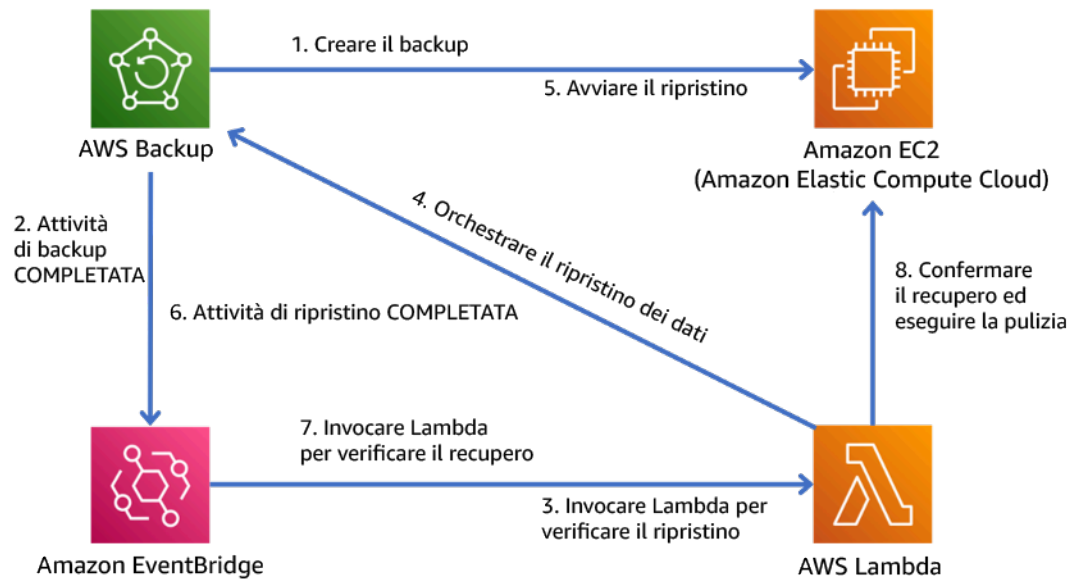


Figura 9. Processo di backup e ripristino automatico

Livello di impegno per il piano di implementazione: da moderato a elevato, a seconda della complessità dei criteri di convalida.

Risorse

Documenti correlati:

- [Automazione della convalida del ripristino di dati con AWS Backup](#)
- [Partner APN: partner per il backup](#)
- [Marketplace AWS: prodotti che possono essere usati per il backup](#)
- [Creazione di una regola EventBridge attivata in base a una pianificazione](#)
- [Backup e ripristino on demand per DynamoDB](#)
- [Che cos'è AWS Backup?](#)
- [Che cos'è AWS Step Functions?](#)
- [Che cos'è AWS Elastic Disaster Recovery?](#)
- [AWS Elastic Disaster Recovery](#)

Esempi correlati:

- [Well-Architected Lab: Esecuzione di test del backup e del ripristino di dati](#)

Utilizzo dell'isolamento dei guasti per proteggere il carico di lavoro

Le barriere per l'isolamento dei guasti limitano l'effetto di un errore all'interno di un carico di lavoro a un numero limitato di componenti. I componenti al di fuori della barriera non subiscono gli effetti del guasto. Utilizzando più barriere per l'isolamento dei guasti, puoi limitare l'impatto sul carico di lavoro.

Best practice

- [REL10-BP01 Implementazione del carico di lavoro in diversi luoghi](#)
- [REL10-BP02 Selezione delle posizioni appropriate per la tua implementazione multiposizione](#)
- [REL10-BP03 Ripristino automatico dei componenti vincolati a una singola posizione](#)
- [REL10-BP04 Utilizzo di architetture a scomparti per limitare la portata dell'impatto](#)

REL10-BP01 Implementazione del carico di lavoro in diversi luoghi

Distribuisci i dati e le risorse del carico di lavoro su più zone di disponibilità o, se necessario, su diverse Regioni AWS. Questi luoghi possono essere diversi a seconda delle necessità.

Uno dei principi fondamentali per la progettazione di servizi su AWS è l'eliminazione di singoli punti di errore nell'infrastruttura fisica sottostante. Questo ci spinge a creare software e sistemi che utilizzano più zone di disponibilità e sono resistenti al fallimento di una singola zona. Allo stesso modo, i sistemi sono costruiti per resistere ai guasti di un singolo nodo di calcolo, singolo volume di archiviazione o singola istanza di un database. Quando si costruisce un sistema che si basa su componenti ridondanti, è importante garantire che i componenti funzionino in modo indipendente e, nel caso delle Regioni AWS, in modo autonomo. I vantaggi ottenuti dai calcoli di disponibilità teorica con componenti ridondanti sono validi solo se questo continua a essere vero.

Zone di disponibilità (AZ)

Le Regioni AWS sono composte da almeno due zone di disponibilità progettate per essere indipendenti. Ogni zona di disponibilità è separata da una distanza fisica significativa da altre zone per evitare scenari di guasto correlati, dovuti a rischi ambientali come incendi, inondazioni e tornado. Ogni zona di disponibilità ha anche un'infrastruttura fisica indipendente: connessioni dedicate di alimentazione di rete, fonti di alimentazione di backup autonome, servizi meccanici

indipendenti e connettività di rete indipendente all'interno e all'esterno della zona di disponibilità. Questa struttura limita gli errori di uno qualsiasi di questi sistemi alla sola AZ interessata. Nonostante siano geograficamente separate, le zone di disponibilità sono situate nella stessa area regionale, il che consente una rete a velocità di trasmissione effettiva elevata e bassa latenza. L'intera Regione AWS (in tutte le zone di disponibilità, costituite da più data center fisicamente indipendenti) può essere trattata come un unico obiettivo logico di implementazione per il carico di lavoro, compresa la possibilità di replicare i dati in modo sincrono (ad esempio, tra i database). Ciò ti consente di utilizzare le zone di disponibilità in una configurazione attiva/attiva o attiva/standby.

Le zone di disponibilità sono indipendenti e pertanto la disponibilità del carico di lavoro aumenta quando il carico di lavoro è progettato per utilizzare più zone di disponibilità. Alcuni servizi AWS (tra cui il piano dati dell'istanza Amazon EC2) sono implementati come servizi strettamente zonali nei quali hanno un destino condiviso con la zona di disponibilità in cui si trovano. Le istanze Amazon EC2 nelle altre AZ non saranno, tuttavia, interessate e continueranno a funzionare. Allo stesso modo, se un errore in una zona di disponibilità causa l'errore di un database Amazon Aurora, un'istanza Aurora di lettura-replica in una AZ non interessata può essere automaticamente promossa a primaria. I servizi regionali AWS, ad esempio Amazon DynamoDB, utilizzano internamente più zone di disponibilità in una configurazione attiva/attiva per raggiungere gli obiettivi di progettazione della disponibilità per quel servizio, senza che sia necessario configurare il posizionamento delle AZ.

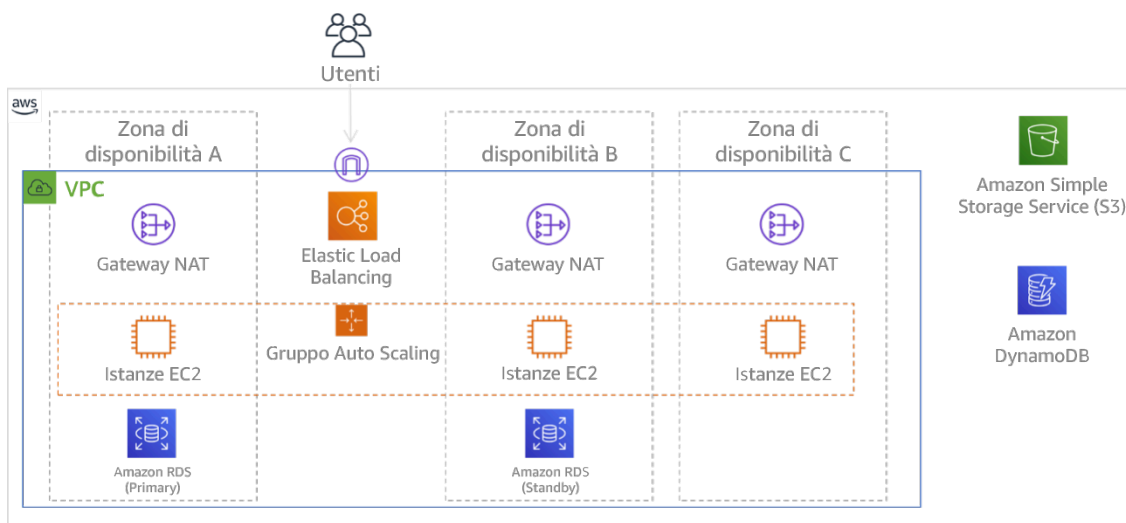


Figura 9. Architettura multi-livello distribuita su tre zone di disponibilità. Tieni presente che Amazon S3 e Amazon DynamoDB sono sempre Multi-AZ automaticamente. L'ELB viene inoltre distribuito in tutte e tre le zone.

Mentre i piani di controllo AWS in genere offrono la possibilità di gestire le risorse all'interno dell'intera Regione (più zone di disponibilità), alcuni piani di controllo (inclusi Amazon EC2 ed Amazon EBS)

hanno la capacità di filtrare i risultati per una singola zona di disponibilità. Con questo approccio, la richiesta viene elaborata solo nella zona di disponibilità specificata, riducendo l'esposizione all'interruzione in altre zone di disponibilità. Questo esempio di AWS CLI illustra come ottenere informazioni su un'istanza Amazon EC2 dalla sola zona di disponibilità us-east-2c:

```
AWS ec2 describe-instances --filters Name=availability-zone,Values=us-east-2c
```

Zone locali AWS

Le Zone locali AWS agiscono in modo simile alle zone di disponibilità nella rispettiva Regione AWS, in quanto possono essere selezionate come ubicazione di posizionamento per le risorse AWS zonali come le sottoreti e le istanze EC2. Ciò che le rende speciali è che non si trovano nella Regione AWS associata, ma vicino a grandi popolazioni, settori e centri IT in cui al momento non esiste alcuna Regione AWS. Tuttavia, mantengono una connessione sicura e a larghezza di banda elevata tra i carichi di lavoro locali nella zona locale e quelli in esecuzione nella Regione AWS. È consigliabile utilizzare le Zone locali AWS per implementare i carichi di lavoro più vicini agli utenti per requisiti a bassa latenza.

Amazon Global Edge Network

Amazon Global Edge Network è costituito da posizioni edge in città di tutto il mondo. Amazon CloudFront utilizza questa rete per fornire contenuti agli utenti finali con una latenza inferiore. AWS Global Accelerator consente di creare gli endpoint del carico di lavoro in queste posizioni edge per fornire l'onboarding alla rete globale AWS vicino agli utenti. Amazon API Gateway permette agli endpoint API ottimizzati per l'edge che utilizzano una distribuzione CloudFront di facilitare l'accesso dei clienti attraverso la posizione edge più vicina.

Regioni AWS

Le Regioni AWS sono progettate per essere autonome; pertanto, per utilizzare un approccio multi-regione, puoi implementare copie dedicate dei servizi in ciascuna Regione.

Un approccio multi-regione è comune per le strategie di ripristino di emergenza per raggiungere gli obiettivi di ripristino quando si verificano eventi unici su larga scala. Consulta [Pianificazione per il disaster recovery \(DR\)](#) per ulteriori informazioni su queste strategie. Qui, tuttavia, si focalizza l'attenzione sulla disponibilità, che cerca di fornire un obiettivo medio di operatività nel tempo. Per gli obiettivi di alta disponibilità, un'architettura multi-regione sarà generalmente progettata per essere attiva/attiva, dove ogni copia del servizio (nelle rispettive Regioni) è attiva (serve le richieste).

Consiglio

Gli obiettivi di disponibilità per la maggior parte dei carichi di lavoro possono essere soddisfatti utilizzando una strategia multi-AZ all'interno di una singola Regione AWS. Considera le architetture multi-regione solo quando i carichi di lavoro hanno requisiti di disponibilità estremi o altri obiettivi aziendali che richiedono un'architettura multi-regione.

AWS offre ai clienti la possibilità di gestire servizi in più Regioni. Ad esempio, AWS fornisce una replica continua e asincrona dei dati utilizzando la replica Amazon Simple Storage Service (Amazon S3), le repliche di lettura Amazon RDS (incluse le repliche di lettura Aurora) e le tabelle globali Amazon DynamoDB. Con la replica continua, le versioni dei dati sono disponibili per un uso quasi immediato in ogni Regione attiva.

Utilizzando AWS CloudFormation, puoi definire l'infrastruttura e implementarla in modo coerente sugli Account AWS e sulle Regioni AWS. Invece, AWS CloudFormation StackSets estende questa funzionalità consentendo di creare, aggiornare o eliminare stack AWS CloudFormation su più account e regioni con un'unica operazione. Per le implementazioni di istanza Amazon EC2, si utilizza un'immagine AMI (Amazon Machine Image) per fornire informazioni quali la configurazione hardware e il software installato. È possibile implementare una pipeline di Amazon EC2 Image Builder che crea le AMI necessarie e le copia nelle regioni attive. Ciò garantisce che le Golden AMI abbiano tutto ciò che serve per implementare e dimensionare il carico di lavoro in ogni nuova regione.

Per instradare il traffico, sia Amazon Route 53 sia AWS Global Accelerator abilitano la definizione di criteri che determinano quali utenti indirizzare a ogni endpoint regionale attivo. Con Global Accelerator imposti un valore di traffico per controllare la percentuale di traffico diretta a ciascun endpoint dell'applicazione. Route 53 supporta questo approccio percentuale e anche diverse altre policy disponibili, tra cui quelle basate sulla geoprossimità e sulla latenza. Global Accelerator sfrutta automaticamente la vasta rete di server edge AWS per convogliare il traffico verso la dorsale di rete AWS il prima possibile, con conseguente riduzione delle latenze delle richieste.

Tutte queste capacità operano in modo da preservare l'autonomia di ogni Regione. Ci sono pochissime eccezioni a questo approccio, inclusi i nostri servizi che forniscono distribuzione edge globale (ad esempio Amazon CloudFront e Amazon Route 53), insieme al piano di controllo per il servizio AWS Identity and Access Management (IAM). La maggior parte dei servizi opera interamente all'interno di una singola Regione.

Data center in locale

Per i carichi di lavoro eseguiti in un data center on-premise, puoi progettare un'esperienza ibrida quando possibile. AWS Direct Connect fornisce una connessione di rete dedicata dalla tua sede ad AWS che consente l'esecuzione in entrambi.

Un'altra opzione è quella di eseguire l'infrastruttura AWS e i servizi on-premise utilizzando AWS Outposts. AWS Outposts è un servizio completamente gestito che estende l'infrastruttura AWS, i servizi AWS, le API e gli strumenti al tuo data center. La stessa infrastruttura hardware utilizzata nel Cloud AWS viene installata nel data center. AWS Outposts è, quindi, connesso alla Regione AWS più vicina. Puoi quindi utilizzare AWS Outposts per supportare i carichi di lavoro che hanno requisiti di bassa latenza o di elaborazione dei dati locali.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

- Utilizza zone di disponibilità multiple e Regioni AWS. Distribuisci i dati e le risorse del carico di lavoro su più zone di disponibilità o, se necessario, su diverse Regioni AWS. Questi luoghi possono essere diversi a seconda delle necessità.
 - I servizi regionali sono distribuiti intrinsecamente in zone di disponibilità.
 - Sono inclusi Amazon S3, Amazon DynamoDB e AWS Lambda (se non collegati a un VPC)
 - Distribuisci il tuo container, istanza e carichi di lavoro basati su funzioni in più zone di disponibilità. Utilizza datastore multi-zona, inclusi sistemi di cache. Utilizza le funzionalità di dimensionamento automatico EC2, posizionamento di attività ECS, configurazione della funzione AWS Lambda in esecuzione nel tuo VPC e i cluster ElastiCache.
 - Utilizza sottoreti che sono in zone di disponibilità separate nella distribuzione di gruppi Auto Scaling.
 - [Esempio: distribuzione di istanze in più zone di disponibilità](#)
 - [Strategie di posizionamento dei processi di Amazon ECS](#)
 - [Configurazione di una funzione AWS Lambda per accedere alle risorse in un Amazon VPC](#)
 - [Scelta di regioni e zone di disponibilità](#)
 - Utilizza sottoreti in zone di disponibilità separate quando distribuisci gruppi Auto Scaling.
 - [Esempio: distribuzione di istanze in più zone di disponibilità](#)
 - Utilizza parametri di posizionamento attività ECS, specificando i gruppi di sottorete DB.
 - [Strategie di posizionamento dei processi di Amazon ECS](#)

- Utilizza sottoreti in più zone di disponibilità quando configuri una funzione da eseguire nel tuo VPC.
 - [Configurazione di una funzione AWS Lambda per accedere alle risorse in un Amazon VPC](#)
- Utilizza più zone di disponibilità con cluster ElastiCache.
 - [Scelta di regioni e zone di disponibilità](#)
- Se il carico di lavoro deve essere implementato in più Regioni, scegli una strategia multi-regione. La maggior parte delle esigenze di affidabilità può essere soddisfatta all'interno di una singola Regione AWS utilizzando una strategia a più zone di disponibilità. Quando necessario, utilizza una strategia multi-Regione per soddisfare le tue esigenze aziendali.
 - [AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications \(Modelli architetturali per applicazioni attive-attive su più Regioni\) \(ARC209-R2\)](#)
 - Il backup in un'altra Regione AWS può garantire ulteriormente che i dati saranno disponibili quando necessario.
 - Alcuni carichi di lavoro hanno requisiti normativi che prevedono l'utilizzo di una strategia multi-regione.
- Valuta AWS Outposts per il tuo carico di lavoro. Se il carico di lavoro richiede bassa latenza nel data center locale o ha requisiti di elaborazione dei dati locali. In tal caso esegui l'infrastruttura e i servizi AWS on-premise utilizzando AWS Outposts.
 - [Che cos'è AWS Outposts?](#)
- Stabilisci se le Zone locali AWS ti aiutano a fornire il servizio ai tuoi utenti. Se hai requisiti di bassa latenza, verifica se le Zone locali AWS si trovano vicino ai tuoi utenti. Se sì, utilizzale per implementare carichi di lavoro più vicini a tali utenti.
 - [Domande frequenti sulle Zone locali AWS](#)

Risorse

Documenti correlati:

- [Infrastruttura globale di AWS](#)
- [Domande frequenti sulle Zone locali AWS](#)
- [Strategie di posizionamento dei processi di Amazon ECS](#)
- [Scelta di regioni e zone di disponibilità](#)
- [Esempio: distribuzione di istanze in più zone di disponibilità](#)

- [Tabelle globali: replica multi-regione con DynamoDB](#)
- [Using Amazon Aurora global databases \(Utilizzo di database Amazon Aurora globali\)](#)
- [Creating a Multi-Region Application with AWS Services blog series \(Creazione di un'applicazione multi-regione con la serie di blog sui servizi AWS\)](#)
- [Che cos'è AWS Outposts?](#)

Video correlati:

- [AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications \(Modelli architetturali per applicazioni attive-attive su più Regioni\) \(ARC209-R2\)](#)
- [AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure \(Innovazione e gestione dell'infrastruttura di rete globale AWS\) \(NET339\)](#)

REL10-BP02 Selezione delle posizioni appropriate per la tua implementazione multiposizione

Risultato desiderato

Per ottenere un'elevata disponibilità, distribuisce sempre (quando possibile) i componenti del carico di lavoro in più zone di disponibilità (AZ), come illustrato nella Figura 10. Per i carichi di lavoro con requisiti di resilienza estremi, valuta attentamente le opzioni per un'architettura multiregione.

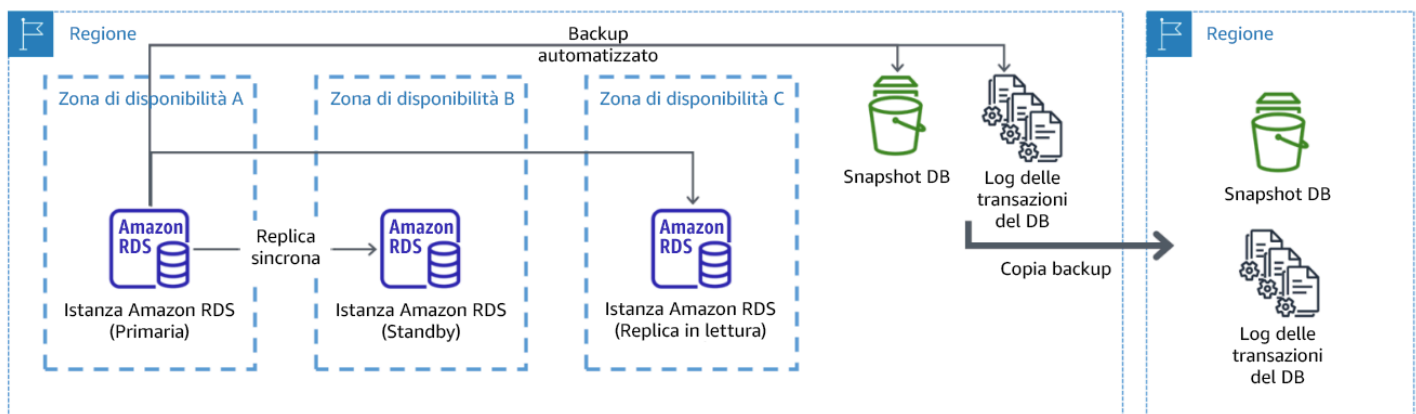


Figura 10: Distribuzione resiliente di un database multi-AZ con backup in un'altra regione AWS

Anti-pattern comuni

- Scelta di progettare un'architettura multi-regione quando un'architettura multi-AZ soddisferebbe i requisiti.
- Non si tiene conto delle dipendenze tra i componenti dell'applicazione se i requisiti di resilienza e multi-sede differiscono tra questi componenti.

Vantaggi dell'adozione di questa best practice

Per la resilienza, devi utilizzare un approccio che costruisca livelli di difesa. Un livello protegge dalle interruzioni più piccole e più comuni costruendo un'architettura ad alta disponibilità utilizzando più AZ. Un altro livello di difesa è destinato a proteggere da eventi rari come disastri naturali diffusi e interruzioni a livello regionale. Questo secondo livello implica l'architettura dell'applicazione in modo che si estenda su più Regioni AWS.

- La differenza tra una disponibilità del 99,5% e una del 99,99% è di oltre 3,5 ore al mese. La disponibilità prevista di un carico di lavoro può raggiungere i "quattro nove" solo se si trova in più AZ.
- Eseguendo il carico di lavoro in più AZ, puoi isolare gli errori di alimentazione, raffreddamento e rete e la maggior parte dei disastri naturali come incendi e inondazioni.
- L'implementazione di una strategia multi-regione per il tuo carico di lavoro aiuta a proteggerlo da disastri naturali diffusi che colpiscono un'ampia regione geografica di un paese o da guasti tecnici di portata regionale. Tieni presente che l'implementazione di un'architettura multi-regione può essere molto complessa e di solito non è necessaria per la maggior parte dei carichi di lavoro.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

Per un evento disastroso basato sull'interruzione o la perdita parziale di una zona di disponibilità, l'implementazione di un carico di lavoro a disponibilità elevata in più zone di disponibilità all'interno di una singola Regione AWS aiuta a mitigare i disastri naturali e tecnici. Ogni Regione AWS è composta da più zone di disponibilità, ciascuna isolata dagli errori nelle altre zone e separate da una distanza significativa. Tuttavia, per un evento di disastro che include il rischio di perdere più componenti della zona di disponibilità, che si trovano a una distanza significativa l'uno dall'altro, è necessario implementare opzioni di ripristino di emergenza per mitigare gli errori di portata regionale. Per i

carichi di lavoro che richiedono un'estrema resilienza (infrastrutture critiche, applicazioni sanitarie, infrastrutture di sistemi finanziari e così via), può essere necessaria una strategia multi-regione.

Passaggi dell'implementazione

1. Valutare il carico di lavoro e determinare se le esigenze di resilienza possono essere soddisfatte da un approccio multi-AZ (Regione AWS singola) o se richiedono un approccio multi-regione. L'implementazione di un'architettura multi-regione per soddisfare questi requisiti introdurrà un'ulteriore complessità, quindi considera attentamente il tuo caso d'uso e i suoi requisiti. I requisiti di resilienza possono quasi sempre essere soddisfatti utilizzando un singolo Regione AWS. Per stabilire se è necessario utilizzare più Regioni, considera i seguenti possibili requisiti:
 - a. Ripristino di emergenza: per un evento disastroso basato sull'interruzione o la perdita parziale di una zona di disponibilità, l'implementazione di un carico di lavoro a disponibilità elevata in più zone di disponibilità all'interno di una singola Regione AWS aiuta a mitigare i disastri naturali e tecnici. In caso di eventi disastrosi che comportano il rischio di perdere più componenti della zone di disponibilità, che si trovano a una distanza significativa l'uno dall'altro, è necessario implementare il ripristino di emergenza in più regioni per mitigare i disastri naturali o gli errori tecnici di portata regionale.
 - b. Alta disponibilità: è possibile utilizzare un'architettura multi-regione (utilizzando più AZ in ogni regione) per ottenere una disponibilità superiore a quattro 9 (> 99,99%).
 - c. Localizzazione delle risorse: quando si distribuisce un carico di lavoro a un pubblico globale, è possibile distribuire stack localizzati in diverse Regioni AWS per servire il pubblico di quelle regioni. La localizzazione può includere la lingua, la valuta e i tipi di dati memorizzati.
 - d. Prossimità agli utenti: quando si distribuisce un carico di lavoro a un pubblico globale, è possibile ridurre la latenza distribuendo gli stack alle regioni Regioni AWS in prossimità degli utenti finali.
 - e. Posizione fisica dei dati: alcuni carichi di lavoro sono soggetti a requisiti di residenza dei dati, in base ai quali i dati di determinati utenti devono rimanere all'interno dei confini di un determinato Paese. In base alla normativa in questione, è possibile scegliere di distribuire un intero stack o solo i dati nella Regione AWS all'interno di tali confini.
2. Ecco alcuni esempi di funzionalità multi-AZ fornite dai servizi AWS:
 - a. Per proteggere i carichi di lavoro che utilizzano EC2 o ECS, è necessario distribuire un Elastic Load Balancer davanti alle risorse di calcolo. Elastic Load Balancing quindi fornisce la soluzione per rilevare le istanze nelle zone non integre e instradare il traffico verso quelle integre.
 - i. [Nozioni di base su Application Load Balancers](#)

- ii. [Nozioni di base su Network Load Balancer](#)
 - b. Nel caso di istanze EC2 che eseguono software commerciale pronto all'uso e che non supportano il bilanciamento del carico, puoi ottenere una forma di tolleranza ai guasti implementando una metodologia di ripristino di emergenza multi-AZ.
 - i. [the section called “REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino”](#)
 - c. Per le attività Amazon ECS, distribuire il servizio in modo uniforme su tre AZ per ottenere un equilibrio tra disponibilità e costi.
 - i. [Amazon ECS availability best practices | Containers \(Best practice di disponibilità ECS | Container\)](#)
 - d. Per non Aurora Amazon RDS, puoi scegliere multi-AZ come opzione di configurazione. In caso di errore dell'istanza del database primario, Amazon RDS promuove automaticamente un database standby per ricevere il traffico in un'altra zona di disponibilità. Puoi inoltre creare repliche di lettura multi-regione per migliorare la resilienza.
 - i. [Implementazioni Multi-AZ Amazon RDS](#)
 - ii. [Creazione di una replica di lettura in un'altra Regione AWS](#)
3. Ecco alcuni esempi di funzionalità multi-AZ fornite dai servizi AWS:
- a. Per i carichi di lavoro Amazon S3 in cui la disponibilità multi-AZ è fornita automaticamente dal servizio, considera i punti di accesso multi-regione se è necessaria un'implementazione multi-regione.
 - i. [Punti di accesso multi-regione in Amazon S3](#)
 - b. Per le tabelle DynamoDB in cui la disponibilità multi-AZ è fornita automaticamente dal servizio, è possibile convertire facilmente le tabelle esistenti in tabelle globali per sfruttare più regioni.
 - i. [Convert Your Single-Region Amazon DynamoDB Tables to Global Tables \(Convertire le tabelle Amazon DynamoDB di una singola regione in tabelle globali\)](#)
 - c. Se il carico di lavoro è gestito da Application Load Balancers o da Network Load Balancer, utilizza AWS Global Accelerator per migliorare la disponibilità dell'applicazione indirizzando il traffico verso più regioni che contengono endpoint integri.
 - i. [Endpoints for standard accelerators in AWS Global Accelerator - AWS Global Accelerator \(Endpoint per acceleratori standard in AWS Global Accelerator\) \(amazon.com\)](#)
 - d. Per le applicazioni che sfruttano AWS EventBridge, considera i bus tre regioni per inoltrare gli eventi ad altre regioni selezionate.

- i. [Sending and receiving Amazon EventBridge events between Regioni AWS \(Invio e ricezione di eventi Amazon EventBridge tra regioni AWS\)](#)
- e. Per i database Amazon Aurora, considera i database globali Aurora, che si estendono su più regioni AWS. I cluster esistenti possono essere modificati per aggiungere anche nuove Regioni.
 - i. [Nozioni di base sui database globali Amazon Aurora](#)
- f. Se il carico di lavoro include chiavi di crittografia AWS Key Management Service (AWS KMS), valuta se le chiavi multi-regione sono adatte all'applicazione.
 - i. [Chiavi multi-regione in AWS KMS](#)
- g. Per altre funzionalità del servizio AWS, vedi questa serie di blog su [Creating a Multi-Region Application with AWS Services series \(Creazione di un'applicazione multi-regione con la serie di servizi AWS\)](#)

Livello di impegno per il piano di implementazione: da moderato ad alto

Risorse

Documenti correlati:

- [Creating a Multi-Region Application with AWS Services series \(Creazione di un'applicazione multi-regione con la serie di servizi AWS\)](#)
- [Disaster Recovery \(DR\) Architecture on AWS, Part IV: Multi-site Active/Active \(Architettura di ripristino di emergenza su AWS, parte IV: attiva/attiva multi-sito\)](#)
- [Infrastruttura globale di AWS](#)
- [Domande frequenti su AWS Local Zones](#)
- [Disaster Recovery \(DR\) Architecture on AWS, Part I: Strategies for Recovery in the Cloud \(Architettura di ripristino di emergenza su AWS parte I: strategie per il ripristino nel cloud\)](#)
- [Il ripristino di emergenza è differente nel cloud](#)
- [Tabelle globali: replica multi-regione con DynamoDB](#)

Video correlati:

- [AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications \(Modelli di architettura per applicazioni attive-attive multiregione\) \(ARC209-R2\)](#)
- [Auth0: architettura ad alta disponibilità multi-Regione che raggiunge più di 1,5 miliardi di accessi al mese con failover automatico](#)

Esempi correlati:

- [Disaster Recovery \(DR\) Architecture on AWS, Part I: Strategies for Recovery in the Cloud \(Architettura di ripristino di emergenza su AWS parte I: strategie per il ripristino nel cloud\)](#)
- [DTCC raggiunge livelli di resilienza superiori a quelli che raggiunge on-premise](#)
- [Expedia Group utilizza un'architettura multi-regione, a più zone di disponibilità con un servizio DNS proprietario per aggiungere resilienza alle applicazioni](#)
- [Uber: ripristino di emergenza per Kafka multi-Regione](#)
- [Netflix: attivo-attivo per la resilienza multi-regione](#)
- [Come costruiamo la posizione fisica dei dati per Atlassian Cloud](#)
- [Intuit TurboTax funziona in due regioni](#)

REL10-BP03 Ripristino automatico dei componenti vincolati a una singola posizione

Se i componenti del carico di lavoro possono essere eseguiti in una sola zona di disponibilità o in un data center on-premise, devi rendere possibile la ricostruzione completa del carico di lavoro in base agli obiettivi di ripristino definiti.

Livello di rischio associato alla mancata adozione di questa best practice: medio

Guida all'implementazione

Se, a causa di vincoli tecnologici, non è possibile seguire le linee guida per distribuire il carico di lavoro in più posizioni, è necessario implementare un percorso alternativo mirato alla resilienza. È necessario automatizzare la possibilità di ricreare l'infrastruttura necessaria, ridistribuire le applicazioni e ricreare i dati necessari per questi casi.

Ad esempio, Amazon EMR lancia tutti i nodi per un determinato cluster nella stessa zona di disponibilità: eseguire un cluster nella stessa zona migliora le prestazioni dei flussi di lavoro poiché fornisce una velocità di accesso ai dati più elevata. Se questo componente è necessario per la resilienza del carico di lavoro, è necessario disporre di un modo per implementare nuovamente il cluster e i relativi dati. Inoltre, per Amazon EMR è necessario effettuare il provisioning della ridondanza in modi diversi dall'utilizzo di Multi-AZ. Puoi effettuare il provisioning di [più nodi](#). Usando

il [file system EMR \(EMRFS\)](#), i dati in EMR possono essere ripristinati in Amazon S3, che a sua volta può essere replicato tra più zone di disponibilità o Regioni AWS.

Analogamente, Amazon Redshift per impostazione predefinita effettua il provisioning del cluster in una zona di disponibilità casuale all'interno della Regione AWS selezionata. Tutti i nodi del cluster vengono assegnati nella stessa zona.

Per carichi di lavoro basati su server stateful implementati in un data center on-premise, puoi usare AWS Elastic Disaster Recovery per proteggerli in AWS. Se il carico di lavoro è già ospitato in AWS, puoi usare Elastic Disaster Recovery per proteggerlo in una zona di disponibilità o regione alternativa. Elastic Disaster Recovery usa la replica a livello di blocco continua in un'area di staging leggera per fornire il ripristino rapido e affidabile di applicazioni on-premise e basate sul cloud.

Passaggi dell'implementazione

1. Implementa l'autoriparazione. Distribuisci le tue istanze o container utilizzando, quando possibile, il ridimensionamento automatico. Se non è possibile utilizzare il ridimensionamento automatico, utilizza il ripristino automatico per istanze EC2 o implementa l'automazione di autoriparazione in base agli eventi del ciclo di vita di container Amazon EC2 o ECS.
 - Usa [gruppi con Amazon EC2 Auto Scaling](#) per carichi di lavoro in istanze e container che non abbiano requisiti di un singolo indirizzo IP, indirizzo IP privato o indirizzo IP elastico per le istanze e di metadati delle istanze.
 - È possibile usare i dati utente del modello di avvio per implementare l'automazione per la riparazione automatica della maggior parte dei carichi di lavoro.
 - Usa il [ripristino automatico di istanze Amazon EC2](#) per i carichi di lavoro che richiedono un singolo indirizzo IP, indirizzo IP privato o indirizzo IP elastico per le istanze e metadati delle istanze.
 - Il ripristino automatico invierà avvisi sullo stato del ripristino a un argomento SNS quando viene rilevato l'errore dell'istanza.
 - Usa [eventi del ciclo di vita delle istanze Amazon EC2](#) o [eventi Amazon ECS](#) per automatizzare la riparazione automatica quando il dimensionamento automatico o il recupero in EC2 non sono opzioni valide.
 - Utilizza gli eventi per invocare l'automazione che riparerà il tuo componente secondo la logica di processo richiesta.
 - Proteggi i carichi di lavoro stateful limitati a una singola posizione usando [AWS Elastic Disaster Recovery](#).

Risorse

Documenti correlati:

- [Eventi Amazon ECS](#)
- [Hook del ciclo di vita Amazon EC2 Auto Scaling](#)
- [Recover your instance.](#)
- [Scalabilità automatica del servizio](#)
- [Che cos'è Amazon EC2 Auto Scaling?](#)
- [AWS Elastic Disaster Recovery](#)

REL10-BP04 Utilizzo di architetture a scomparti per limitare la portata dell'impatto

Implementa architetture a scomparti (note anche come architetture basate su celle) per limitare l'effetto di un guasto all'interno di un carico di lavoro a un numero ridotto di componenti.

Risultato desiderato: un'architettura basata su celle usa più istanze isolate di un carico di lavoro, in cui ogni istanza è nota come cella. Ogni cella è indipendente, non condivide lo stato con altre celle e gestisce un sottoinsieme delle richieste complessive del carico di lavoro. Questo approccio riduce il possibile impatto di un errore, ad esempio un aggiornamento software non valido, a una singola cella e alle richieste elaborate. Se un carico di lavoro usa 10 celle per gestire 100 richieste e si verifica un errore, il 90% delle richieste complessive non sarà interessato dall'errore.

Anti-pattern comuni:

- Aumento illimitato delle celle.
- Applicazione di aggiornamenti o implementazioni del codice in tutte le celle contemporaneamente.
- Condivisione dello stato dei componenti tra celle (con l'eccezione del livello di instradamento).
- Aggiunta di logica di business o instradamento complessa al livello di instradamento.
- Le interazioni tra celle non sono ridotte al minimo.

Vantaggi dell'adozione di questa best practice: con un'architettura basata su celle, molti tipi comuni di errore sono limitati alla cella stessa, per un ulteriore isolamento degli errori. Queste limitazioni possono fornire resilienza a determinati tipi di errore altrimenti difficili da contenere, tra

cui implementazioni del codice non riuscite o richieste compromesse o che attivano una modalità di errore specifica, note anche come richieste poison pill.

Guida all'implementazione

Su una nave gli scomparti permettono di limitare la falla di uno scafo a una sola sezione dello scafo. In sistemi complessi questo modello viene spesso replicato per consentire l'isolamento degli errori. Le limitazioni per l'isolamento degli errori riducono l'effetto di un errore all'interno di un carico di lavoro a un numero limitato di componenti. I componenti al di fuori della barriera non subiscono gli effetti del guasto. Utilizzando più barriere per l'isolamento dei guasti, puoi limitare l'impatto sul carico di lavoro. In AWS i clienti possono usare più zone di disponibilità e regioni per fornire l'isolamento degli errori, ma questo concetto può essere esteso anche all'architettura del carico di lavoro.

Il carico di lavoro complessivo viene partizionato in celle tramite una chiave di partizione. Questa chiave deve essere allineata alla granularità del servizio o al modo naturale in cui il carico di lavoro del servizio può essere suddiviso con interazioni minime tra celle. Esempi di chiavi di partizione sono un ID cliente, un ID risorsa o qualsiasi altro parametro facilmente accessibile nella maggior parte delle chiamate API. Un livello di instradamento alle celle distribuisce le richieste a singole celle in base alla chiave di partizione e presenta un unico endpoint ai client.

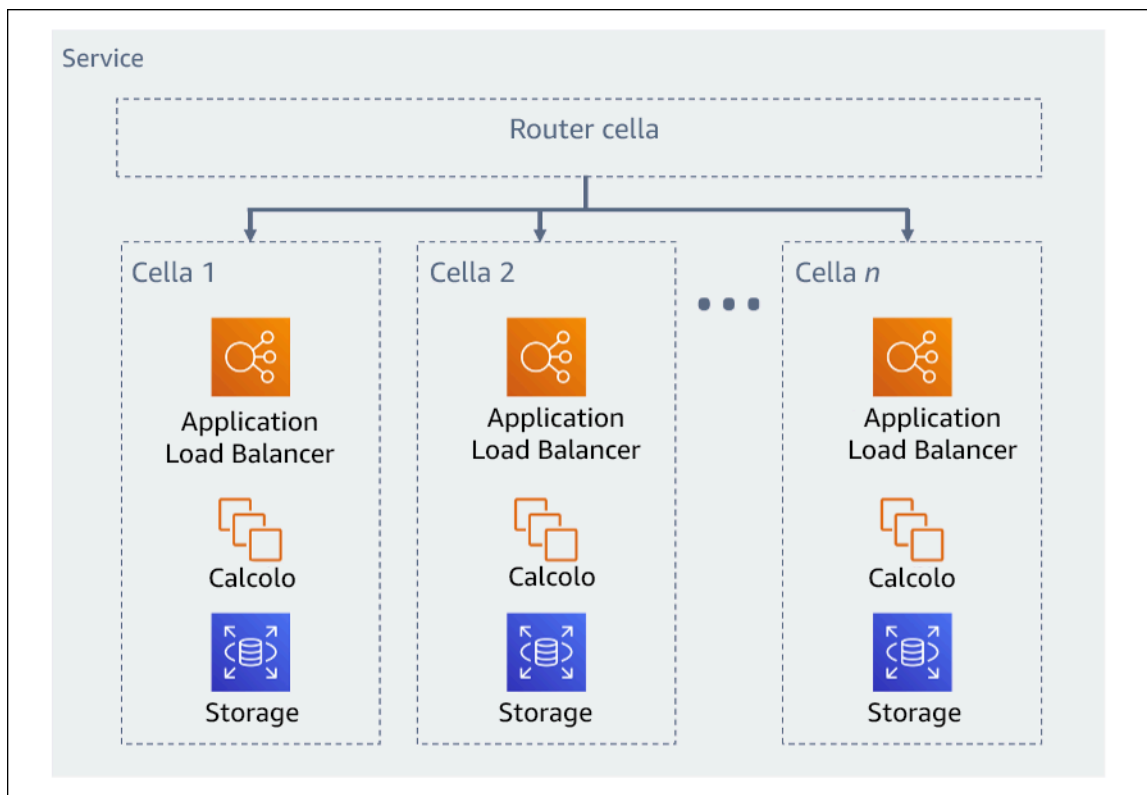


Figura 11. Architettura basata su celle

Passaggi dell'implementazione

Nel progettare un'architettura basata su celle, devi tenere conto di diversi aspetti della progettazione:

1. Chiave di partizione: la scelta della chiave di partizione impone alcune considerazioni speciali.
 - Questa chiave deve essere allineata alla granularità del servizio o al modo naturale in cui il carico di lavoro del servizio può essere suddiviso con interazioni minime tra celle. Alcuni esempi: `ID cliente` oppure `ID risorsa`.
 - La chiave di partizione deve essere disponibile in tutte le richieste, direttamente o in modo da poter essere facilmente dedotta in modo deterministico da altri parametri.
2. Mappatura persistente delle celle: i servizi a monte devono interagire solo con un'unica cella per l'intero ciclo di vita delle risorse correlate.
 - A seconda del carico di lavoro, può essere necessaria una strategia di migrazione delle celle per la migrazione dei dati da una cella a un'altra. Un possibile scenario in cui è necessaria la migrazione delle celle è quando una risorsa o un utente specifico nel carico di lavoro diventa troppo grande e richiede una cella dedicata.
 - Le celle non devono condividere lo stato o i componenti.
 - Di conseguenza, l'interazione tra celle deve essere evitata o mantenuta al minimo, in quanto le interazioni creano dipendenze tra le celle e riducono quindi i vantaggi forniti dall'isolamento degli errori.
3. Livello di instradamento: è un componente condiviso tra celle e di conseguenza non può seguire la stessa strategia di compartimentazione applicata alle celle.
 - È consigliabile che il livello di instradamento distribuisca richieste a singole celle usando un algoritmo di mappatura delle partizioni efficiente in termini di risorse di calcolo, ad esempio combinando funzioni hash crittografiche e aritmetica modulare per mappare le chiavi di partizione alle celle.
 - Per evitare l'impatto su più celle, il livello di instradamento deve restare il più semplice e orizzontalmente scalabile possibile, evitando logica di business complessa in questo livello. Questo approccio offre il vantaggio aggiuntivo di semplificare la comprensione del suo comportamento previsto in ogni momento, permettendo test esaustivi. Come spiegato da Colm MacCárthaigh in [Reliability, constant work, and a good cup of coffee](#), progettazioni semplici e modelli di lavoro costanti producono sistemi affidabili e riducono l'antifragilità.
4. Dimensione delle celle: le celle devono avere una dimensione massima che non deve essere superata.

- La dimensione massima deve essere identificata attraverso l'esecuzione di test completi, fino a raggiungere i punti di rottura e definire i margini operativi. Per ulteriori informazioni su come implementare procedure di test, consulta [REL07-BP04 Esecuzione di un test di carico sul carico di lavoro](#)
 - L'aumento del carico di lavoro complessivo deve essere gestito tramite l'aggiunta di celle, in modo da poterlo dimensionare in base al crescere della domanda.
5. Strategie multi-Az e multi-regione: è consigliabile utilizzare più livelli di resilienza a tipi di errore diversi.
- Per la resilienza, devi utilizzare un approccio che costruisca livelli di difesa. Un livello protegge dalle interruzioni minime e più comuni attraverso la creazione di un'architettura a disponibilità elevata tramite più zone di disponibilità. Un altro livello di difesa è destinato a proteggere da eventi rari come disastri naturali diffusi e interruzioni a livello regionale. Questo secondo livello implica l'architettura dell'applicazione in modo che si estenda su più Regioni AWS. L'implementazione di una strategia multi-regione per il tuo carico di lavoro aiuta a proteggerlo da disastri naturali diffusi che colpiscono un'ampia regione geografica di un paese o da guasti tecnici di portata regionale. Tieni presente che l'implementazione di un'architettura multi-regione può essere molto complessa e di solito non è necessaria per la maggior parte dei carichi di lavoro. Per ulteriori informazioni, consulta [REL10-BP02 Selezione delle posizioni appropriate per la tua implementazione multiposizione](#).
6. Implementazione del codice: una strategia di implementazione scaglionata del codice è preferibile rispetto all'implementazione di modifiche del codice a tutte le celle contemporaneamente.
- In questo modo, è possibile ridurre al minimo eventuali errori in più celle a causa di un'implementazione non corretta o dell'errore umano. Per ulteriori informazioni, consulta [Automatizzazione di implementazioni pratiche e sicure](#).

Livello di rischio associato alla mancata adozione di questa best practice: elevato

Risorse

Best practice correlate:

- [REL07-BP04 Esecuzione di un test di carico sul carico di lavoro](#)
- [REL10-BP02 Selezione delle posizioni appropriate per la tua implementazione multiposizione](#)

Documenti correlati:

- [Reliability, constant work, and a good cup of coffee](#)
- [AWS e compartimentazione](#)
- [Isolamento del carico di lavoro tramite sharding casuale](#)
- [Automatizzazione di implementazioni pratiche e sicure](#)

Video correlati:

- [AWS re:Invent 2018: Cicli chiusi e menti aperte: come assumere il controllo di sistemi grandi e piccoli](#)
- [AWS re:Invent 2018: AWS riduce al minimo il raggio di esplosione degli errori \(ARC338\)](#)
- [Partizionamento casuale: AWS re:Invent 2019: Introduzione alla Libreria dei costruttori di Amazon \(DOP328\)](#)
- [AWS Summit ANZ 2021:Gli errori si verificano sempre e ovunque: una progettazione per la resilienza](#)

Esempi correlati:

- [Well-Architected Lab: Isolamento degli errori con il partizionamento casuale](#)

Progettazione di un carico di lavoro resistente agli errori dei componenti

I carichi di lavoro con requisiti di disponibilità elevata e MTTR (Mean Time To Recovery) basso devono essere progettati per garantire la resilienza.

Best practice

- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)
- [REL11-BP02 Failover e passaggio a risorse integre](#)
- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)
- [REL11-BP04 Fare affidamento al piano dati invece che al piano di controllo durante il ripristino](#)
- [REL11-BP05 Utilizzo della stabilità statica per evitare un comportamento bimodale](#)
- [REL11-BP06 Invio di notifiche quando gli eventi influiscono sulla disponibilità](#)

- [REL11-BP07 Progettazione del prodotto in modo da soddisfare gli obiettivi di disponibilità e i contratti sul livello di servizio per i tempi di attività](#)

REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti

Monitora costantemente lo stato del carico di lavoro, in modo che tu e i tuoi sistemi automatizzati siate consapevoli di errori o guasti non appena si verificano. Monitora gli indicatori chiave di prestazioni (KPI) in base al valore aziendale.

Tutti i meccanismi di ripristino e correzione devono essere in grado di rilevare rapidamente i problemi. I guasti tecnici devono essere rilevati prima in modo che possano essere risolti. Tuttavia, la disponibilità si basa sulla capacità del carico di lavoro di fornire valore aziendale, quindi gli indicatori chiave di prestazione (KPI) che misurano questo aspetto devono far parte della strategia di rilevamento e correzione.

Risultato desiderato: I componenti essenziali di un carico di lavoro vengono monitorati in modo indipendente per rilevare guasti e fornire avvisi quando e dove si verificano.

Anti-pattern comuni:

- Non sono stati configurati allarmi, pertanto le interruzioni si verificano senza notifica.
- Gli allarmi esistono, ma a soglie che non forniscono tempo adeguato per reagire.
- I parametri non vengono raccolti abbastanza spesso da soddisfare l'obiettivo di tempo di ripristino (RTO, recovery time objective).
- Solo le interfacce del carico di lavoro rivolte al cliente vengono monitorate attivamente.
- Viene effettuata solo la raccolta di parametri tecnici, senza includere quelli delle funzioni aziendali.
- Non è presente alcun parametro che misuri l'esperienza utente del carico di lavoro.
- Vengono creati troppi monitoraggi.

Vantaggi dell'adozione di questa best practice: Eseguire un monitoraggio appropriato a tutti i livelli consente di ridurre i tempi di rilevamento, velocizzando quindi il ripristino.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

Identifica tutti i carichi di lavoro che verranno esaminati per il monitoraggio. Dopo aver identificato tutti i componenti del carico di lavoro da monitorare, devi determinare l'intervallo di monitoraggio. L'intervallo di monitoraggio ha un impatto diretto sulla velocità con cui il ripristino viene avviato, che dipende dal tempo impiegato per rilevare un errore. Il tempo medio di rilevamento (MTTD) è il tempo che intercorre tra il verificarsi di un guasto e l'inizio delle operazioni di riparazione. L'elenco dei servizi deve essere ampio e completo.

Il monitoraggio deve includere tutti i livelli dello stack applicativo, come applicazione, piattaforma, infrastruttura e rete.

La strategia di monitoraggio deve tenere in considerazione l'impatto di guasti nell'area grigia. Per ulteriori dettagli sui guasti nell'area grigia, consulta [il whitepaper Gray failures](#) in the Advanced Multi-AZ Resilience Patterns

Passaggi dell'implementazione

- L'intervallo di monitoraggio dipende dalla velocità con cui è necessario ripristinare. Il tempo di ripristino dipende dal tempo necessario a ripristinare, perciò è necessario determinare la frequenza della raccolta considerando tale tempo e l'obiettivo di tempo di ripristino (RTO, recovery time objective).
- Configura il monitoraggio dettagliato per componenti e servizi gestiti.
 - Determina se [il monitoraggio dettagliato per le istanze EC2](#) e [Auto Scaling](#) è necessario. Il monitoraggio dettagliato fornisce metriche a intervalli di un minuto, mentre il monitoraggio predefinito fornisce metriche a intervalli di cinque minuti.
 - Determina se [il monitoraggio avanzato](#) per RDS è necessario. Il monitoraggio avanzato utilizza un agente sulle istanze RDS per ottenere informazioni utili su diversi processi o thread.
 - Determina i requisiti di monitoraggio dei componenti serverless critici per [Lambda](#), [API Gateway](#), [Amazon EKS](#), [Amazon ECS](#) e tutti i tipi di [sistema di bilanciamento del carico](#).
 - Determina i requisiti di monitoraggio dei componenti di archiviazione per [Amazon S3](#), [Amazon FSx](#), [Amazon EFS](#) e [Amazon EBS](#).
- Crea [metriche personalizzate](#) per misurare gli indicatori di prestazione (KPI) fondamentali per il tuo business. I carichi di lavoro implementano funzioni aziendali fondamentali, che devono essere utilizzate come KPI che aiutano a identificare quando si verifica un problema indiretto.
- Monitoraggio della presenza di errori nell'esperienza utente tramite le canary degli utenti [Test delle transazioni sintetiche](#) (noto anche come test canary, ma da non confondere con l'implementazione

canary) è uno dei processi di test più importanti in quanto è in grado di eseguire e simulare il comportamento dei clienti. Esegui questi test costantemente sugli endpoint del carico di lavoro da diverse posizioni remote.

- Crea [metriche personalizzate](#) che monitorino l'esperienza dell'utente. Dotare l'esperienza del cliente di strumenti consente di determinare quando essa peggiora.
- [Imposta allarmi](#) per rilevare quando una qualsiasi parte del carico di lavoro non funziona correttamente e per indicare quando dimensionare automaticamente le risorse. È possibile mostrare visivamente gli allarmi sulle dashboard, inviarli tramite Amazon SNS o e-mail e utilizzarli con Auto Scaling per aumentare o ridurre le risorse del carico di lavoro.
- Crea [dashboard](#) per visualizzare le metriche. Utilizza le dashboard per visualizzare tendenze, valori anomali e altri indicatori di potenziali problemi, oppure per fornire un'indicazione dei problemi che potresti voler approfondire.
- Crea [il monitoraggio del tracciamento distribuito](#) per i tuoi servizi. Con il monitoraggio distribuito puoi comprendere le prestazioni della tua applicazione e dei relativi servizi sottostanti per identificare e risolvere la causa ultima di problemi ed errori riguardanti le prestazioni.
- Utilizza [CloudWatch](#) oppure [X-Ray](#) per creare dashboard di sistemi di monitoraggio e di raccolta dati in una regione e in un account separati.
- Crea l'integrazione per [Amazon Health Aware](#) per consentire il monitoraggio della visibilità sulle risorse AWS che potrebbero presentare un deterioramento. Per i carichi di lavoro aziendali essenziali, questa soluzione fornisce l'accesso ad avvisi proattivi e in tempo reale per i servizi AWS.

Risorse

Best practice correlate:

- [Definizione di disponibilità](#)
- [REL11-BP06 Invio di notifiche quando gli eventi influiscono sulla disponibilità](#)

Documenti correlati:

- [Amazon CloudWatch Synthetics consente di creare i Canary dell'utente](#)
- [Abilitare o disabilitare il monitoraggio dettagliato della propria istanza](#)
- [Monitoraggio avanzato](#)

- [Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch](#)
- [Pubblicazione di parametri personalizzati](#)
- [Utilizzo degli allarmi di Amazon CloudWatch](#)
- [Using CloudWatch Dashboards](#)
- [Using Cross Region Cross Account CloudWatch Dashboards](#)
- [Using Cross Region Cross Account X-Ray Tracing](#)
- [Understanding availability](#)
- [Implementing Amazon Health Aware \(AHA\)](#)

Video correlati:

- [Mitigating gray failures](#)

Esempi correlati:

- [Well-Architected Lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability](#)
- [One Observability Workshop: Explore X-Ray](#)

Strumenti correlati:

- [CloudWatch](#)
- [CloudWatch X-Ray](#)

REL11-BP02 Failover e passaggio a risorse integre

Se si verifica un errore in una risorsa, le risorse integre dovrebbero continuare a soddisfare le richieste. Per posizioni compromesse (ad esempio una zona di disponibilità o una Regione AWS), assicurati di disporre di sistemi che possano eseguire il failover e passare a risorse integre in posizioni non danneggiate.

Durante la progettazione di un servizio, distribuisci il carico tra risorse, zone di disponibilità o regioni. Pertanto, il guasto o la compromissione di una singola risorsa può essere mitigato spostando il traffico sulle risorse integre rimanenti. Considera come vengono rilevati e indirizzati i servizi in caso di guasto.

Progetta i tuoi servizi tenendo a mente il recupero dai guasti. In AWS, progettiamo servizi per ridurre al minimo i tempi di recupero da guasti e l'impatto sui dati. I nostri servizi utilizzano principalmente archivi di dati che riconoscono le richieste solo dopo che queste sono state archiviate in modo duraturo su più repliche in una Regione. Sono costruiti con il criterio dell'isolamento basato sulle celle ed utilizzano l'isolamento dei guasti fornito dalle zone di disponibilità. Facciamo ampio uso dell'automazione nelle nostre procedure operative. Ottimizziamo anche la nostra funzionalità di sostituzione e riavvio per un ripristino rapidamente dalle interruzioni.

I modelli e i progetti che consentono il failover variano a seconda dei servizi della AWS. Molti servizi AWS gestiti nativi si trovano in più zone di disponibilità (come Lambda o API Gateway) in modo nativo. Altri servizi AWS (come EC2 ed EKS) richiedono procedure ottimali specifiche per supportare il failover delle risorse o l'archiviazione di dati tra le zone di disponibilità.

Il monitoraggio deve essere impostato per verificare che la risorsa di failover sia integra, tenere traccia dell'avanzamento del failover delle risorse e monitorare il ripristino dei processi aziendali.

Risultato desiderato: I sistemi sono in grado di utilizzare automaticamente o manualmente nuove risorse per il ripristino dopo un evento di deterioramento.

Anti-pattern comuni:

- La pianificazione degli errori non fa parte della fase di pianificazione e progettazione.
- L'obiettivo del tempo di ripristino (RTO) e l'obiettivo del punto di ripristino (RPO) non sono stabiliti.
- Monitoraggio insufficiente per rilevare risorse difettose.
- Isolamento adeguato dei domini di errore.
- Il failover multi-regione non è considerato.
- Il rilevamento dei guasti è troppo sensibile o aggressivo quando si decide di eseguire il failover.
- Non è possibile testare o convalidare il progetto di failover.
- Esecuzione dell'automazione del risanamento automatico, ma senza la notifica della necessità di una correzione.
- Mancanza di un periodo di mitigazione per evitare che l'errore si ripresenti troppo presto.

Vantaggi dell'adozione di questa best practice: È possibile creare sistemi più resilienti che mantengano l'affidabilità in caso di guasti eseguendo prima un deterioramento lento e poi un ripristino rapido.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

I servizi AWS, come [Elastic Load Balancing](#) e [Amazon EC2 Auto Scaling](#), aiutano a distribuire il carico tra risorse e zone di disponibilità. Pertanto, il guasto di una singola risorsa (come un'istanza EC2) o la compromissione di una zona di disponibilità possono essere mitigati spostando il traffico sulle risorse integre rimanenti.

Per i carichi di lavoro multi-regione, i progetti sono più complicati. Ad esempio, le repliche di lettura multi-regione consentono di implementare i dati su Regioni AWS multiple. Tuttavia, il failover è ancora necessario per promuovere la replica di lettura a principale e quindi indirizzare il traffico verso il nuovo endpoint. Amazon Route 53, Route 53 Route 53 ARC, CloudFront e AWS Global Accelerator possono aiutare a instradare il traffico tra le Regioni AWS.

Servizi AWS come Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge o Amazon DynamoDB vengono implementati automaticamente in più zone di disponibilità da AWS. In caso di guasto, questi servizi AWS instradano automaticamente il traffico verso posizioni integre. I dati sono archiviati in modo ridondante in più zone di disponibilità e rimangono disponibili.

Per Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon EKS o Amazon ECS, l'implementazione Multi-AZ è un'opzione di configurazione. AWS può indirizzare il traffico verso l'istanza integra se viene avviato il failover. Questa azione di failover può essere intrapresa direttamente da AWS o su richiesta del cliente.

Per istanze Amazon EC2, Amazon Redshift, attività Amazon ECS o pod Amazon EKS, sei tu a scegliere in quali zone di disponibilità eseguire la distribuzione. Per alcuni progetti, Elastic Load Balancing fornisce la soluzione per rilevare istanze in zone corrotte e instradare il traffico verso quelle integre. Elastic Load Balancing può anche indirizzare il traffico verso i componenti del data center on-premise.

Per il failover del traffico multi-regione, il reindirizzamento può sfruttare Amazon Route 53, Route 53 ARC, AWS Global Accelerator, Route 53 Private DNS for VPCs o CloudFront per fornire una modalità per definire domini Internet e assegnare policy di routing, compresi i controlli dell'integrità, per instradare il traffico verso regioni integre. AWS Global Accelerator fornisce indirizzi IP statici che operano come punto di ingresso fisso all'applicazione, che indirizzano il traffico verso gli endpoint delle Regioni AWS di tua scelta utilizzando la rete AWS globale anziché Internet per prestazioni e affidabilità migliori.

Passaggi dell'implementazione

- Crea progetti di failover per tutte le applicazioni e i servizi appropriati. Isola ogni componente dell'architettura e crea progetti di failover che soddisfino l'RTO e l'RPO per ogni componente.
- Configura ambienti inferiori (come sviluppo o test) con tutti i servizi necessari per disporre di un piano di failover. Implementa le soluzioni utilizzando l'infrastruttura come codice (IaC) per garantire la ripetibilità.
- Configura un sito di ripristino, ad esempio una seconda regione, per implementare e testare i progetti di failover. Se necessario, le risorse per i test possono essere configurate temporaneamente per limitare i costi aggiuntivi.
- Determina quali piani di failover sono automatizzati da AWS, quali possono essere automatizzati da un processo DevOps e quali possono essere manuali. Documenta e misura l'RTO e l'RPO di ogni servizio.
- Crea un playbook per il failover e includi tutti i passaggi necessari per eseguire il failover di ogni risorsa, applicazione e servizio.
- Crea un playbook di failback e includi tutti i passaggi per eseguire il failback (con tempistiche) di ogni risorsa, applicazione e servizio.
- Crea un piano per avviare e testare il playbook. Usa simulazioni e test del caos per testare i passaggi e l'automazione del playbook.
- Per posizioni compromesse (ad esempio una zona di disponibilità o una Regione AWS), assicurati di disporre di sistemi che possano eseguire il failover e passare a risorse integre in posizioni non danneggiate. Verifica la quota, i livelli di dimensionamento automatico e le risorse in esecuzione prima dei test di failover.

Risorse

Best practice Well-Architected correlate:

- [REL13- Pianificazione per il disaster recovery \(DR\)](#)
- [REL10 - Utilizzo dell'isolamento dei guasti per proteggere il carico di lavoro](#)

Documenti correlati:

- [Setting RTO and RPO Targets](#)
- [Set up Route 53 ARC with application loadbalancers](#)

- [Failover using Route 53 Weighted routing](#)
- [DR with Route 53 ARC](#)
- [EC2 with autoscaling](#)
- [EC2 Deployments - Multi-AZ](#)
- [ECS Deployments - Multi-AZ](#)
- [Switch traffic using Route 53 ARC](#)
- [Lambda with an Application Load Balancer and Failover](#)
- [ACM Replication and Failover](#)
- [Parameter Store Replication and Failover](#)
- [ECR cross region replication and Failover](#)
- [Secrets manager cross region replication configuration](#)
- [Enable cross region replication for EFS and Failover](#)
- [EFS Cross Region Replication and Failover](#)
- [Networking Failover](#)
- [S3 Endpoint failover using MRAP](#)
- [Crea una replica tra regioni per S3](#)
- [Failover Regional API Gateway with Route 53 ARC](#)
- [Failover using multi-region global accelerator](#)
- [Failover with DRS](#)
- [Creating Disaster Recovery Mechanisms Using Amazon Route 53](#)

Esempi correlati:

- [Disaster Recovery on AWS](#)
- [Elastic Disaster Recovery on AWS](#)

REL11-BP03 Automatizzazione della riparazione a tutti i livelli

Al rilevamento di un guasto, utilizza funzionalità automatizzate per eseguire azioni da correggere. I guasti possono essere riparati automaticamente tramite meccanismi di servizio interni oppure riavviando o rimuovendo le risorse tramite azioni correttive.

Per applicazioni gestite dal cliente e per il ripristino tra regioni, è possibile attingere a modelli di ripristino e processi di riparazione automatizzati dalle [best practice esistenti](#).

La possibilità di riavviare o rimuovere una risorsa è uno strumento importante per risolvere i guasti. Una best practice consiste nel rendere i servizi stateless, ove possibile. In questo modo si evita la perdita di dati o di disponibilità durante il riavvio della risorsa. Nel cloud è possibile, e in genere si dovrebbe, sostituire l'intera risorsa (ad esempio, un'istanza di calcolo o una funzione serverless) come parte del riavvio. Il riavvio stesso è un modo semplice e affidabile per eseguire il ripristino in caso di guasto. Molti tipi diversi di guasto si verificano nei carichi di lavoro. Possono verificarsi guasti a livello di hardware, software, comunicazione e operazioni.

Il riavvio o i nuovi tentativi come pratiche risolutive si applicano anche alle richieste di rete. Adotta lo stesso approccio di ripristino sia a un timeout di rete sia a un guasto di dipendenza in cui la dipendenza restituisce un guasto. Entrambi gli eventi hanno un effetto simile sul sistema, quindi piuttosto che tentare di trasformare entrambi gli eventi in un caso speciale, adotta una strategia analoga di nuovi tentativi limitati con un backoff e un jitter esponenziali. La capacità di riavvio è un meccanismo di ripristino presente nelle architetture di cluster ROC (Recovery-oriented computing) e ad alta disponibilità.

Risultato desiderato: Vengono eseguite azioni automatiche di risoluzione a seguito del rilevamento di un errore.

Anti-pattern comuni:

- Provisioning di risorse senza dimensionamento automatico.
- Implementazione individuale di applicazioni in istanze/container.
- Distribuzione di applicazioni che non possono essere distribuite in più posizioni senza utilizzare il ripristino automatico.
- Riparazione manuale delle applicazioni che il dimensionamento e il ripristino automatici non sono stati in grado di riparare.
- Nessuna automazione dei database di failover.
- Mancanza di metodi automatizzati per reinstradare il traffico verso nuovi endpoint.
- Nessuna replica dell'archiviazione.

Vantaggi dell'adozione di questa best practice: La riparazione automatica può ridurre il tempo medio di ripristino e migliorare la disponibilità.

Livello di rischio associato se questa best practice non fosse adottata: alto

Guida all'implementazione

I progetti per Amazon EKS o altri servizi Kubernetes devono includere il numero minimo e massimo di repliche o di stateful set e la dimensione minima dei cluster e dei gruppi di nodi. Questi meccanismi forniscono una quantità minima di risorse di elaborazione continuamente disponibili mentre riparano automaticamente eventuali guasti utilizzando il piano di controllo Kubernetes.

I modelli di progettazione a cui si accede tramite un sistema di bilanciamento del carico che utilizza cluster di calcolo dovrebbero sfruttare i gruppi Auto Scaling. Elastic Load Balancing (ELB) distribuisce automaticamente il traffico delle applicazioni in entrata su più destinazioni e applicazioni virtuali in una o più zone di disponibilità (AZ).

I progetti basati su cluster computing che non utilizzano il bilanciamento del carico devono avere dimensioni progettate per la perdita di almeno un nodo. Ciò consentirà al servizio di rimanere in esecuzione con una capacità potenzialmente ridotta durante il ripristino di un nuovo nodo. Servizi di esempio sono Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK e Amazon OpenSearch Service. Molti di questi servizi possono essere progettati con funzionalità di riparazione automatica aggiuntive. Alcune tecnologie di cluster devono generare un avviso in caso di perdita di un nodo attivando un flusso di lavoro automatico o manuale per creare un nuovo nodo. È possibile automatizzare questo flusso di lavoro utilizzando AWS Systems Manager per risolvere rapidamente i problemi.

Amazon EventBridge può essere utilizzato per monitorare e filtrare eventi come allarmi CloudWatch o cambiamenti di stato in altri servizi AWS. In base alle informazioni sugli eventi, può quindi richiamare AWS Lambda, Systems Manager Automation o altre destinazioni per eseguire una logica di riparazione personalizzata sul carico di lavoro. È possibile configurare Amazon EC2 Auto Scaling per verificare lo stato delle istanze EC2. Se l'istanza è in uno stato diverso da quello in esecuzione o se lo stato del sistema è danneggiato, Amazon EC2 Auto Scaling considera l'istanza come non integra e ne avvia una sostitutiva. Per le sostituzioni su larga scala (ad esempio la perdita di un'intera zona di disponibilità), è preferibile adottare la stabilità statica per ottenere un'elevata disponibilità.

Passaggi dell'implementazione

- Utilizza gruppi di Auto Scaling per distribuire livelli in un carico di lavoro. [Auto Scaling](#) è in grado di eseguire il risanamento automatico sulle applicazioni stateless e aggiungere o rimuovere capacità.
- Per le istanze di calcolo indicate in precedenza, usa [il bilanciamento del carico](#) e scegli il tipo di sistema di bilanciamento del carico appropriato.

- Considera l'opzione della riparazione per Amazon RDS. Con le istanze di standby, configura il [failover automatico](#) verso l'istanza di standby. Per le repliche in lettura Amazon RDS, è necessario un flusso di lavoro automatizzato per rendere primaria una replica di lettura.
- Implementa [il ripristino automatico su istanze EC2](#) che includono applicazioni distribuite non implementabili in più sedi e che possono tollerare il riavvio in caso di guasto. Il ripristino automatico può essere utilizzato per sostituire l'hardware guasto e riavviare l'istanza quando l'applicazione non è in grado di essere distribuita in più posizioni. I metadati dell'istanza e gli indirizzi IP associati vengono conservati, così come i [volumi EBS](#) e i punti di montaggio su [Amazon Elastic File System](#) o [i file system per Lustre](#) e [Windows](#). Utilizzando [AWS OpsWorks](#) puoi configurare la riparazione automatica delle istanze EC2 a livello del layer.
- Implementa il ripristino automatico utilizzando [AWS Step Functions](#) e [AWS Lambda](#) quando non è possibile utilizzare il dimensionamento automatico o il ripristino automatico o quando il ripristino automatico non va a buon fine. Quando non puoi utilizzare il dimensionamento automatico né il ripristino automatico o il ripristino automatico non riesce, puoi automatizzare la riparazione utilizzando AWS Step Functions e AWS Lambda.
- [Amazon EventBridge](#) può essere usato per monitorare e filtrare eventi come [avvisi CloudWatch](#) o cambiamenti di stato in altri servizi AWS. In base alle informazioni sugli eventi, può quindi richiamare AWS Lambda (o altre destinazioni) per eseguire una logica di riparazione personalizzata sul tuo carico di lavoro.

Risorse

Best practice correlate:

- [Definizione di disponibilità](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)

Documenti correlati:

- [Come funziona AWS Auto Scaling](#)
- [Ripristino automatico Amazon EC2](#)
- [Amazon Elastic Block Store \(Amazon EBS\)](#)
- [Amazon Elastic File System \(Amazon EFS\)](#)
- [What is Amazon FSx for Lustre?](#)
- [What is Amazon FSx for Windows File Server?](#)

- [AWS OpsWorks: utilizzare il ripristino automatico per sostituire le istanze in errore](#)
- [Che cos'è AWS Step Functions?](#)
- [Che cos'è AWS Lambda?](#)
- [Che cos'è Amazon EventBridge?](#)
- [Utilizzo degli allarmi di Amazon CloudWatch](#)
- [Amazon RDS Failover](#)
- [SSM - Systems Manager Automation](#)
- [Resilient Architecture Best Practices](#)

Video correlati:

- [Automatically Provision and Scale OpenSearch Service](#)
- [Amazon RDS Failover Automatically](#)

Esempi correlati:

- [Workshop su Auto Scaling](#)
- [Workshop su Amazon RDS Failover](#)

Strumenti correlati:

- [CloudWatch](#)
- [CloudWatch X-Ray](#)

REL11-BP04 Fare affidamento al piano dati invece che al piano di controllo durante il ripristino

I piani di controllo forniscono le API amministrative utilizzate per creare, leggere e descrivere, aggiornare, eliminare ed elencare (CRUDL) risorse, mentre i piani di dati gestiscono il traffico quotidiano del servizio. Durante l'implementazione di risposte di ripristino o mitigazione a eventi che possono influire sulla resilienza, concentrati sull'utilizzo di un numero minimo di operazioni del piano di controllo per ripristinare, ridimensionare, ristabilire, riparare il servizio o eseguirne il failover. Le operazioni del piano dati dovrebbero avere la precedenza su qualsiasi attività durante questi eventi che causano deterioramento.

Ad esempio, le seguenti sono tutte azioni del piano di controllo: avvio di una nuova istanza di calcolo, creazione di storage a blocchi e descrizione dei servizi di coda. Quando avvii istanze di calcolo, il piano di controllo deve eseguire diverse attività, come trovare un host fisico con capacità, allocare interfacce di rete, preparare volumi di storage a blocchi locali, generare credenziali e aggiungere regole di sicurezza. I piani di controllo tendono ad avere un'orchestrazione complicata.

Risultato desiderato: Quando lo stato di risorsa viene compromesso, il sistema è in grado di ripristinarsi automaticamente o manualmente spostando il traffico da risorse danneggiate a risorse integre.

Anti-pattern comuni:

- Dipendenza dalla modifica dei record DNS per reindirizzare il traffico.
- Dipendenza dalle operazioni di dimensionamento del piano di controllo per sostituire i componenti danneggiati a causa di un provisioning delle risorse insufficiente.
- Affidarsi ad azioni intense, multiservizio e multi-API del piano di controllo per porre rimedio a qualsiasi categoria di deterioramento.

Vantaggi dell'adozione di questa best practice: Una maggiore percentuale di successo in termini di riparazione automatica può ridurre il tempo medio di ripristino e migliorare la disponibilità del carico di lavoro.

Livello di rischio associato se questa best practice non fosse adottata: Medio: per determinati tipi di deterioramento del servizio, vengono compromessi i piani di controllo. Le dipendenze dall'uso intenso del piano di controllo per la riparazione possono aumentare il tempo di ripristino (RTO) e il tempo medio di ripristino (MTTR).

Guida all'implementazione

Per limitare le azioni del piano dati, esegui una valutazione servizio per servizio per determinare le azioni necessarie per ripristinarlo.

Sfrutta Amazon Route 53 Application Recovery Controller per spostare il traffico DNS. Queste funzionalità monitorano continuamente la capacità dell'applicazione di ristabilirsi dai guasti e consentono di controllarne il ripristino su più Regioni AWS, zone di disponibilità e on-premise.

Le policy di instradamento di Route 53 utilizzano il piano di controllo, quindi non fare affidamento su di esso per il ripristino. I piani dati di Route 53 rispondono alle query DNS ed eseguono e valutano i

controlli di integrità. Sono distribuiti a livello globale e progettati per un [accordo sul livello di servizio \(SLA\) con disponibilità al 100%](#).

Le API e la console di gestione di Route 53, dove si creano, aggiornano ed eliminano le risorse di Route 53, funzionano su piani di controllo progettati per privilegiare la forte coerenza e la durata necessarie per la gestione del DNS. A tal fine, i piani di controllo sono situati in un'unica regione: Stati Uniti orientali (Virginia settentrionale). Sebbene entrambi i sistemi siano costruiti per essere molto affidabili, i piani di controllo non sono inclusi nello SLA. Possono verificarsi eventi rari in cui la progettazione resiliente del piano dati consente di mantenere la disponibilità mentre i piani di controllo non lo fanno. Per i meccanismi di ripristino di emergenza e failover, utilizzare le funzioni del piano dati per garantire la migliore affidabilità possibile.

Per Amazon EC2, utilizzare progetti di stabilità statica per limitare le azioni del piano di controllo. Le azioni del piano di controllo includono l'aumento delle risorse, in maniera individuale o utilizzando gruppi Auto Scaling (ASG). Per ottenere i massimi livelli di resilienza, è necessario fornire una capacità sufficiente nel cluster utilizzato per il failover. Se è necessario limitare questa soglia di capacità, imposta acceleratori sull'intero sistema end-to-end per limitare in modo sicuro il traffico totale che raggiunge il set limitato di risorse.

L'utilizzo di servizi come Amazon DynamoDB, Amazon API Gateway, sistemi di bilanciamento del carico e AWS Lambda serverless avviene sfruttando il piano dati. Tuttavia, la creazione di nuove funzioni, sistemi di bilanciamento del carico, gateway API o tabelle DynamoDB è un'azione del piano di controllo e deve essere completata prima del deterioramento come preparazione a un evento e test delle azioni di failover. Per Amazon RDS, le azioni del piano dati consentono l'accesso ai dati.

Per ulteriori informazioni sui piani dati, sui piani di controllo e su come AWS costruisce i servizi per soddisfare gli obiettivi di alta disponibilità, consulta [Stabilità statica utilizzando le zone di disponibilità](#).

Capire quali operazioni sono sul piano dati e quali sul piano di controllo.

Passaggi dell'implementazione

Per ogni carico di lavoro che deve essere ripristinato dopo un evento di deterioramento, valuta il runbook di failover, il design ad alta disponibilità, il progetto di riparazione automatica o il piano di ripristino delle risorse HA. Identifica ogni azione che potrebbe essere considerata un'azione del piano di controllo.

Prendi in considerazione la possibilità di modificare l'azione di controllo in un'azione del piano dati:

- Auto Scaling (piano di controllo) rispetto alle risorse Amazon EC2 predimensionate (piano dati)

- Esegui la migrazione verso Lambda e i relativi metodi di dimensionamento (piano dati) oppure verso Amazon EC2 e ASG (piano di controllo)
- Valuta qualsiasi progetto utilizzando Kubernetes e considerando la natura delle azioni del piano di controllo. L'aggiunta di pod è un'azione del piano dati in Kubernetes. Le azioni devono limitarsi all'aggiunta di pod e non all'aggiunta di nodi. L'utilizzo [di nodi con provisioning eccessivo](#) è il metodo preferibile per limitare le azioni del piano di controllo

Prendi in considerazione approcci alternativi che consentano alle azioni del piano dati di incidere sulla stessa correzione.

- Modifica del record di Route 53 (piano di controllo) o Route 53 ARC (piano dati)
- [Controlli dell'integrità di Route 53 per aggiornamenti più automatizzati](#)

Se il servizio è mission critical, prendi in considerazione alcuni servizi in una regione secondaria per consentire più azioni del piano di controllo e del piano dati in una regione non interessata dal problema.

- Amazon EC2 Auto Scaling o Amazon EKS in una regione primaria rispetto a Amazon EC2 Auto Scaling o Amazon EKS in una regione secondaria e instradamento del traffico verso una regione secondaria (azione del piano di controllo)
- Crea una replica di lettura nella regione secondaria o tenta la stessa azione nella regione principale (azione del piano di controllo)

Risorse

Best practice correlate:

- [Definizione di disponibilità](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)

Documenti correlati:

- [Partner APN: partner che possono essere d'aiuto con l'automazione della tua tolleranza ai guasti](#)
- [Marketplace AWS: prodotti utilizzabili per la tolleranza ai guasti](#)
- [The Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control](#)

- [API Amazon DynamoDB \(piano di controllo e piano dati\)](#)
- [AWS Lambda Executions](#) (suddivise in piano di controllo e piano dati)
- [Piano dati di AWS Elemental MediaStore](#)
- [Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack](#)
- [Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack](#)
- [Creating Disaster Recovery Mechanisms Using Amazon Route 53](#)
- [Che cos'è il Sistema di controllo Route 53 per il ripristino di applicazioni](#)
- [Piano di controllo e piano dati di Kubernetes](#)

Video correlati:

- [Back to Basics - Using Static Stability](#)
- [Building resilient multi-site workloads using AWS global services](#)

Esempi correlati:

- [Introducing Amazon Route 53 Application Recovery Controller](#)
- [The Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control](#)
- [Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack](#)
- [Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack](#)
- [Stabilità statica utilizzando le zone di disponibilità](#)

Strumenti correlati:

- [Amazon CloudWatch](#)
- [AWS X-Ray](#)

REL11-BP05 Utilizzo della stabilità statica per evitare un comportamento bimodale

I carichi di lavoro devono essere staticamente stabili e funzionare in una singola modalità normale. Il comportamento bimodale si verifica quando il carico di lavoro presenta un comportamento diverso in modalità normale e in modalità di guasto.

Ad esempio, ciò potrebbe accadere nel momento in cui si prova a ripristinare un guasto nella zona di disponibilità avviando nuove istanze in una zona di disponibilità diversa. Questo approccio può comportare una risposta bimodale durante una modalità di guasto. È invece necessario creare carichi di lavoro che siano staticamente stabili e operino in una sola modalità. In questo esempio, le nuove istanze avrebbero dovuto essere rese disponibili nella seconda zona di disponibilità già prima del guasto. Questo design staticamente stabile verifica che il carico di lavoro funzioni in una sola modalità.

Risultato desiderato: i carichi di lavoro non presentano un comportamento bimodale in modalità normale e in modalità di guasto.

Anti-pattern comuni:

- Supporre che le risorse possano sempre essere rese disponibili indipendentemente dall'ambito del guasto.
- Tentare di acquisire risorse in modo dinamico durante un guasto.
- Non rendere disponibili risorse adeguate tra zone o regioni diverse fino a quando non si verifica un guasto.
- Considerare i progetti staticamente stabili solo per risorse di calcolo.

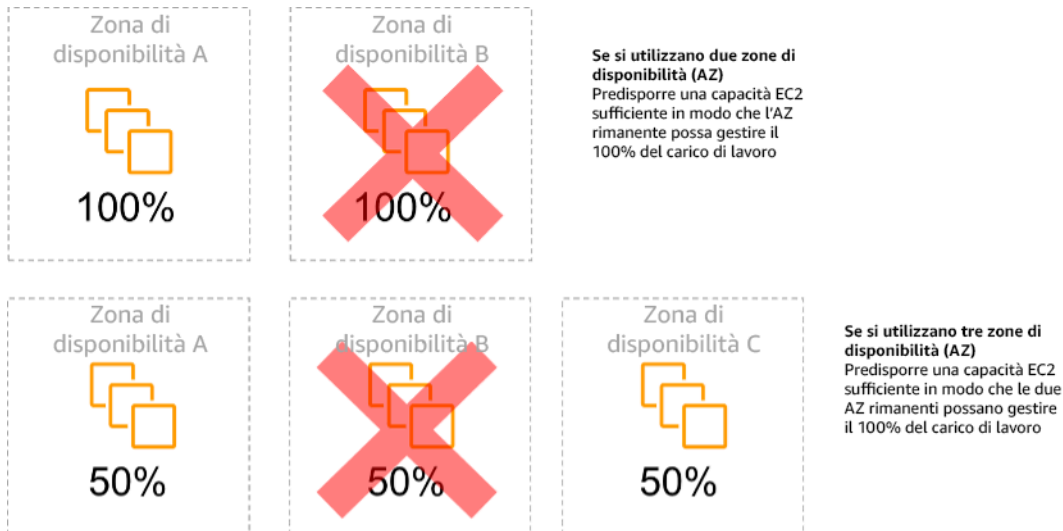
Vantaggi dell'adozione di questa best practice: I carichi di lavoro eseguiti con progetti staticamente stabili sono in grado di avere risultati prevedibili durante eventi normali e di guasto.

Livello di rischio associato se questa best practice non fosse adottata: medio

Guida all'implementazione

Il comportamento bimodale ha luogo quando il carico di lavoro mostra un comportamento diverso in modalità normale e di guasto, ad esempio facendo affidamento sull'avvio di nuove istanze se una zona di disponibilità presenta un malfunzionamento. Un esempio di comportamento bimodale è quello che si verifica quando design stabili di Amazon EC2 rendono disponibili un numero sufficiente

di istanze in ciascuna zona di disponibilità per gestire il carico di lavoro in caso di rimozione di una di tali zone. Elastic Load Balancing o Amazon Route 53 possono effettuare un controllo di integrità per spostare un carico lontano dalle istanze danneggiate. Dopo il trasferimento del traffico, è possibile utilizzare AWS Auto Scaling per sostituire in modo asincrono le istanze della zona interessata dal guasto avviandole nelle zone integre. La stabilità statica per il deployment delle risorse di calcolo (ad esempio istanze EC2 o container) determinerà la massima affidabilità.



Stabilità statica delle istanze EC2 nelle diverse zone di disponibilità

Questo approccio deve essere valutato rispetto al costo associato al modello e al valore aziendale attribuito al mantenimento della disponibilità del carico di lavoro in tutti i casi di resilienza. Fornire una minore capacità di elaborazione e affidarsi all'avvio di nuove istanze in caso di guasto è meno costoso. Tuttavia, in caso di guasti su larga scala, come una zona di disponibilità o un problema a livello regionale, tale approccio è meno efficace, perché si basa su un piano operativo e sulla disponibilità di risorse sufficienti nelle zone o nelle regioni non interessate dal problema.

La soluzione deve inoltre valutare l'affidabilità rispetto ai costi necessari per il carico di lavoro. Gli approcci che garantiscono la stabilità statica si applicano a una varietà di architetture, tra cui istanze di calcolo distribuite tra zone di disponibilità, progetti di repliche di lettura di database, progetti di cluster Kubernetes (Amazon EKS) e architetture di failover multiregione.

È anche possibile implementare un progetto staticamente più stabile utilizzando più risorse in ciascuna zona. Aggiungendo più zone, si riduce la quantità di elaborazione aggiuntiva necessaria per la stabilità statica.

Un altro esempio di comportamento bimodale potrebbe derivare da un timeout di rete in grado di causare un tentativo di aggiornamento dello stato di configurazione dell'intero sistema. Ciò potrebbe

aggiungere un carico imprevisto su un altro componente che potrebbe quindi generare un errore, innescando ulteriori conseguenze impreviste. Questo loop di feedback negativo influisce sulla disponibilità del carico di lavoro. Al contrario, è possibile creare sistemi che siano staticamente stabili e funzionino in una sola modalità. Un progetto staticamente stabile potrebbe eseguire con continuità un'attività e aggiornare sempre, con cadenza regolare, lo stato della configurazione. Quando una chiamata fallisce, il carico di lavoro può utilizzare il valore precedentemente memorizzato nella cache e segnalare un allarme.

Un altro esempio di comportamento bimodale è consentire ai client di bypassare la cache del carico di lavoro quando si verificano dei guasti. Potrebbe sembrare una soluzione che soddisfa le esigenze del client, ma non dovrebbe essere consentita perché modifica in modo significativo le richieste sul carico di lavoro e potrebbe causare dei guasti.

Valuta i carichi di lavoro critici per determinare quali carichi di lavoro richiedono questo tipo di progettazione di resilienza. Per quelli considerati critici, deve essere esaminato ogni componente dell'applicazione. Alcuni tipi di servizi che richiedono valutazioni di stabilità statica sono:

- Calcolo: Amazon EC2, EKS-EC2, ECS-EC2, EMR-EC2
- Database: Amazon Redshift, Amazon RDS, Amazon Aurora
- Storage: Amazon S3 (Zona singola), Amazon EFS (supporti), Amazon FSx (supporti)
- Sistemi di bilanciamento del carico: In base a determinati modelli

Passaggi dell'implementazione

- Realizzare sistemi che siano staticamente stabili e operino in una sola modalità. In questo caso, effettuare il provisioning di un numero sufficiente di istanze in ogni zona o regione di disponibilità per gestire la capacità del carico di lavoro qualora venga rimossa una zona o regione di disponibilità. Per l'indirizzamento verso risorse integre è possibile utilizzare una varietà di servizi, come:
 - [Cross Region DNS Routing](#)
 - [Routing Amazon S3 multiregionale MRAP](#)
 - [AWS Global Accelerator](#)
 - [Amazon Route 53 Application Recovery Controller](#)
- Configura [repliche di lettura del database](#) per tenere conto della perdita di una singola istanza primaria o di una replica di lettura. Se il traffico viene servito da repliche di lettura, la quantità in

ogni zona di disponibilità e in ogni regione deve corrispondere al fabbisogno complessivo in caso di guasto della zona o della regione.

- Configurare i dati critici nel sistema di archiviazione Amazon S3 progettato per essere staticamente stabile rispetto ai dati archiviati in caso di guasto della zona di disponibilità. Se si verifica un [Se viene utilizzata la classe di archiviazione Amazon S3 One Zone-IA](#) questa non deve essere considerata staticamente stabile, poiché la perdita di tale zona riduce al minimo l'accesso ai dati archiviati.
- [I sistemi di bilanciamento del carico](#) sono a volte configurati in modo errato o sono progettati per servire una zona di disponibilità specifica. In questo caso, il progetto staticamente stabile potrebbe consistere nel distribuire un carico di lavoro su più zone di disponibilità seguendo un design più complesso. Il design originale potrebbe essere utilizzato per ridurre il traffico tra zone per motivi di sicurezza, latenza o costi.

Risorse

Best practice Well-Architected correlate:

- [Definizione di disponibilità](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)
- [REL11-BP04 Fare affidamento al piano dati invece che al piano di controllo durante il ripristino](#)

Documenti correlati:

- [Minimizing Dependencies in a Disaster Recovery Plan](#)
- [The Amazon Builders' Library: Stabilità statica con le zone di disponibilità](#)
- [Fault Isolation Boundaries](#)
- [Stabilità statica utilizzando le zone di disponibilità](#)
- [Multi-Zone RDS](#)
- [Minimizing Dependencies in a Disaster Recovery Plan](#)
- [Cross Region DNS Routing](#)
- [Routing Amazon S3 multiregionale MRAP](#)
- [AWS Global Accelerator](#)
- [Route 53 ARC](#)

- [Amazon S3 a singola zona](#)
- [Cross Zone Load Balancing](#)

Video correlati:

- [Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library \(DOP328\)](#)

Esempi correlati:

- [The Amazon Builders' Library: Stabilità statica con le zone di disponibilità](#)

REL11-BP06 Invio di notifiche quando gli eventi influiscono sulla disponibilità

Le notifiche vengono inviate al rilevamento del superamento delle soglie, anche se l'evento causato dal problema è stato risolto automaticamente.

Il ripristino automatizzato consente al carico di lavoro di risultare affidabile. Tuttavia, potrebbe anche nascondere problemi sottostanti che devono essere risolti. Implementa il monitoraggio e gli eventi appropriati in modo da poter rilevare i modelli di problemi, inclusi quelli risolti dalla diagnostica automatica e risolvere così i problemi della causa principale.

I sistemi resilienti sono progettati in modo che gli eventi di degrado vengano immediatamente comunicati ai team appropriati. Queste notifiche devono essere inviate tramite uno o più canali di comunicazione.

Risultato desiderato: Gli avvisi vengono inviati immediatamente ai team operativi quando vengono superate soglie come i tassi di errore, la latenza o altri parametri critici degli indicatori chiave di prestazione (KPI), in modo che questi problemi vengano risolti il prima possibile e l'impatto sugli utenti sia evitato o ridotto al minimo.

Anti-pattern comuni:

- Invio di un numero eccessivo di avvisi.
- Invio di avvisi non utilizzabili.
- Impostazione di soglie di allarme troppo alte (troppo sensibili) o troppo basse (troppo poco sensibili).

- Mancato invio di avvisi per dipendenze esterne.
- Non considerando [guasti nell'area grigia](#) nella progettazione di sistemi di monitoraggio e allarmi.
- Eseguire l'automazione del risanamento, ma senza avvisare il team competente che era necessario un intervento di ripristino.

Vantaggi dell'adozione di questa best practice: Le notifiche di ripristino rendono i team operativi e aziendali consapevoli dei peggioramenti del servizio in modo che possano reagire immediatamente per ridurre al minimo sia il tempo medio di rilevamento (MTTD) che il tempo medio di riparazione (MTTR). Le notifiche degli eventi di ripristino consentono anche di non ignorare i problemi che si verificano di rado.

Livello di rischio associato se questa best practice non fosse adottata: medio. La mancata implementazione di meccanismi di monitoraggio e notifica degli eventi appropriati può comportare l'impossibilità di rilevare i modelli di problemi, compresi quelli risolti mediante la correzione automatica. Un team verrà informato del degrado del sistema solo nel momento in cui gli utenti contattano il servizio clienti o per caso.

Guida all'implementazione

Quando si definisce una strategia di monitoraggio, un allarme attivato è un evento comune. Questo evento dovrebbe contenere un identificatore dell'allarme, lo stato dell'allarme (ad esempio IN ALLARME o OK) e dettagli su cosa l'ha innescato. In molti casi, è necessario rilevare un evento di allarme e inviare una notifica tramite e-mail. Questo è un esempio di azione su un allarme. La notifica degli allarmi è fondamentale per l'osservabilità, in quanto informa le persone giuste della presenza di un problema. Tuttavia, quando le operazioni eseguite sulla base degli eventi raggiungono un certo grado di maturità nella soluzione di osservabilità, è possibile risolvere automaticamente il problema senza la necessità dell'intervento umano.

Una volta stabiliti gli allarmi di monitoraggio dei KPI, è necessario inviare avvisi ai team appropriati quando vengono superate le soglie. Tali avvisi possono essere utilizzati anche per attivare processi automatizzati che tenteranno di porre rimedio al danno o alla compromissione.

Per un monitoraggio delle soglie più complesso, è necessario prendere in considerazione gli allarmi compositi. Gli allarmi compositi utilizzano una serie di allarmi di monitoraggio dei KPI per creare un avviso basato sulla logica di business operativa. Gli allarmi CloudWatch possono essere configurati per l'invio di e-mail o per la registrazione di file di log nei sistemi di monitoraggio di terze parti tramite l'integrazione con Amazon SNS o Amazon EventBridge.

Passaggi dell'implementazione

Crea vari tipi di allarmi in base al modo in cui vengono monitorati i carichi di lavoro, ad esempio:

- Gli allarmi applicativi vengono utilizzati per rilevare quando una parte del carico di lavoro non funziona correttamente.
- [Allarmi infrastrutturali](#) indicano quando dimensionare le risorse. Gli allarmi possono essere visualizzati visivamente sui pannelli di controllo, essere inviati tramite Amazon SNS o tramite e-mail e utilizzati con Auto Scaling per aumentare o diminuire le risorse del carico di lavoro.
- Semplice [allarmi statici](#) per monitorare quando una metrica supera una soglia statica per un numero specificato di periodi di valutazione.
- [Gli allarmi compositi](#) possono tenere conto di allarmi complessi provenienti da più fonti.
- Una volta creato l'allarme è possibile generare eventi di notifica appropriati. Puoi richiamare direttamente una [API Amazon SNS](#) per inviare notifiche e collegare qualsiasi automazione per la correzione o la comunicazione.
- integra [Amazon Health Aware](#) per consentire il monitoraggio della visibilità sulle risorse AWS che potrebbero presentare un deterioramento. Per i carichi di lavoro aziendali essenziali, questa soluzione fornisce l'accesso ad avvisi proattivi e in tempo reale per i servizi AWS.

Risorse

Best practice Well-Architected correlate:

- [Definizione di disponibilità](#)

Documenti correlati:

- [Creare un allarme CloudWatch basato su una soglia statica](#)
- [Che cos'è Amazon EventBridge?](#)
- [Che cos'è Amazon Simple Notification Service?](#)
- [Pubblicazione di parametri personalizzati](#)
- [Utilizzo degli allarmi di Amazon CloudWatch](#)
- [Amazon Health Aware \(AHA\)](#)
- [Configurazione degli allarmi compositi di CloudWatch](#)

- [Cosa c'è di nuovo in AWS Observability at re:Invent 2022](#)

Strumenti correlati:

- [CloudWatch](#)
- [CloudWatch X-Ray](#)

REL11-BP07 Progettazione del prodotto in modo da soddisfare gli obiettivi di disponibilità e i contratti sul livello di servizio per i tempi di attività

Progetta il tuo prodotto in modo da soddisfare gli obiettivi di disponibilità e i contratti sul livello di servizio per i tempi di attività. Se pubblichi o accetti privatamente obiettivi di disponibilità o contratti sul livello di servizio per i tempi di attività, verifica che l'architettura e i processi operativi siano progettati in modo da supportarli.

Risultato desiderato: definizione per ogni applicazione di un obiettivo definito per la disponibilità e di un contratto sul livello di servizio per le metriche di prestazioni, che possono essere monitorati e gestiti per realizzare i risultati aziendali.

Anti-pattern comuni:

- Progettazione e implementazione di carichi di lavoro senza impostare alcun contratto sul livello di servizio.
- Impostazione di metriche elevate per il contratto sul livello di servizio senza fondamento logico o requisiti aziendali.
- Impostazione di contratti sul livello di servizio senza tenere conto delle dipendenze e dei relativi contratti sul livello di servizio sottostanti.
- Progettazione delle applicazioni senza tenere conto del Modello di responsabilità condivisa per la resilienza.

Vantaggi dell'adozione di questa best practice: la progettazione di applicazioni in base ai principali obiettivi di resilienza ti aiuta a realizzare gli obiettivi aziendali e a soddisfare le aspettative dei clienti. Questi obiettivi orientano un processo di progettazione delle applicazioni in grado di valutare diverse tecnologie e tenere conto di vari compromessi.

Livello di rischio associato se questa best practice non fosse adottata: Medium

Guida all'implementazione

La progettazione delle applicazioni deve tenere conto di una serie eterogenea di requisiti derivati da obiettivi aziendali, operativi e finanziari. Nell'ambito dei requisiti operativi, i carichi di lavoro devono avere obiettivi specifici in termini di metriche di resilienza, in modo da poter essere monitorati e supportati correttamente. Le metriche di resilienza non devono essere impostate o derivate dopo l'implementazione del carico di lavoro. Devono invece essere definite durante la fase di progettazione e contribuire a determinare i diversi compromessi e decisioni.

- Ogni carico di lavoro deve avere una serie di metriche di resilienza propria. Le metriche possono essere diverse da quelle di altre applicazioni aziendali.
- La riduzione delle dipendenze può avere un impatto positivo sulla disponibilità. Per ogni carico di lavoro è necessario considerare le dipendenze e i relativi contratti sul livello di servizio. In generale, seleziona dipendenze con obiettivi di disponibilità uguali o maggiori rispetto agli obiettivi del carico di lavoro.
- Prendi in considerazione progettazioni senza integrazioni serrate in modo che il carico di lavoro possa funzionare correttamente anche in caso di dipendenze compromesse, se possibile.
- Riduci le dipendenze del piano di controllo (control-plane), in particolare durante un ripristino o un peggioramento delle prestazioni. Valuta le progettazioni staticamente stabili per carichi di lavoro mission critical. Usa il contenimento delle risorse per aumentare la disponibilità delle dipendenze in un carico di lavoro.
- La visibilità e la strumentazione sono essenziali per soddisfare i contratti sul livello di servizio attraverso la riduzione del tempo medio di rilevamento (MTTD) e del tempo medio di ripristino (MTTR).
- Errori meno frequenti (tempo medio tra guasti, o MTBF, più lungo), tempi di rilevamento degli errori più brevi (MTTD minore) e tempi di riparazione più brevi (MTTR minore) sono i tre fattori usati per migliorare la disponibilità in sistemi distribuiti.
- La definizione e l'applicazione di metriche di resilienza per un carico di lavoro sono essenziali per qualsiasi progettazione efficace. Queste progettazioni devono tenere conto dei compromessi introdotti dalla complessità di progettazione, delle dipendenze dei servizi, delle prestazioni, del dimensionamento e dei costi.

Passaggi dell'implementazione

- Esamina e documenta la progettazione del carico di lavoro cercando di rispondere alle domande seguenti:

- Dove vengono usati piani di controllo (control-plane) nel carico di lavoro?
- Come viene implementata la tolleranza ai guasti nel carico di lavoro?
- Quali sono i modelli di progettazione per dimensionamento, scalabilità automatica, ridondanza e componenti a disponibilità elevata?
- Quali sono i requisiti per la coerenza e la disponibilità dei dati?
- Vi sono aspetti da considerare in fatto di contenimento delle risorse o stabilità statica delle risorse?
- Quali sono le dipendenze dei servizi?
- Definisci insieme agli stakeholder le metriche per il contratto sul livello di servizio in base all'architettura del carico di lavoro. Tieni conto dei contratti sul livello di servizio di tutte le dipendenze usate dal carico di lavoro.
- Una volta definiti gli obiettivi del contratto sul livello di servizio, ottimizza l'architettura in modo da soddisfare il contratto.
- Una volta impostata una progettazione che soddisfa il contratto sul livello di servizio, implementa modifiche operative, automazione dei processi e runbook anch'essi incentrati sulla riduzione dell'MTTD e dell'MTTR.
- Dopo aver implementato il contratto sul livello di servizio, devi monitorarlo e documentarlo.

Risorse

Best practice correlate:

- [REL03-BP01 Scelta del tipo di segmentazione del carico di lavoro](#)
- [REL10-BP01 Implementazione del carico di lavoro in diversi luoghi](#)
- [REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti](#)
- [REL11-BP03 Automatizzazione della riparazione a tutti i livelli](#)
- [REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos](#)
- [REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati](#)
- [Comprendere lo stato del carico di lavoro](#)

Documenti correlati:

- [Disponibilità con ridondanza](#)

- [Principio dell'affidabilità: disponibilità](#)
- [Misurazione della disponibilità](#)
- [Limiti di isolamento dei guasti di AWS](#)
- [Modello di responsabilità condivisa per la resilienza](#)
- [stabilità statica utilizzando le zone di disponibilità](#)
- [Contratti sul livello di servizio AWS](#)
- [Linee guida per le architetture basate su celle su AWS](#)
- [Infrastruttura AWS](#)
- [Whitepaper sui modelli di resilienza multi-AZ avanzati](#)

Servizi correlati:

- [Amazon CloudWatch](#)
- [AWS Config](#)
- [AWS Trusted Advisor](#)

Test dell'affidabilità

Dopo aver progettato il carico di lavoro in modo da essere resiliente alle sollecitazioni della produzione, i test sono l'unico modo per garantire il funzionamento corretto e offrire la resilienza prevista.

Effettua test per verificare che il carico di lavoro soddisfi i requisiti funzionali e non funzionali, evitando bug o colli di bottiglia delle prestazioni che potrebbero compromettere l'affidabilità. Testa la resilienza del tuo carico di lavoro per trovare bug latenti che emergono solo in produzione. Esegui questi test regolarmente.

Best practice

- [REL12-BP01 Utilizzo dei playbook per analizzare gli errori](#)
- [REL12-BP02 Esecuzione di analisi post-incidente](#)
- [REL12-BP03 Test dei requisiti funzionali](#)
- [REL12-BP04 Test dei requisiti di dimensionamento e prestazioni](#)
- [REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos](#)
- [REL12-BP06 Esecuzione regolare di giornate di gioco](#)

REL12-BP01 Utilizzo dei playbook per analizzare gli errori

Abilita risposte coerenti e tempestive a scenari di guasto che non sono ben compresi, documentando il processo di analisi nei playbook. I playbook sono le fasi predefinite eseguite per identificare i fattori che contribuiscono a uno scenario di guasto. I risultati provenienti da un passaggio del processo vengono utilizzati per stabilire i passaggi successivi da intraprendere fino all'identificazione o alla risoluzione del problema.

Il playbook è una pianificazione proattiva che è necessario eseguire, in modo da potere intraprendere azioni reattive in modo efficace. Quando durante la produzione si verificano scenari di guasto non coperti dal playbook, risolvi innanzitutto il problema (spegni l'incendio). Quindi torna indietro e osserva le fasi intraprese per risolvere il problema e utilizzale per aggiungere una nuova voce al playbook.

Tieni presente che i playbook vengono utilizzati in risposta a specifici incidenti, mentre i runbook vengono utilizzati per ottenere esiti specifici. Spesso, i runbook vengono utilizzati per le attività di routine e i playbook vengono utilizzati per rispondere a eventi non di routine.

Anti-pattern comuni:

- Pianificare la distribuzione di un carico di lavoro senza conoscere i processi per diagnosticare i problemi o rispondere agli incidenti.
- Decisioni non pianificate sui sistemi da cui raccogliere log e parametri durante l'analisi di un evento.
- Non conservare parametri e eventi abbastanza a lungo da poter recuperare i dati.

Vantaggi dell'adozione di questa best practice: L'acquisizione di playbook garantisce l'esecuzione coerente dei processi. La codifica dei playbook limita l'introduzione di errori derivanti dall'attività manuale. L'automazione dei playbook riduce il tempo necessario per rispondere a un evento eliminando il requisito per l'intervento dei membri del team o fornendo loro informazioni aggiuntive quando inizia l'intervento.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

- Utilizza playbook per identificare i problemi. I playbook sono processi documentati per eseguire indagini sui problemi. Abilita risposte coerenti e tempestive agli scenari di errore documentando i processi nei playbook. I playbook devono contenere le informazioni e le istruzioni necessarie

affinché una persona adeguatamente qualificata possa raccogliere le informazioni applicabili, identificare potenziali fonti di errore, isolare i guasti e stabilire i fattori che contribuiscono all'origine di un problema (eseguire l'analisi post-incidente).

- Implementazione dei playbook come codice. Esegui le operazioni come codice mediante lo scripting dei playbook per assicurare coerenza e ridurre gli errori causati dai processi manuali. I playbook possono essere composti da più script che rappresentano le diverse fasi che potrebbero essere necessarie per identificare i fattori che contribuiscono all'origine di un problema. Le attività dei runbook possono essere attivate o eseguite nell'ambito delle attività dei playbook oppure possono richiedere l'esecuzione di un playbook in risposta agli eventi identificati.
 - [Automazione dei playbook operativi con AWS Systems Manager](#)
 - [AWS Systems Manager Run Command](#)
 - [AWS Systems Manager Automation](#)
 - [Cos'è AWS Lambda?](#)
 - [Che cos'è Amazon EventBridge?](#)
 - [Utilizzo degli allarmi di Amazon CloudWatch](#)

Risorse

Documenti correlati:

- [AWS Systems Manager Automation](#)
- [AWS Systems Manager Run Command](#)
- [Automazione dei playbook operativi con AWS Systems Manager](#)
- [Utilizzo degli allarmi di Amazon CloudWatch](#)
- [Utilizzo di Canary \(Amazon CloudWatch Synthetics\)](#)
- [Che cos'è Amazon EventBridge?](#)
- [Cos'è AWS Lambda?](#)

Esempi correlati:

- [Automating operations with Playbooks and Runbooks \(Automazione delle operazioni con Playbook e Runbook\)](#)

REL12-BP02 Esecuzione di analisi post-incidente

Esamina gli eventi che influiscono sui clienti e identifica i fattori che vi hanno contribuito e gli elementi di azione preventivi. Utilizza queste informazioni per sviluppare modi per limitare o prevenire il ripetersi degli imprevisti. Sviluppa procedure per attivare risposte rapide ed efficaci. Comunica i fattori che hanno contribuito al presentarsi dell'imprevisto e le azioni correttive secondo necessità, specificamente mirate per il pubblico di destinazione. All'occorrenza, adotta un metodo per comunicare queste cause ad altri.

Valuta perché i test esistenti non hanno individuato il problema. Aggiungi i test per questo caso se i test non esistono già.

Risultato desiderato: i tuoi team hanno un approccio coerente e concordato alla gestione dell'analisi post-incidente. Un meccanismo è il [processo di correzione dell'errore \(COE\)](#). Il processo COE aiuta i team a individuare, comprendere e gestire le cause principali degli incidenti, creando al contempo meccanismi e guardrail per limitare la probabilità che lo stesso incidente si ripeta.

Anti-pattern comuni:

- Individuare i fattori che hanno contribuito al verificarsi dell'incidente, ma non continuare a cercare in maniera più approfondita altri potenziali problemi e approcci da mitigare.
- Identificare le cause degli errori umani senza fornire alcuna formazione o automazione che potrebbe prevenirli.
- Concentrarsi sull'attribuzione delle colpe piuttosto che sulla comprensione della causa principale, creando così una cultura della paura e ostacolando la comunicazione costruttiva
- Mancata condivisione delle informazioni, che mantiene i risultati dell'analisi degli incidenti all'interno di un gruppo ristretto e impedisce ad altri di beneficiare delle lezioni apprese
- Nessun meccanismo che consenta di acquisire le conoscenze formali; in questo modo si perdono informazioni preziose in quanto non vengono preservate le lezioni apprese sotto forma di best practice aggiornate, con il conseguente rischio che gli incidenti si ripetano con la stessa causa principale o causa simile

Vantaggi dell'adozione di questa best practice: l'esecuzione di analisi post-incidente e la condivisione dei risultati consente ad altri carichi di lavoro di mitigare il rischio se hanno implementato gli stessi fattori che hanno contribuito al verificarsi dell'incidente e consente loro di implementare la mitigazione o il ripristino automatico prima che si verifichi un incidente.

Livello di rischio associato alla mancata adozione di questa best practice: elevato

Guida all'implementazione

Una buona analisi post-incidente fornisce opportunità per proporre soluzioni comuni a problemi con modelli di architettura utilizzati in altri punti nei tuoi sistemi.

Un elemento fondamentale del processo COE è la documentazione e la risoluzione dei problemi. È consigliabile definire un modo standard per documentare le cause principali critiche e assicurarsi che queste vengano esaminate e risolte. Assegna in modo chiaro il responsabile del processo di analisi post-incidente. Designa un team o una persona responsabile della supervisione delle indagini e dei follow-up degli incidenti.

Promuovi una cultura basata sull'apprendimento e sul miglioramento piuttosto che sull'attribuzione di colpe. Insisti sul fatto che l'obiettivo è prevenire incidenti futuri e non penalizzare le persone.

Sviluppa procedure ben definite per l'esecuzione delle analisi post-incidente. Queste procedure dovrebbero stabilire le misure da adottare, le informazioni da raccogliere e le questioni chiave da risolvere durante l'analisi. Svolgi indagini approfondite sugli incidenti, andando oltre le cause immediate per identificare le cause principali e i fattori determinanti. Usa tecniche come i [Cinque Perché](#) per analizzare approfonditamente i problemi sottostanti.

Mantieni un archivio delle conclusioni derivanti dalle analisi degli incidenti. Queste conoscenze formali possono fungere da riferimento per futuri incidenti e attività di prevenzione. Condividi i risultati e gli approfondimenti delle analisi post-incidente e valuta la possibilità di organizzare riunioni di revisione post-incidente con invito aperto per discutere i risultati e le conclusioni.

Passaggi dell'implementazione

- Durante l'analisi post-incidente, assicurati che il processo non comporti la colpevolizzazione delle parti coinvolte. Ciò consente alle parti interessate di essere imparziali rispetto delle azioni correttive proposte, nonché di promuovere l'autovalutazione e la collaborazione a livello di team.
- Definisci una procedura standardizzata per documentare i problemi critici. Una struttura di esempio per tale documento è la seguente:
 - Cosa è successo?
 - Quale impatto ha avuto su clienti e attività?
 - Qual è stata la causa principale?
 - Di quali dati disponi a supporto di questo problema?
 - Ad esempio, metriche e grafici

- Quali sono state le principali implicazioni sui pilastri critici, specialmente per quanto riguarda la sicurezza?
 - Quando progetti l'architettura dei carichi di lavoro, devi trovare dei compromessi tra i pilastri su cui si regge il contesto aziendale. Questo tipo di decisioni aziendali deve essere alla base delle tue priorità ingegneristiche. Potresti ridurre i costi a spese dell'affidabilità in ambienti di sviluppo oppure, per quanto riguarda le soluzioni mission-critical, potresti ottimizzare l'affidabilità con costi maggiori. La sicurezza ha la massima priorità quando si tratta di proteggere i tuoi clienti.
- Quali lezioni hai imparato?
- Quali azioni correttive stai adottando?
 - Azioni correttive
 - Articoli correlati
- Crea precise procedure operative standard per lo svolgimento delle analisi post-incidente.
- Configura un processo standardizzato di segnalazione degli incidenti. Documenta in modo esaustivo tutti gli incidenti, includendo il rapporto iniziale sull'incidente, i log, le comunicazioni e le azioni intraprese durante l'incidente.
- Ricorda che un incidente non necessariamente comporta un'interruzione del servizio. Potrebbe trattarsi di un near miss o di un sistema che funziona in modo imprevisto pur continuando a svolgere la sua funzione aziendale.
- Migliora continuamente il processo di analisi post-incidente sulla base dei feedback e delle lezioni apprese.
- Acquisisci i risultati chiave in un sistema di gestione delle conoscenze e valuta eventuali modelli da aggiungere alle linee guida per gli sviluppatori o alle liste di controllo usate nella fase di pre-implementation.

Risorse

Documenti correlati:

- [Why you should develop a correction of error \(COE\) \(Perché sviluppare una correzione dell'errore\)](#)

Video correlati:

- [Amazon's approach to failing successfully](#)

- [AWS re:Invent 2021 - Amazon Builders' Library: Operational Excellence at Amazon](#)

REL12-BP03 Test dei requisiti funzionali

Utilizza tecniche come i test unitari e i test di integrazione per convalidare le funzionalità richieste.

Puoi ottenere i migliori risultati quando questi test vengono eseguiti automaticamente come parte delle operazioni di sviluppo e distribuzione. Ad esempio, utilizzando AWS CodePipeline, gli sviluppatori affidano le modifiche a un repository di origine in cui CodePipeline rileva automaticamente le modifiche. Queste modifiche vengono create e vengono eseguiti test. Una volta completati i test, il codice creato viene distribuito ai server temporaneo per il test. Dal server temporaneo, CodePipeline esegue più test, come quelli di integrazione o caricamento. Una volta completati con successo i test, CodePipeline distribuisce il codice testato e approvato alle istanze di produzione.

Inoltre, l'esperienza dimostra che i test sintetici delle transazioni (noti anche come test canary, ma da non confondere con le implementazioni canary) in grado di eseguire e simulare il comportamento dei clienti sono uno dei processi di test più importanti. Esegui questi test costantemente sugli endpoint del carico di lavoro da diverse posizioni remote. Amazon CloudWatch Synthetics ti consente di [creare "canary"](#) per monitorare gli endpoint e le API.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

- Test dei requisiti funzionali. Includono test delle unità e test di integrazione che convalidano la funzionalità richiesta.
 - [Utilizzo di CodePipeline con AWS CodeBuild per testare il codice ed eseguire compilazioni](#)
 - [AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild \(AWS CodePipeline aggiunge il supporto per i test di unità e integrazione personalizzati con AWS CodeBuild\)](#)
 - [Distribuzione continua e integrazione continua](#)
 - [Utilizzo di Canary \(Amazon CloudWatch Synthetics\)](#)
 - [Automazione e test del software](#)

Risorse

Documenti correlati:

- [Partner APN: partner che possono essere d'aiuto nell'implementazione di una pipeline di integrazione continua](#)
- [AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild \(AWS CodePipeline aggiunge il supporto per i test di unità e integrazione personalizzati con AWS CodeBuild\)](#)
- [Marketplace AWS: prodotti utilizzabili per l'integrazione continua](#)
- [Distribuzione continua e integrazione continua](#)
- [Automazione e test del software](#)
- [Utilizzo di CodePipeline con AWS CodeBuild per testare il codice ed eseguire compilazioni](#)
- [Utilizzo di Canary \(Amazon CloudWatch Synthetics\)](#)

REL12-BP04 Test dei requisiti di dimensionamento e prestazioni

Utilizza tecniche come i test di carico per convalidare che il carico di lavoro soddisfi i requisiti di dimensionamento e prestazioni.

Nel cloud, puoi creare un ambiente di test su scala di produzione on demand per il tuo carico di lavoro. Se esegui questi test su un'infrastruttura ridotta, devi dimensionare i risultati osservati in base a ciò che pensi accadrà in produzione. I test di carico e prestazioni possono essere eseguiti anche in produzione se si fa attenzione a non influire sugli utenti effettivi e si contrassegna con tag i dati di test in modo da non utilizzare dati utente reali e non danneggiare le statistiche di utilizzo o i report di produzione.

Con i test, assicurati che le risorse di base, le impostazioni di dimensionamento, le quote di servizio e la progettazione di resilienza funzionino come previsto sotto carico.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

- Test dei requisiti di dimensionamento e prestazioni. Esegui test del carico per verificare che il carico di lavoro soddisfi i requisiti di dimensionamento e prestazioni.
 - [Distributed Load Testing on AWS \(Test di carico distribuito su AWS\): simula migliaia di utenti connessi](#)

- [Apache JMeter](#)
 - Distribuisci la tua applicazione in un ambiente identico al tuo ambiente di produzione ed esegui un test di carico.
 - Utilizza un'infrastruttura come code concept per creare un ambiente il più simile possibile al tuo ambiente di produzione.

Risorse

Documenti correlati:

- [Distributed Load Testing on AWS \(Test di carico distribuito su AWS\): simula migliaia di utenti connessi](#)
- [Apache JMeter](#)

REL12-BP05 Test della resilienza tramite l'utilizzo dell'ingegneria del caos

Esegui regolarmente esperimenti di ingegneria del caos in ambienti di produzione o per quanto possibile ambienti analoghi per capire in che modo il sistema risponde a condizioni avverse.

Risultato desiderato:

La resilienza del carico di lavoro viene regolarmente verificata mediante l'applicazione dell'ingegneria del caos sotto forma di esperimenti di iniezione di errori o di inserimento di carichi imprevisti, nonché mediante il test della resilienza che convalida i comportamenti previsti noti del carico di lavoro durante un evento. Combina l'ingegneria del caos e i test della resilienza per verificare se il carico di lavoro è in grado di superare i guasti dei componenti ed eseguire il ripristino da interruzioni del servizio impreviste con un impatto minimo o nullo.

Anti-pattern comuni:

- Progettazione della resilienza, ma mancata verifica del funzionamento del carico di lavoro nel suo complesso in caso di errori.
- Mancata sperimentazione in scenari reali e con carichi previsti.
- Mancato trattamento degli esperimenti come codice o loro conservazione durante il ciclo di sviluppo.
- Mancata esecuzione degli esperimenti di ingegneria del caos sia nella pipeline CI/CD che esternamente alle implementazioni.

- Mancato utilizzo delle precedenti analisi post-incidente durante la determinazione degli errori su cui eseguire i test.

Vantaggi dell'adozione di questa best practice: l'introduzione di errori per verificare la resilienza del carico di lavoro consente di verificare che le procedure di ripristino della progettazione resiliente funzionerà se viene generato un vero e proprio errore.

Livello di rischio associato se questa best practice non fosse adottata: Medio

Guida all'implementazione

L'ingegneria del caos offre ai team la possibilità di continuare a inserire scenari di errore reali (simulazioni) in modo controllato a livello di fornitore di servizi, infrastruttura, carico di lavoro e componente con un impatto minimo o nullo per i clienti. Consente inoltre ai team di imparare dagli errori e osservare, misurare e migliorare la resilienza dei carichi di lavoro, nonché verificare l'attivazione degli avvisi e se tali avvisi vengono recapitati ai team se si verifica un evento definito.

Se applicata in modo continuativo, l'ingegneria del caos può mettere in evidenza i difetti del carico di lavoro che, se non risolti, possono avere ripercussioni negative sulla disponibilità e sulle operazioni.

Note

L'ingegneria del caos è la disciplina che sperimenta un sistema per creare fiducia nella capacità del sistema di affrontare condizioni turbolenti nella produzione. – [Principi di ingegneria del caos](#)

Se un sistema è in grado di sopportare queste interruzioni, l'esperimento di ingegneria del caos deve essere convertito in test automatico di regressione. In questo modo, gli esperimenti di ingegneria del caos devono essere eseguiti nell'ambito del ciclo di vita dello sviluppo dei sistemi (SDLC) e della pipeline CI/CD.

Per garantire che il carico di lavoro sia in grado di gestire un guasto del componente, esegui l'iniezione di eventi di errore reali durante l'esecuzione degli esperimenti. Ad esempio, esegui esperimenti relativi alla perdita di istanze Amazon EC2 o a eventi di failover delle istanze database Amazon RDS primario e quindi verifica che il carico di lavoro non sia stato compromesso oppure o che si stato interessato solo in minima parte. Utilizza una combinazione di errori dei componenti per simulare gli eventi che possono essere causati da un'interruzione del servizio in una zona di disponibilità.

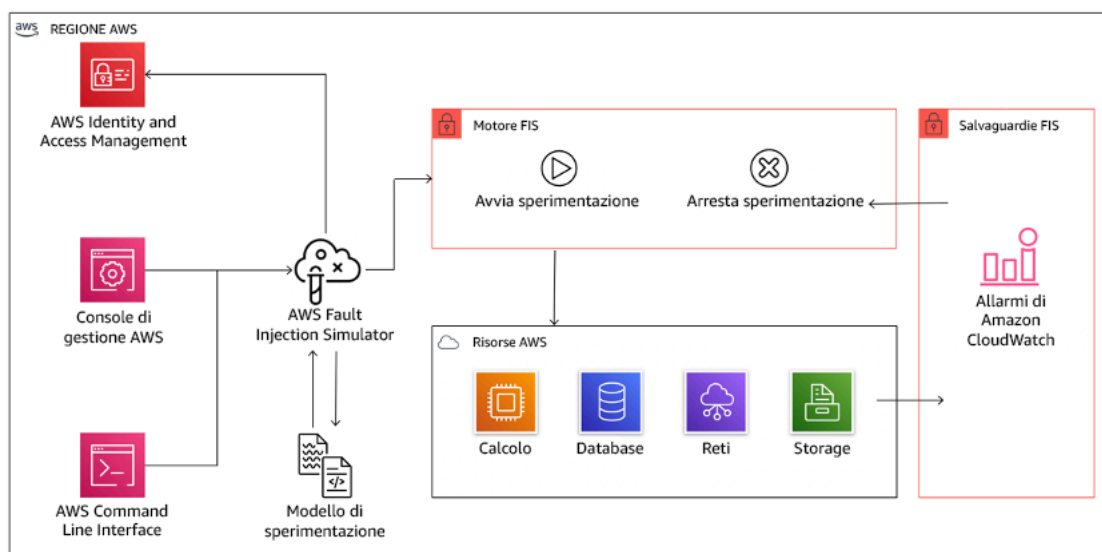
Per gli errori a livello di applicazione, ad esempio gli arresti anomali, puoi iniziare utilizzando fattori di stress, ad esempio l'esaurimento della memoria o della CPU.

Per convalidare i [meccanismi di fallback o failover](#) per le dipendenze esterne causate da interruzioni intermittenti dei servizi di rete, i componenti devono simulare tale evento bloccando l'accesso ai fornitori di terze parti per una durata specificata, che può durare da pochi secondi ad alcune ore.

Altre modalità di degrado possono causare funzionalità ridotte e risposte lente, spesso con conseguente interruzione dei servizi. Le fonti comuni di questo degrado sono una maggiore latenza nei servizi critici e una comunicazione di rete inaffidabile (pacchetti persi). Gli esperimenti basati su questi errori, inclusi gli effetti a livello di rete come latenza, messaggi eliminati ed errori DNS, possono prevedere l'incapacità di risolvere un nome, raggiungere il servizio DNS o stabilire connessioni a servizi dipendenti.

Strumenti dell'ingegneria del caos

AWS Fault Injection Service (AWS FIS) è un servizio completamente gestito per l'esecuzione di esperimenti di iniezione di errori che possono essere utilizzati come parte della pipeline di CD o al suo esterno. AWS FIS è una soluzione estremamente valida da utilizzare durante i giorni di gioco dell'ingegneria del caos. Supporta l'introduzione simultanea di errori in diversi tipi di risorse, ad esempio Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS) e Amazon RDS. Questi errori includono la cessazione delle risorse, la forzatura dei failover, l'applicazione di fattori di stress a CPU o memoria, la limitazione della lunghezza di banda della rete, la latenza e la perdita di pacchetti. Poiché è integrato con gli allarmi Amazon CloudWatch, è possibile impostare condizioni di arresto come guardrail per eseguire il rollback di un esperimento se causa un impatto inatteso.



AWS Fault Injection Service è integrato con le risorse AWS per consentire l'esecuzione di esperimenti di iniezione di errori per i carichi di lavoro.

Esistono anche diverse opzioni di terze parti per gli esperimenti di iniezione di errori. Queste includono strumenti open source, ad esempio [Chaos Toolkit](#), [Chaos Meshe](#) [Litmus Chaos](#), nonché opzioni commerciali come Gremlin. Per ampliare l'ambito degli errori che possono essere inseriti in AWS, AWS FIS [si integra con Chaos Mesh e Litmus Chaos](#) ciò consente di coordinare i flussi di lavoro relativi all'iniezione di errori tra più strumenti. Ad esempio, puoi eseguire un test di stress sulla CPU di un pod utilizzando gli errori di Chaos Mesh o Litmus Chaos durante la cessazione di una percentuale casualmente selezionata di nodi di cluster mediante le operazioni di errore di AWS FIS.

Passaggi dell'implementazione

- Determinazione degli errori da utilizzare per gli esperimenti.

Valutazione della progettazione del carico di lavoro a livello di resilienza. Tali progettazioni, create mediante le best practice del [Canone di architettura AWS](#)) giustificano i rischi in base alle dipendenze critiche, agli eventi pregressi, alle problematiche note e ai requisiti di conformità. Elenca i singoli elementi della progettazione che devono conservare la resilienza e gli errori per mitigare i quali è stata sviluppata. Per ulteriori informazioni su questi elenchi, consulta [il whitepaper relativo alla prontezza operativa](#) , contenente linee guida su come creare un processo per impedire che si verifichino di nuovo incidenti già noti. Il processo FMEA (Failure Modes and Effects Analysis) fornisce un framework per l'esecuzione di un'analisi degli errori a livello di componente e del relativo impatto sul carico di lavoro. Il processo FMEA è descritto più in dettaglio nell'articolo di Adrian Cockcroft su [modalità di errore e resilienza continua](#).

- Assegna una priorità a ogni errore.

Comincia con una categorizzazione approssimativa, ad esempio alta, media o bassa. Per assegnare la priorità, considera la frequenza dell'errore e l'impatto dell'errore sul carico di lavoro nel suo complesso.

Durante la valutazione della frequenza di un errore specifico, analizza i precedenti dati per lo stesso carico di lavoro, se disponibili. Se non sono disponibili, utilizza i dati di altri carichi di lavoro eseguiti in un ambiente simile.

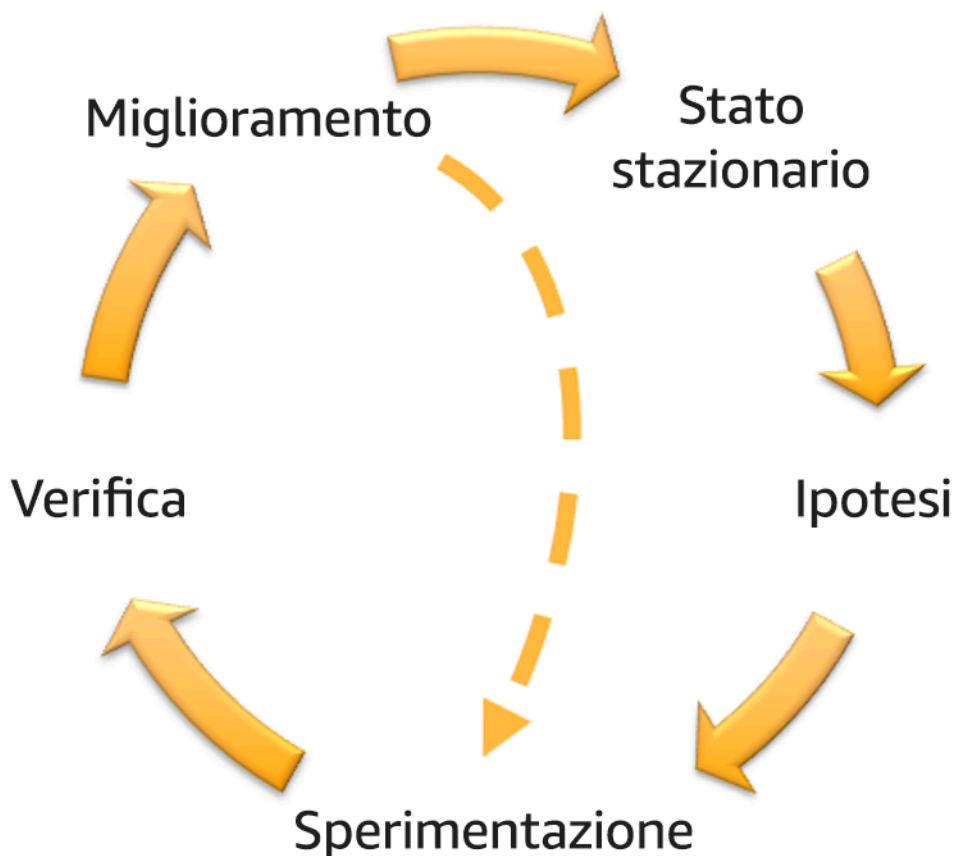
Durante la valutazione dell'impatto di un errore specifico, in genere maggiore è l'ambito dell'errore, maggiore sarà l'impatto. Considera la progettazione e lo scopo del carico di lavoro. Ad esempio, la capacità di accedere ai datastore di origine è di cruciale importanza per un carico di lavoro responsabile della trasformazione e dell'analisi dei dati. In questo caso, darai la precedenza agli

esperimenti relativi agli errori di accesso, nonché a quelli con accesso limitato a livello di larghezza di banda e inserimento di latenza.

Le analisi post-incidente rappresentano un'ottima fonte di dati per la comprensione della frequenza e dell'impatto delle modalità di errore.

Utilizza la priorità assegnata per determinare il primo errore su cui eseguire l'esperimento e l'ordine in cui sviluppare i nuovi esperimenti di iniezione di errori.

- Per ogni esperimento eseguito, attieniti ai principi del volano dell'ingegneria del caos e della resilienza continua.



Volano dell'ingegneria del caos e della resilienza continua, che utilizza il metodo scientifico di Adrian Hornsby.

- Definisci lo stato stazionario come output misurabile di un carico di lavoro che indica un comportamento normale.

Il carico di lavoro è associato allo stato stazionario se il suo funzionamento è affidabile e conforme a quanto previsto. Verifica pertanto che il carico di lavoro sia integro prima di definire lo stato stazionario. Lo stato stazionario non necessariamente indica l'assenza di impatto sul carico di lavoro se si verifica un errore in quanto una data percentuale di errori può rientrare nei limiti di valori accettabili. Lo stato stazionario rappresenta il punto di riferimento che verrà osservato durante l'esperimento e che metterà in evidenza le anomalie se le ipotesi definite nel passaggio successivo non sono conformi alle previsioni.

Ad esempio, lo stato stazionario di un sistema di pagamento può essere definito come elaborazione di 300 TPS con una percentuale di successo pari al 99% e un tempo di round trip pari a 500 ms.

- Definisci un'ipotesi in merito alle reazioni del carico di lavoro all'errore.

Un'ipotesi ottimale fa riferimento al modo in cui il carico di lavoro presumibilmente è in grado di ridurre l'impatto dell'errore e salvaguardare lo stato stazionario. Nell'ipotesi è definito che, dato un errore di un tipo specifico, il sistema o il carico di lavoro rimarrà nello stato stazionario perché la progettazione del carico di lavoro ha previsto sistemi specifici di attenuazione degli errori. Il tipo di errore specifico e i sistemi di attenuazione devono essere specificati nell'ipotesi.

Per l'ipotesi è possibile utilizzare il seguente modello, anche se è accettabile una formulazione diversa:

Note

Se si verifica un *errore specifico*, il carico di lavoro *nome del carico di lavoro* descriverà *i controlli di attenuazione* per controbilanciare *l'impatto sulle metriche aziendali o tecniche*.

Ad esempio:

- In caso di arresto del 20% dei nodi nel gruppo di nodi Amazon EKS, l'API di creazione delle transazioni continua a servire il 99° percentile delle richieste in meno di 100 ms (stato stazionario). Verrà eseguito il ripristino dei nodi Amazon EKS entro cinque minuti; i pod verranno riprogrammati ed elaboreranno il traffico entro otto minuti dall'inizio dell'esperimento. Gli avvisi verranno attivati entro tre minuti.
- Se si verifica un errore in un'istanza Amazon EC2, il controllo dell'integrità Elastic Load Balancing del sistema degli ordini farà sì che Elastic Load Balancing si limiti a inviare richieste

alle rimanenti istanze integre, mentre la funzionalità Amazon EC2 Auto Scaling sostituirà l'istanza in errore, garantendo un incremento inferiore allo 0,01% degli errori (5xx) lato server (stato stazionario).

- Se l'istanza database primario Amazon RDS restituisce un errore, il carico di lavoro della raccolta di dati della catena di approvvigionamento eseguirà il failover e si conatterà all'istanza database in standby Amazon RDS per mantenere meno di un minuto di errori di lettura o scrittura del database (stato stazionario).
- Esegui l'esperimento inserendo l'errore.

Per impostazione predefinita, un esperimento deve essere a prova di errore e tollerato dal carico di lavoro. Se sei consapevole del fatto che il carico di lavoro avrà esito negativo, non eseguire l'esperimento. L'ingegneria del caos deve essere utilizzata per individuare scenari noti sconosciuti o scenari completamente sconosciuti. "Scenari noti sconosciuti" fanno riferimento a quegli scenari di cui sei consapevole, ma non ne comprendi completamente la natura, mentre con "scenari completamente sconosciuti" si intendono quegli scenari a te non noti e di cui non ne comprendi la natura o i motivi. L'esecuzione di esperimenti su un carico di lavoro non funzionante non può fornire nuovi approfondimenti chiarificatori. L'esperimento deve infatti essere pianificato con attenzione, essere caratterizzato da un ambito ben definito relativamente al suo impatto, nonché fornire un meccanismo di rollback applicabile in caso di esiti negativi imprevisti. Se il criterio di due diligence indica che il carico di lavoro è in grado di sostenere l'esperimento, procedi ed esegui l'esperimento. Sono disponibili varie opzioni per l'inserimento degli errori. Per i carichi di lavoro in AWS, [AWS FIS](#) fornisce numerose simulazioni di errore predefinite denominate [operazioni](#). Puoi anche definire operazioni personalizzate eseguibili in AWS FIS utilizzando i [documenti AWS Systems Manager](#).

È sconsigliato l'uso di script personalizzati per gli esperimenti di ingegneria del caos, a meno che gli script non siano in grado di rilevare lo stato corrente del carico di lavoro, generare log e fornire meccanismi di rollback e condizioni di arresto, laddove possibile.

Un framework o set di strumenti efficace che supporta l'ingegneria del caos deve tenere traccia dello stato corrente di un esperimento, generare log e fornire meccanismi di rollback a supporto dell'esecuzione controllata di un esperimento. Inizia utilizzando un servizio noto, ad esempio AWS FIS, che consente di eseguire esperimenti con ambiti e meccanismi di sicurezza ben definiti in grado di eseguire il rollback dell'esperimento in caso di esiti negativi imprevisti. Per ulteriori informazioni sull'intera gamma di esperimenti che utilizzano AWS FIS, consulta anche la sezione relativa al [laboratorio relativo alle app Well-Architected resilienti con ingegneria del caos](#).

Inoltre, [AWS Resilience Hub](#) analizzerà il carico di lavoro e creerà gli esperimenti che potrai scegliere di implementare ed eseguire in AWS FIS.

Note

Per ogni esperimento, devi essere consapevole del suo ambito e del relativo impatto. È consigliabile eseguire la simulazione dell'errore in un ambiente non di produzione prima di eseguirla in un ambiente di produzione vero e proprio.

Gli esperimenti devono essere eseguiti in ambienti di produzione con un carico reale mediante [implementazioni canary](#), che attivano sistemi sperimentali e di controllo, laddove possibile. L'esecuzione degli esperimenti durante gli orari non di punta è altamente consigliata al fine di ridurre al massimo potenziali eventi negativi durante la prima esecuzione dell'esperimento negli ambienti di produzione. Inoltre, se l'utilizzo dell'effettivo traffico clienti costituisce un rischio eccessivo, puoi eseguire gli esperimenti utilizzando una sintesi del traffico nell'infrastruttura di produzione utilizzando implementazioni sperimentali e di controllo. Se l'utilizzo di un ambiente di produzione non è possibile, esegui gli esperimenti in ambienti di pre-produzione il più simili possibile agli effettivi ambienti di produzione.

Devi definire e monitorare i guardrail per essere sicuro che l'esperimento non abbia un impatto sul traffico di produzione o sugli altri sistemi che superi i limiti accettabili. Definisci condizioni di arresto per interrompere l'esperimento se viene raggiunta la soglia definita nella metrica del guardrail. In tali condizioni devono essere incluse le metriche relative allo stato stazionario del carico di lavoro e le metriche riferite ai componenti in cui inserisci l'errore. Un [monitor sintetico](#) (definito anche canary utente) è una metrica che in genere deve essere inclusa come proxy utente. [Le condizioni di arresto per AWS FIS](#) sono supportate nel modello di esperimento, nella misura di un massimo di cinque condizioni di arresto per modello.

Uno dei principi dell'ingegneria del caos prevede la riduzione dell'ambito dell'esperimento e del relativo impatto.

Se da un lato deve essere prevista la possibilità di un determinato impatto negativo a breve termine, dall'altro il contenimento e la riduzione delle conseguenze negative degli esperimenti sono una responsabilità esclusiva dell'addetto all'ingegneria del caos.

Un metodo per verificare l'ambito e il potenziale impatto prevede l'esecuzione dell'esperimento dapprima in un ambiente non di produzione, la verifica che le soglie delle condizioni di arresto

vengano attivate come previsto durante lo svolgimento di un esperimento e l'utilizzo effettivo delle misure di osservabilità finalizzate all'acquisizione di un'eccezione, anziché eseguire l'esperimento direttamente in produzione.

Durante l'esecuzione di esperimenti di iniezione di errori, verifica che tutte le parti responsabili ne siano a conoscenza. Comunica ai team appropriati, ad esempio i team responsabili delle operazioni, dell'affidabilità dei servizi e del supporto clienti, quando verranno eseguiti gli esperimenti e l'impatto previsto. Metti a disposizione di questi team strumenti di comunicazione che consentano loro di informare i responsabili dell'esperimento di eventuali effetti avversi.

È necessario ripristinare lo stato originario del carico di lavoro e dei relativi sistemi sottostanti. La progettazione resiliente del carico di lavoro è spesso caratterizzata da funzionalità di riparazione automatica. Tuttavia, alcune progettazioni difettose o alcuni esperimenti non riusciti possono compromettere in modo imprevisto lo stato del carico di lavoro. Entro la fine dell'esperimento dovrai essere consapevole di questa situazione e ripristinare il carico di lavoro e i sistemi. Con AWS FIS puoi impostare una configurazione di rollback, definita anche post-operazione, all'interno dei parametri operativi. Una post-operazione ripristina una destinazione allo stato in cui si trovava prima dell'esecuzione dell'operazione stessa. Indipendentemente dal fatto che vengano eseguite in modalità automatica, ad esempio utilizzando AWS FIS, o manuale, queste post-operazioni devono essere incluse in un playbook in cui vengono descritte le procedure di rilevamento e gestione degli errori.

- Verifica l'ipotesi.

[Principi di ingegneria del caos](#) è un documento contenente le linee guida su come verificare lo stato stazionario del carico di lavoro.

È necessario concentrarsi sull'output misurabile di un sistema e non sugli attributi interni del sistema. Le misurazioni di tale output in un breve periodo di tempo costituiscono un'attestazione dello stato stazionario del sistema. La velocità di trasmissione effettiva del sistema nel suo complesso, le percentuali di errori e i percentili della latenza possono essere considerati metriche di interesse che rappresentano il comportamento di uno stato stazionario. Sulla base dei rilevamenti dei modelli di comportamento sistematico durante gli esperimenti, l'ingegneria del caos verifica che il sistema funzioni correttamente anziché tentare di convalidare il modo in cui funziona.

Nei due esempi precedenti sono state incluse le metriche dello stato stazionario relative a un incremento inferiore allo 0,01% di errori (5xx) lato server e inferiore a un minuto di errori di lettura e scrittura del database.

Gli errori 5xx rappresentano una buona metrica perché sono la conseguenza della modalità di errore che un client del carico di lavoro sperimenterà direttamente. La misurazione degli errori del database risulta valida come conseguenza diretta dell'errore, ma deve essere supportata da una misurazione diretta dell'impatto, ad esempio le richieste cliente non riuscite o gli errori restituiti a livello di client. Includi anche un monitor sintetico, definito canary utente, in qualsiasi API o URI a cui il client del carico di lavoro ha accesso diretto.

- Migliora la progettazione del carico di lavoro con un occhio di riguardo per la resilienza.

Se lo stato stazionario non è stato preservato, analizza in che modo puoi migliorare la progettazione del flusso di lavoro per azzerare l'impatto dell'errore applicando le best practice descritte nel [Pilastro AWS Well-Architected relativo all'affidabilità](#). Ulteriori linee guida e risorse sono disponibili nella [libreria di AWS Builder](#), dove sono contenuti articoli su come [migliorare i controlli dell'integrità](#) oppure [impiegare nuovi tentativi con backoff nel codice dell'applicazione](#).

Dopo aver implementato queste modifiche, esegui di nuovo l'esperimento (rappresentato dalla linea punteggiata nel volano relativo all'ingegneria del caos) per determinare la relativa efficacia. Se nella fase di verifica risulta che l'ipotesi è vera, il carico di lavoro sarà in stato stazionario e il ciclo continuerà.

- Esegui gli esperimenti con regolarità.

Un esperimento di ingegneria del caos è un ciclo e gli esperimenti devono essere eseguiti regolarmente nell'ambito dell'ingegneria del caos. Se un carico di lavoro è conforme all'ipotesi dell'esperimento, l'esperimento deve essere automatizzato affinché venga eseguito continuamente come fase di regressione della pipeline CI/CD. Per ulteriori informazioni in merito, consulta questo blog relativamente alle [procedure di esecuzione degli esperimenti AWS FIS utilizzando AWS CodePipeline](#). Questo laboratorio relativo a esperimenti [AWS FIS ricorrenti in una pipeline CI/CD](#) ti consente di fare esperienza pratica.

Gli esperimenti di iniezione di errori fanno inoltre parte delle giornate di gioco (consulta [REL12-BP06 Esecuzione regolare di giornate di gioco](#)). Le giornate di gioco simulano un errore o un evento per verificare sistemi, processi e risposte dei team. Lo scopo è di eseguire effettivamente le azioni che compirebbe il team come se si verificasse un evento eccezionale.

- Acquisisci e archivia i risultati degli esperimenti.

I risultati degli esperimenti di iniezione di errori devono essere acquisiti e resi persistenti. Includi tutti i dati necessari, ad esempio orari, carico di lavoro e condizioni, in modo da essere in grado di analizzare i risultati e i trend in un secondo momento. I risultati potrebbero includere, ad esempio,

screenshot dei pannelli di controllo, dump in formato CSV del database delle metriche oppure appunti scritti a mano relativi a eventi e osservazioni associati all'esperimento. [La registrazione degli esperimenti mediante AWS FIS](#) può rientrare nel processo di acquisizione dei dati.

Risorse

Best practice correlate:

- [REL08-BP03 Esecuzione di test di resilienza come parte integrante dell'implementazione](#)
- [REL13-BP03 Esecuzione di test sull'implementazione del ripristino di emergenza per convalidare l'implementazione](#)

Documenti correlati:

- [What is AWS Fault Injection Service? \(Che cos'è AWS Fault Injection Service?\)](#)
- [What is AWS Resilience Hub? \(Che cos'è AWS Resilience Hub?\)](#)
- [Principi di ingegneria del caos](#)
- [Chaos Engineering: Planning your first experiment \(Ingegneria del caos: pianificazione del primo esperimento\)](#)
- [Resilience Engineering: Learning to Embrace Failure](#)
- [Chaos Engineering stories \(Storie relative all'ingegneria del caso\)](#)
- [Evitare fallback nei sistemi distribuiti](#)
- [Canary Deployment for Chaos Experiments \(Implementazione canary per gli esperimenti di ingegneria del caos\)](#)

Video correlati:

- [AWS re:Invent 2020: Testing resiliency using chaos engineering \(ARC316\) \(Esecuzione di test di resilienza mediante l'ingegneria del caos \[ARC316\]\)](#)
- [AWS re:Invent 2019: migliorare la resilienza con l'ingegneria del caos \(DOP309-R1\)](#)
- [AWS re:Invent 2019: Performing chaos engineering in a serverless world \(CMY301\) \(Esecuzione dell'ingegneria del caos in uno scenario serverless \[CMY301\]\)](#)

Esempi correlati:

- [Well-Architected lab: Level 300: Testing for Resiliency of Amazon EC2, Amazon RDS, and Amazon S3 \(Test della resilienza di Amazon EC2, Amazon RDS e Amazon S3\)](#)
- [Chaos Engineering on AWS lab \(Laboratorio relativo all'ingegneria del caos in AWS\)](#)
- [Resilient and Well-Architected Apps with Chaos Engineering lab \(Laboratorio relativo alle app Well-Architected resilienti con ingegneria del caos\)](#)
- [Serverless Chaos lab \(Laboratorio relativi a esperimenti di ingegneria del caos per architetture serverless\)](#)
- [Measure and Improve Your Application Resilience with AWS Resilience Hub lab \(Laboratorio di misurazione e ottimizzazione della resilienza dell'applicazione con AWS Resilience Hub\)](#)

Strumenti correlati:

- [AWS Fault Injection Service](#)
- Marketplace AWS: [Gremlin Chaos Engineering Platform \(Piattaforma di ingegneria del caos di Gremlin\)](#)
- [Chaos Toolkit](#)
- [Chaos Mesh](#)
- [Litmus](#)

REL12-BP06 Esecuzione regolare di giornate di gioco

Utilizza le giornate di gioco per provare regolarmente le procedure per rispondere a eventi ed errori nel modo più vicino possibile alla produzione (anche negli ambienti di produzione) con le persone che si occuperanno di eventuali scenari di errore reali. Le giornate di gioco applicano misure per garantire che gli eventi di produzione non influiscano sugli utenti.

Le giornate di gioco simulano un errore o un evento per testare sistemi, processi e risposte dei team. Lo scopo è di eseguire effettivamente le azioni che compirebbe il team come se si verificasse un evento eccezionale. Questo ti aiuta a capire dove puoi apportare dei miglioramenti e ti può aiutare a sviluppare un'esperienza organizzativa nella gestione degli eventi. Tali azioni devono essere svolte regolarmente in modo che il team costruisca una memoria muscolare su come rispondere.

Quando la progettazione per la resilienza è in loco ed è stata testata in ambienti non di produzione, un game day è il modo per garantire che tutto funzioni come pianificato in produzione. Una giornata di gioco, soprattutto la prima, è un'attività di duro lavoro per tutti, in cui tutti gli ingegneri e i team operativi vengono informati in merito a quando accadrà e cosa accadrà. I runbook sono in loco.

Gli eventi simulati, compresi i possibili eventi di guasto, vengono eseguiti nei sistemi di produzione nel modo prescritto e ne viene valutato l'impatto. Se tutti i sistemi funzionano come progettato, il rilevamento e la correzione automatica avvengono con un impatto minimo o nullo. Tuttavia, se si osserva un impatto negativo, viene eseguito il rollback del test e i problemi relativi al carico di lavoro vengono risolti, se necessario manualmente (utilizzando il runbook). Poiché le giornate di gioco hanno spesso luogo in produzione, è necessario prendere tutte le precauzioni per garantire che non vi sia alcun impatto sulla disponibilità per i clienti.

Anti-pattern comuni:

- Documentare le procedure senza mai esercitarle.
- Non includere i responsabili delle decisioni aziendali negli esercizi di test.

Vantaggi dell'adozione di questa best practice: Eseguire giornate di gioco garantisce che tutto il personale segua le policy e le procedure quando si verifica un incidente reale e convalida che tali policy e procedure siano appropriate.

Livello di rischio associato se questa best practice non fosse adottata: Medium

Guida all'implementazione

- Programma giornate di gioco per provare regolarmente i tuoi runbook e playbook. Le giornate di gioco devono coinvolgere tutte le persone implicate in un evento di produzione: proprietari di aziende, personale addetto allo sviluppo, personale operativo e team di risposta agli incidenti.
 - Esegui i test di carico o delle prestazioni e successivamente esegui l'iniezione degli errori.
 - Ricerca anomalie nei tuoi runbook e opportunità di provare i tuoi playbook.
 - In caso di deviazione dai tuoi runbook, perfeziona il runbook o correggi il comportamento. Se ti eserciti sul tuo playbook, identifica il runbook che avrebbe dovuto essere usato, oppure creane uno nuovo.

Risorse

Documenti correlati:

- [Che cos'è AWS GameDay?](#)

Video correlati:

- [AWS re:Invent 2019: migliorare la resilienza con l'ingegneria del caos \(DOP309-R1\)](#)

Esempi correlati:

- [AWS Well-Architected Labs – Test di resilienza](#)

Pianificazione per il disaster recovery (DR)

Avere backup e componenti del carico di lavoro ridondanti in loco è l'inizio della strategia di disaster recovery. [RTO e RPO sono i tuoi obiettivi](#) per il ripristino del carico di lavoro. Imposta questi valori in base alle esigenze aziendali. Implementa una strategia per raggiungere questi obiettivi, prendendo in considerazione le posizioni e la funzione delle risorse e dei dati del carico di lavoro. La probabilità di interruzione e il costo del ripristino sono fattori chiave che aiutano a comunicare il valore aziendale che può avere il ripristino di emergenza per un carico di lavoro.

Sia la disponibilità che il ripristino di emergenza si affidano alle stesse best practice, come il monitoraggio degli errori, la distribuzione su più sedi e il failover automatico. Tuttavia, la disponibilità si focalizza su componenti del carico di lavoro, mentre il ripristino di emergenza si concentra su copie singole dell'intero carico di lavoro. Il ripristino di emergenza ha obiettivi diversi rispetto alla Disponibilità e si focalizza sulle tempistiche di ripristino dopo un disastro.

Best practice

- [REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati](#)
- [REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino](#)
- [REL13-BP03 Esecuzione di test sull'implementazione del ripristino di emergenza per convalidare l'implementazione](#)
- [REL13-BP04 Gestione della deviazione di configurazione nel sito o nella Regione del ripristino di emergenza](#)
- [REL13-BP05 Automatizzazione del ripristino](#)

REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati

Il carico di lavoro ha un Recovery Time Objective (RTO) e Recovery Point Objective (RPO).

Il Recovery Time Objective (RTO) è il ritardo massimo accettabile tra l'interruzione del servizio e il suo ripristino. Questo determina ciò che viene considerato un intervallo di tempo accettabile quando il servizio non è disponibile.

Recovery Point Objective (RPO) è il periodo di tempo massimo accettabile dall'ultimo punto di ripristino dei dati. Questo determina ciò che viene considerato una perdita di dati accettabile tra l'ultimo punto di ripristino e l'interruzione del servizio.

RTO e RPO sono valori importanti quando si seleziona una strategia adeguata di ripristino di emergenza per il proprio carico di lavoro. Tali obiettivi sono stabiliti dall'azienda e poi vengono utilizzati dai team tecnici per selezionare e implementare una strategia di ripristino di emergenza.

Risultato desiderato:

Ogni carico di lavoro ha un RTO e un RPO assegnati, definiti in base all'impatto aziendale. Il carico di lavoro viene assegnato a un livello predefinito, che stabilisce la disponibilità del servizio e la perdita accettabile di dati, con un RTO e un RPO associati. Se tale livello non è raggiungibile, è possibile assegnare un livello personalizzato per carico di lavoro, con l'obiettivo di creare i livelli in un secondo momento. RTO e RPO sono valori fondamentali per la selezione di una strategia di ripristino di emergenza da implementare per il carico di lavoro. Altre riflessioni nel momento della scelta di una strategia di ripristino di emergenza sono i vincoli economici, le dipendenze del carico di lavoro e i requisiti operativi.

Per l'RTO è necessario comprendere l'impatto in base alla durata di un'interruzione. È lineare o ci sono implicazioni non lineari? (Ad esempio, dopo 4 ore, chiudi una linea di produzione fino l'inizio del turno successivo).

Una matrice di ripristino di emergenza, come quella seguente, può aiutarti a capire come la criticità del carico di lavoro sia collegata agli obiettivi di ripristino. (Da notare che i valori reali per gli assi X e Y devono essere personalizzati in base alle esigenze della tua organizzazione).

Matrice di ripristino di emergenza						
		Obiettivo del punto di ripristino				
		meno di 1 minuto	meno di 1 ora	meno di 6 ore	meno di 1 giorno	Più di 1 giorno
Obiettivo del tempo di ripristino	meno di 10 minuti	Critica	Critica	Alta	Medio	Medio
	meno di 2 ore	Critica	Alta	Medio	Medio	Bassa
	meno di 8 ore	Alta	Medio	Medio	Bassa	Bassa
	meno di 24 ore	Medio	Medio	Bassa	Bassa	Bassa
	Più di 24 ore	Medio	Bassa	Bassa	Bassa	Bassa

Figura 16: Matrice di ripristino di emergenza

Anti-pattern comuni:

- Nessun obiettivo di ripristino definito.
- Selezione di obiettivi di ripristino arbitrari.
- Selezione di obiettivi di ripristino troppo tolleranti e che non soddisfano gli obiettivi di business.
- Mancanza di comprensione dell'impatto dei tempi di inattività e perdita dei dati.
- Selezione di obiettivi di ripristino non realistici, come tempo zero di ripristino e nessuna perdita di dati, che potrebbero non essere raggiungibili per la configurazione del tuo carico di lavoro.
- Selezione di obiettivi di ripristino più severi rispetto agli obiettivi aziendali effettivi. Questo costringe a effettuare implementazioni di ripristino di emergenza più costose e complicate rispetto alle esigenze del carico di lavoro.
- Selezione di obiettivi di ripristino non compatibili con quelli di un carico di lavoro dipendente.
- I tuoi obiettivi di ripristino non considerano i requisiti di conformità normativa.
- RTO e RPO definiti per un carico di lavoro, ma mai testati.

Vantaggi dell'adozione di questa best practice: Gli obiettivi di ripristino in termini di tempo e perdita di dati sono necessari per guidare l'implementazione del disaster recovery.

Livello di rischio associato se questa best practice non fosse adottata: Alta

Guida all'implementazione

Per un dato carico di lavoro devi considerare l'impatto dei tempi di inattività e della perdita dei dati per la tua azienda. L'impatto generalmente aumenta all'aumentare dei tempi di inattività o della perdita dei dati, ma il ritmo di tale crescita cambia in base al tipo di carico di lavoro. Ad esempio, potresti tollerare l'inattività per massimo un'ora con conseguenze minime, ma successivamente l'impatto diventerebbe rapidamente più serio. L'impatto sull'azienda si manifesta in forme diverse, tra cui costi economici (come perdita di fatturato), fiducia del cliente (e impatto sulla reputazione), problematiche operative (come stipendi in ritardo o diminuzione della produttività) e rischi normativi. Usa i passaggi seguenti per comprendere questi aspetti e impostare i valori RTO e RPO per il tuo carico di lavoro.

Passaggi dell'implementazione

1. Individua gli stakeholder aziendali per questo carico di lavoro e collabora con loro per implementare questi passaggi. Gli obiettivi di ripristino di un carico di lavoro sono il frutto di una decisione aziendale. I team tecnici, quindi, lavorano con gli stakeholder aziendali e usano questi obiettivi per selezionare una strategia di ripristino di emergenza.

Note

Per i passaggi 2 e 3 puoi usare [the section called “Foglio di lavoro di implementazione”](#).

2. Raccogli le informazioni necessarie per prendere una decisione rispondendo alle domande qui di seguito.
3. Hai categorie o livelli di criticità in termini di impatto del tuo carico di lavoro nella tua organizzazione?
 - a. Se sì, assegna questo carico di lavoro a una categoria
 - b. Se no, definisci queste categorie. Crea al massimo cinque categorie e perfeziona l'intervallo del tuo Obiettivo del tempo di ripristino (RTO) per ognuna. Ecco alcune categorie di esempio: critico, alto, medio, basso. Per capire come mappare i carichi di lavoro rispetto alle categorie devi considerare se il carico di lavoro è mission-critical, importante per l'azienda o non trainante.
 - c. Imposta i valori RTO e RPO del carico di lavoro in base alla categoria. Scegli sempre una categoria più severa (RTO e RPO inferiori) rispetto ai valori grezzi calcolati in questa fase. Se ciò comporta una variazione significativa di valore non rispondente alle esigenze, prendi in considerazione la possibilità di creare una nuova categoria.
4. In base alle risposte assegna i valori RTO e RPO al carico di lavoro. Puoi farlo direttamente o assegnando il carico di lavoro a un livello predefinito di servizio.

5. Crea un documento con il piano di ripristino di emergenza (DRP) per questo carico di lavoro, che sarà parte del [piano di continuità aziendale della tua organizzazione \(BCP\)](#), in un punto accessibile al team del carico di lavoro e agli stakeholder.
 - a. Registra i valori RTO e RPO e le informazioni usate per definire questi valori. Includi la strategia utilizzata per valutare l'impatto del carico di lavoro sull'azienda.
 - b. Registra altre metriche, oltre ai valori RTO e RPO che stai monitorando o che pensi di monitorare per gli obiettivi di ripristino di emergenza.
 - c. Dopo aver creato questi valori, potrai aggiungere i dettagli della tua strategia di ripristino di emergenza e il runbook.
6. Osservando le criticità del carico di lavoro in una matrice come quella della Figura 15, puoi iniziare a stabilire livelli predefiniti di servizio per la tua organizzazione.
7. Dopo aver implementato una strategia di ripristino di emergenza (o un proof of concept per una strategia di ripristino di emergenza) secondo quanto stabilito da [the section called “REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino”](#), testa questa strategia per stabilire i valori reali di RTC (Recovery Time Capability) e di RPC (Recovery Point Capability) del carico di lavoro. Se questi valori non sono in linea con gli obiettivi target di ripristino, puoi collaborare con gli stakeholder della tua azienda per modificarli o cambiare la strategia di ripristino di emergenza in modo che possa soddisfare tali obiettivi.

Domande principali

1. Qual è il tempo massimo durante il quale il carico di lavoro può essere inattivo prima che questo abbia un impatto grave sull'attività?
 - a. Definisci il costo monetario (impatto finanziario diretto) sull'attività al minuto se il carico di lavoro è inattivo.
 - b. Considera che l'impatto non è sempre lineare. L'impatto può essere limitato all'inizio e poi aumentare rapidamente oltre un punto critico specifico.
2. Qual è la quantità massima di dati che possiamo perdere prima che questo abbia un impatto grave sull'attività?
 - a. Considera questo valore per gli archivi di dati più strategici. identifica le criticità relative ad altri archivi di dati.
 - b. I dati del carico di lavoro possono essere ricreati se persi? Se questo è operativamente più facile rispetto al backup e al ripristino, scegli il valore RPO in base alla criticità dei dati di origine utilizzati per ricreare i dati del carico di lavoro.

3. Quali sono gli obiettivi di ripristino e le aspettative di disponibilità dei carichi di lavoro da cui questo valore dipende (downstream) o i carichi di lavoro che dipendono da questo valore (upstream)?
 - a. Scegli obiettivi di ripristino che consentono a questo carico di lavoro di soddisfare i requisiti delle dipendenze upstream.
 - b. Scegli obiettivi di ripristino che sono raggiungibili considerate le funzionalità di ripristino delle dipendenze downstream. Possono essere escluse le dipendenze downstream non critiche (quelle che puoi "aggirare"). In alternativa, lavora con dipendenze downstream critiche per migliorare le funzionalità di ripristino, laddove necessario.

Domande aggiuntive

Considera queste domande e come possono essere applicate a questo carico di lavoro:

4. Hai RTO e RPO diversi a seconda del tipo di interruzione (Regione rispetto ad AZ e così via)?
5. Esiste un periodo specifico (stagionalità, eventi commerciali, lanci di prodotto) in cui RTO/RPO possono cambiare? Se sì, qual è la misurazione diversa e il vincolo temporale?
6. Se il carico di lavoro viene perturbato, quanti clienti ne subiranno l'impatto?
7. Qual è l'impatto sulla reputazione se il carico di lavoro è perturbato?
8. Quali altri impatti operativi possono verificarsi se il carico di lavoro subisce perturbazioni? Ad esempio, l'impatto sulla produttività dei dipendenti se i sistemi e-mail non sono disponibili o se i sistemi di buste paga non sono in grado di inviare le transazioni.
9. In che modo il carico di lavoro e i valori RTO e RPO si allineano alla linea di business e alla strategia di ripristino di emergenza dell'organizzazione?
10. Esistono obblighi contrattuali interni per fornire un servizio? Esistono delle penali nel caso in cui non siano soddisfatti?
11. Quali sono i limiti normativi o di conformità dei dati?

Foglio di lavoro di implementazione

Puoi usare questo foglio di lavoro per le fasi 2 e 3 dell'implementazione. Adegua questo foglio di lavoro in base alle tue esigenze specifiche, aggiungendo, ad esempio, altre domande.

Passo 2: domande principali	Si applica al carico di lavoro?	RTO del carico di lavoro	RPO del carico di lavoro	RTO rettif.	RPO rettif.	Istruzioni
[1] tempo massimo di inattività del carico di lavoro						misurato in tempo dall'inizio del malfunzionamento al ripristino
[2] quantità massima di dati che possono essere persi						misurato in tempo trascorso dall'ultimo set di dati integro ripristinabile
[3a] dipendenze a monte						inserire gli obiettivi di recupero a monte più rigorosi
[3b] riconciliazione delle dipendenze a valle						inserire gli obiettivi di recupero a valle meno rigorosi
[3a] riconciliazione delle dipendenze a monte						Se il valore a monte è inferiore ai valori attuali e il valore a valle è superiore,
[3b] riconciliazione delle dipendenze a valle						operare sulle dipendenze per riconciliare i valori e inserirli qui.
[3] dipendenze						ridurre i valori per soddisfare le dipendenze a monte o alzarli in base alle capacità delle dipendenza a valle
Passo 2: domande aggiuntive						Indicare se la domanda è pertinente. Saltarla in caso affermativo
RTO/RPO di base						Riportare qui i valori di RTO e RPO sopra indicati
[4] tipo di malfunzionamento	[] S / [] N					Inserire gli obiettivi di recupero per i tipi di evento con i requisiti più rigorosi
[5] obiettivi specifici basati sul tempo	[] S / [] N					Inserire gli obiettivi di recupero per i tempi con i requisiti più rigorosi
[6] clienti che sperimentano il disservizio	[] S / [] N					Tracciare un grafico dei clienti che sperimentano il disservizio in funzione del tempo di inattività o dei dati persi. Utilizzare tale grafico per inserire i valori massimi di RTO e RPO ammissibili in base all'impatto sui clienti
[7] impatto reputazionale	[] S / [] N					Lavorare in modo congiunto con l'azienda per determinare i massimi valori di RTO e RPO in base all'impatto sulla reputazione
[8] impatto operativo	[] S / [] N					Inserire i valori massimi di RTO e RPO sulla base dell'impatto operativo
[9] allineamento aziendale	[] S / [] N					Inserire i valori massimi di RTO e RPO per i carichi di lavoro di questo tipo in base ai requisiti LOB e organizzativi
[10] obblighi contrattuali	[] S / [] N					Inserire i valori massimi di RTO e RPO sulla base degli obblighi contrattuali
[11] conformità normativa	[] S / [] N					Inserire i valori massimi di RTO e RPO sulla base delle norme di conformità applicabili
obiettivo sulla base delle domande aggiuntive						Selezionare il valore minimo (valore più rigoroso) dalle domande 4-11 e inserirlo qui
obiettivo rettificato						Se non è possibile raggiungere gli obiettivi indicati nella riga precedente, collaborare con le parti interessate per allentare i vincoli e inserire un nuovo minimo qui.
RTO/RPO rettificato						Inserire il valore inferiore tra RPO/RTO di base e valore obiettivo rettificato
Passo 3						
Mappatura su categorie o livelli predefiniti						Regolare entrambi i valori verso il basso (requisito più rigoroso) per allinearsi al livello più vicino definito

Foglio di lavoro

Livello di impegno per il piano di implementazione: Bassa

Risorse

Best practice correlate:

- [the section called “REL09-BP04 Ripristino periodico dei dati per verificare l'integrità e i processi di backup:”](#)
- [the section called “REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino”](#)
- [the section called “REL13-BP03 Esecuzione di test sull'implementazione del ripristino di emergenza per convalidare l'implementazione”](#)

Documenti correlati:

- [AWS Architecture Blog: Disaster Recovery Series](#)
- [Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud \(whitepaper di AWS\)](#)

- [Gestire le policy di resilienza con AWS Resilience Hub](#)
- [Partner APN: partner che possono assistere con disaster recovery](#)
- [Marketplace AWS: prodotti utilizzabili per il disaster recovery](#)

Video correlati:

- [AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications \(Modelli architetturali per applicazioni attive-attive su più Regioni\) \(ARC209-R2\)](#)
- [Ripristino di emergenza di carichi di lavoro su AWS](#)

REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino

Definisci una strategia di ripristino di emergenza (DR) che soddisfi gli obiettivi di ripristino del carico di lavoro. Scegli una strategia, ad esempio backup e ripristino, standby (attivo/passivo) o attivo/attivo.

Risultato desiderato: definizione e implementazione di una strategia di ripristino di emergenza per ogni carico di lavoro che permette al carico di lavoro di realizzare gli obiettivi di ripristino di emergenza. Le strategie di ripristino di emergenza tra carichi di lavoro utilizzano modelli riutilizzabili (come strategie descritte in precedenza),

Anti-pattern comuni:

- Implementazione di procedure di ripristino incoerenti per carichi di lavoro con obiettivi di ripristino simili.
- Implementazione di una strategia di ripristino di emergenza ad-hoc quando si verifica un disastro.
- Assenza di piani per il ripristino di emergenza.
- Dipendenza dalle operazioni del piano di controllo durante il ripristino.

Vantaggi dell'adozione di questa best practice:

- L'utilizzo di strategie di ripristino definite consente di utilizzare strumenti e procedure di test comuni.
- L'uso di strategie di ripristino definite permette la condivisione delle informazioni tra team e l'implementazione del ripristino di emergenza nei carichi di lavoro di loro proprietà.

Livello di rischio associato alla mancata adozione di questa best practice: elevato Senza una strategia di ripristino di emergenza pianificata, implementata e testata, è poco probabile riuscire a raggiungere gli obiettivi di ripristino in caso di eventi disastrosi.

Guida all'implementazione

Una strategia di ripristino di emergenza si basa sulla capacità di creare il tuo carico di lavoro in un sito di ripristino se la tua sede principale non è disponibile per eseguire il carico di lavoro. Gli obiettivi di ripristino più comuni sono RTO e RPO, come discusso in [REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati](#).

Una strategia di ripristino di emergenza (DR) su più zone di disponibilità (AZ) all'interno di un singolo Regione AWS può offrire la mitigazione rispetto a eventi disastrosi come incendi, alluvioni e interruzioni gravi dell'energia. Se è un requisito implementare una protezione rispetto a un evento improbabile che impedisca al tuo carico di lavoro di poter essere eseguito in un determinato Regione AWS, puoi usare una strategia di ripristino di emergenza basata su più regioni.

Quando pianifichi una strategia di ripristino di emergenza su più regioni, devi scegliere una delle seguenti strategie. Sono elencate in ordine crescente di complessità e di costi e in ordine decrescente di RTO e RPO. Per regione di ripristino si intende una Regione AWS diversa da quella primaria usata per il carico di lavoro.

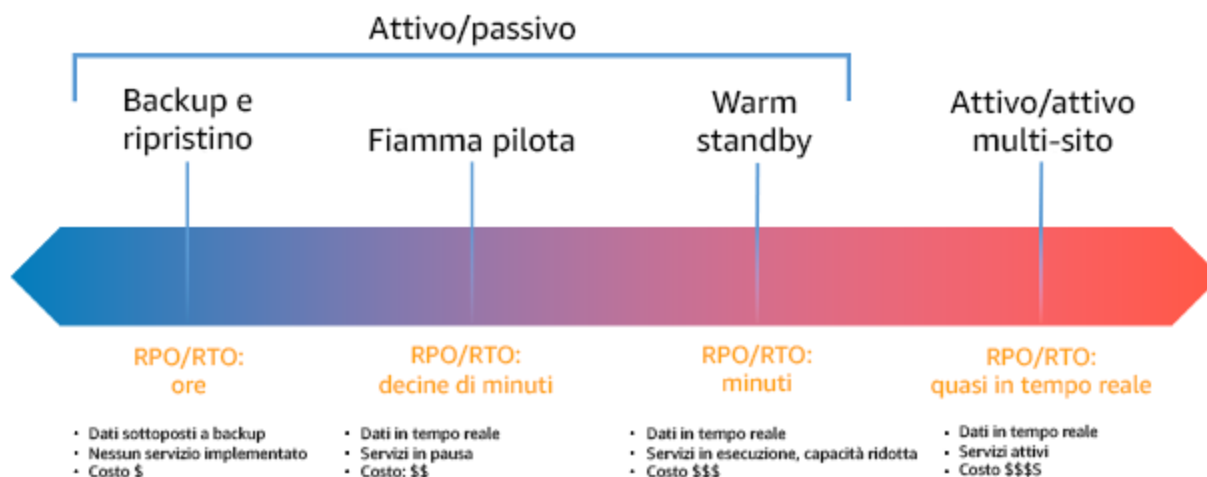


Figura 17: Strategie di ripristino di emergenza (DR)

- Backup e ripristino (RPO nell'ordine di ore, RTO in 24 ore o meno): backup dei dati e delle applicazioni nella regione di ripristino. Adottando backup continui o automatizzati otterrai un ripristino point-in-time che può ridurre il valore dell'RPO fino a raggiungere in alcuni casi 5 minuti.

Nel caso in cui si verifichi un disastro, distribuirai l'infrastruttura (usando l'infrastruttura come codice per ridurre l'RTO), distribuirai il codice e ripristinerai i dati del backup dopo un disastro nella regione di ripristino.

- **Pilot Light** (RPO nell'ordine di minuti, RTO nell'ordine di decine di minuti): provisioning di una copia dell'infrastruttura principale del carico di lavoro nella regione di ripristino. Replica i dati nella regione di ripristino e crea un backup in essa. Le risorse necessarie per supportare la replica dei dati e il backup, come database e archiviazione di oggetti, sono sempre attive. Altri elementi come i server applicativi o il calcolo serverless non vengono distribuiti, ma possono essere creati quando necessari con la configurazione e il codice applicativo richiesti.
- **Warm Standby** (RPO nell'ordine di secondi, RTO nell'ordine di minuti): esecuzione continua di una versione ridotta ma completamente funzionale del carico di lavoro nella regione di ripristino. I sistemi business critical sono completamente duplicati e sono sempre accesi, ma con un parco istanze ridimensionato. I dati vengono replicati e si trovano nella regione di ripristino. Quando viene il momento del ripristino, il sistema viene dimensionato rapidamente per gestire il carico di produzione. Maggiore è il dimensionamento nella strategia di Warm Standby, più bassi saranno l'RTO e la dipendenza del piano di controllo (control-plane). Quando il dimensionamento è completo, si parla di standby a caldo.
- **Attivo/attivo multi-regione (multisito)** (RPO quasi pari a zero, RTO potenzialmente pari a zero): il carico di lavoro viene implementato in più regioni AWS e distribuisce attivamente il traffico da più Regioni AWS. Questa strategia comporta la sincronizzazione dei dati tra le regioni. È necessario evitare o gestire possibili conflitti causati da scritture sullo stesso record in due diverse repliche regionali, un'attività che potrebbe rivelarsi complessa. La replica dei dati è utile per la sincronizzazione dei dati e ti proteggerà da alcuni tipi di disastri, ma non dalla corruzione o dalla distruzione dei dati, a meno che la tua soluzione non includa opzioni per il ripristino point-in-time.

Note

La differenza tra Pilot Light e Warm Standby può talvolta essere difficile da comprendere. Entrambe prevedono un ambiente nella tua regione di ripristino con copie degli asset della tua regione principale. La differenza è che la strategia Pilot Light non può elaborare le richieste senza aver prima intrapreso altre azioni, mentre Warm Standby può gestire immediatamente il traffico (a livelli ridotti di capacità). La strategia Pilot Light richiede l'attivazione dei server, possibilmente l'implementazione di un'infrastruttura aggiuntiva (non principale) e l'aumento di risorse, mentre Warm Standby richiede solo l'aumento di risorse (tutto è già stato implementato ed è in esecuzione). Scegli tra queste opzioni in base alle tue esigenze di RTO e RPO.

Quando i costi sono un motivo di preoccupazione e vuoi realizzare obiettivi RPO ed RTO simili a quelli definiti nella strategia di Warm Standby, puoi prendere in considerazione soluzioni native nel cloud, come AWS Elastic Disaster Recovery, che adotta l'approccio Pilot Light e offre obiettivi RPO ed RTO migliori.

Passaggi dell'implementazione

1. Definisci una strategia di ripristino di emergenza in linea con i requisiti di ripristino di questo carico di lavoro.

La scelta di una strategia di ripristino di emergenza è un compromesso tra la riduzione dei tempi di inattività e della perdita di dati (RTO ed RPO) e i costi e la complessità di implementazione della strategia. Dovresti evitare di implementare una strategia che sia più severa del necessario, in quanto questo comporterebbe costi aggiuntivi.

Ad esempio, nel diagramma seguente, l'azienda ha stabilito l'RTO massimo concesso e il limite di spesa per la strategia di ripristino del servizio. Considerati gli obiettivi dell'azienda, le strategie di ripristino di emergenza Pilot Light o di Warm Standby soddisfano sia l'RTO sia i criteri per i costi.

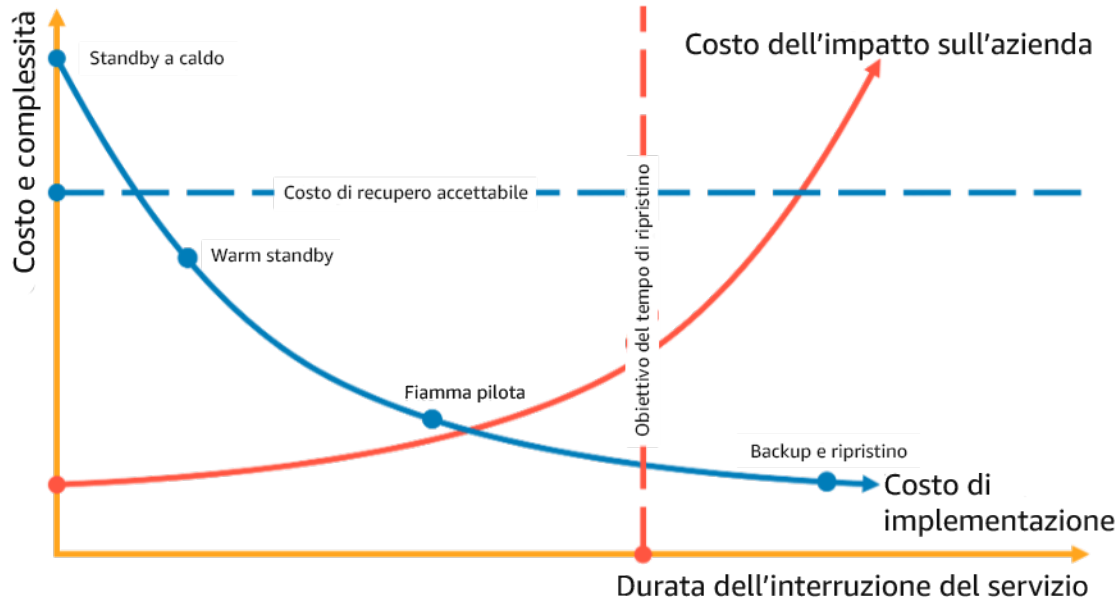


Figure 18: Scegliere una strategia di ripristino di emergenza in base all'RTO e ai costi

Per ulteriori informazioni, consulta [Piano di continuità aziendale](#).

2. Esamina i modelli con cui la strategia di ripristino di emergenza selezionata può essere implementata.

Questo passaggio consiste nel capire come implementare la strategia selezionata. Le strategie vengono spiegate con Regioni AWS come siti principali e di ripristino. Tuttavia, puoi anche decidere di utilizzare le zone di disponibilità in una singola regione come strategia di ripristino di emergenza, utilizzando aspetti di più strategie.

Nei passaggi seguenti puoi applicare la strategia al carico di lavoro specifico.

Backup e ripristino

La strategia di backup e ripristino è la meno complessa da implementare, ma richiede più tempo e impegno per il ripristino del carico di lavoro, causando un RTO e un RPO più elevati. È buona pratica creare sempre backup dei dati e copiarli in un altro sito (ad esempio, un'altra Regione AWS).

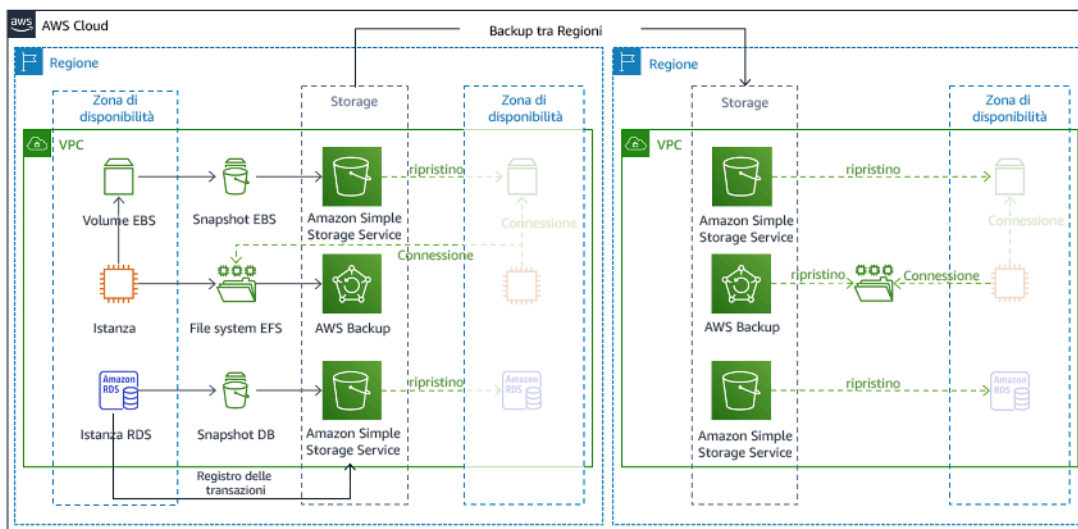


Figura 19: architettura di backup e ripristino

Per ulteriori informazioni su questa strategia, consulta [Architettura di ripristino di emergenza su AWS, parte II: backup e ripristino con recupero rapido](#).

Pilot light

Con l'approccio Pilot Light puoi replicare i dati dalla regione primaria alla regione di ripristino. Le risorse di base utilizzate per l'infrastruttura del carico di lavoro vengono distribuite nella regione di ripristino; tuttavia sono comunque necessarie risorse aggiuntive ed eventuali dipendenze per rendere

funzionale questo stack. Ad esempio, nella Figura 20 non viene implementata alcuna risorsa di calcolo.

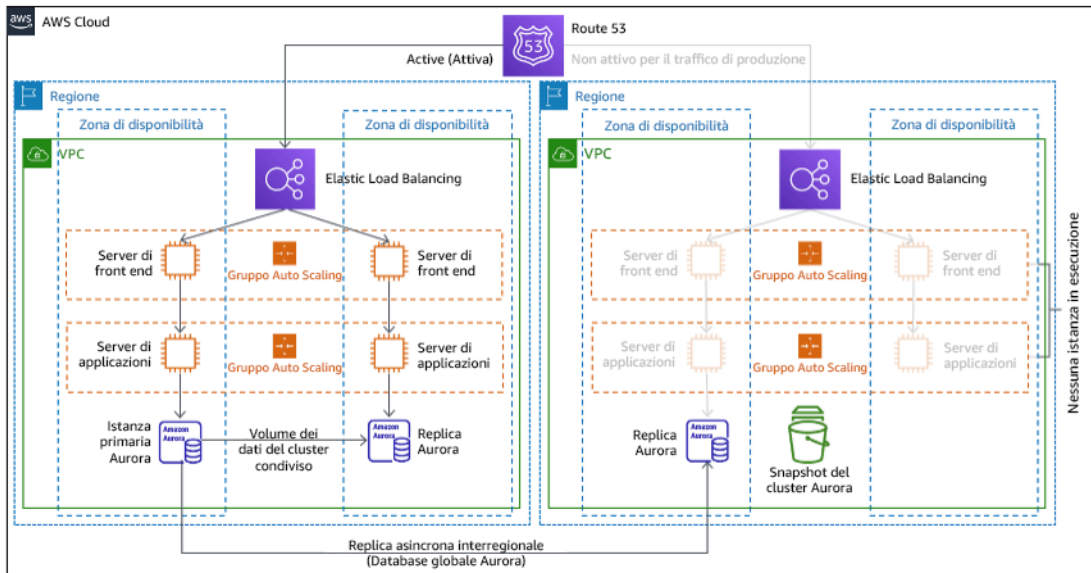


Figura 20: architettura pilot light

Per ulteriori informazioni su questa strategia, consulta [Architettura di ripristino di emergenza su AWS, parte III: Pilot Light e Warm Standby](#).

Warm standby

L'approccio Warm Standby garantisce che vi sia una copia ridotta ma completamente funzionale dell'ambiente di produzione in un'altra regione. Questo approccio estende il concetto di Pilot Light e diminuisce il tempo di ripristino, poiché il carico di lavoro è sempre attivo in un'altra regione. Se la regione di ripristino è implementata alla massima capacità, la strategia è nota come standby a caldo.

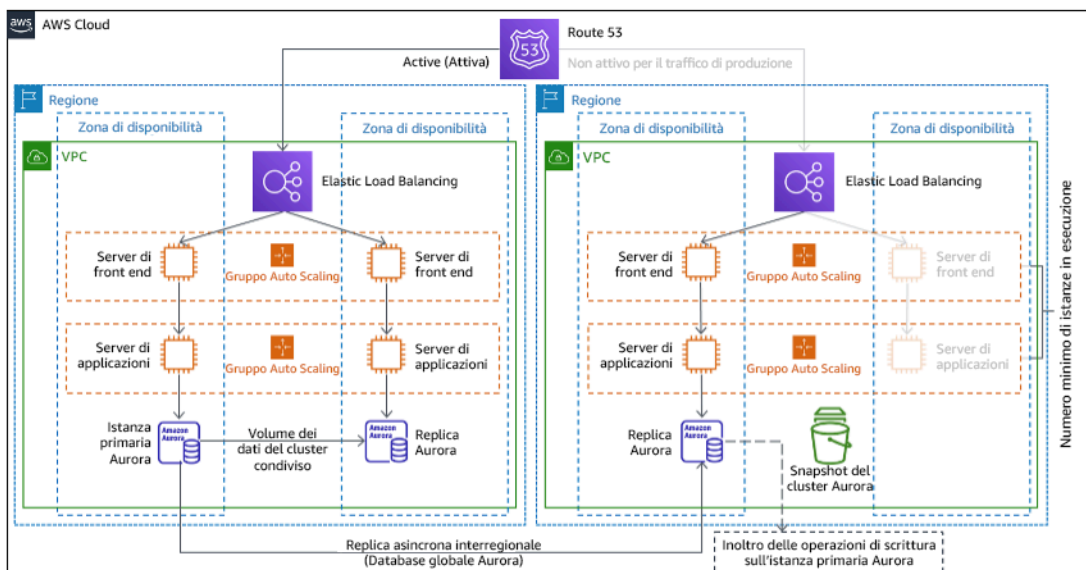


Figure 21: Architettura Warm Standby

Se si utilizza Warm Standby o Pilot Light è necessario un aumento delle risorse nella regione di ripristino. Per verificare che sia disponibile capacità sufficiente quando necessario, valuta se usare [prenotazioni di capacità](#) per istanze EC2. Se usi AWS Lambda, la [concorrenza assegnata](#) può fornire ambienti di esecuzione pronti a rispondere immediatamente alle chiamate della funzione.

Per ulteriori informazioni su questa strategia, consulta [Architettura di ripristino di emergenza su AWS, parte III: Pilot Light e Warm Standby](#).

Attivo/attivo multi-sito

Puoi eseguire il carico di lavoro simultaneamente in più regioni come parte di una strategia attivo/attivo multisito. La strategia attivo/attivo multi-sito serve il traffico da tutte le regioni in cui è distribuita. I clienti possono selezionare questa strategia per motivi diversi dal ripristino di emergenza. Può essere utilizzata per aumentare la disponibilità o nella distribuzione di un carico di lavoro a un pubblico globale (per posizionare l'endpoint più vicino agli utenti e/o per distribuire stack localizzati al pubblico di quella regione). Come strategia di ripristino di emergenza, se il carico di lavoro non può essere supportato in una delle Regioni AWS in cui viene implementato, la regione viene evacuata e vengono usate le regioni rimanenti per garantire la disponibilità. Attivo/attivo multi-sito è la strategia di ripristino operativamente più complessa e dovrebbe essere selezionata solo quando lo richiedono i requisiti aziendali.

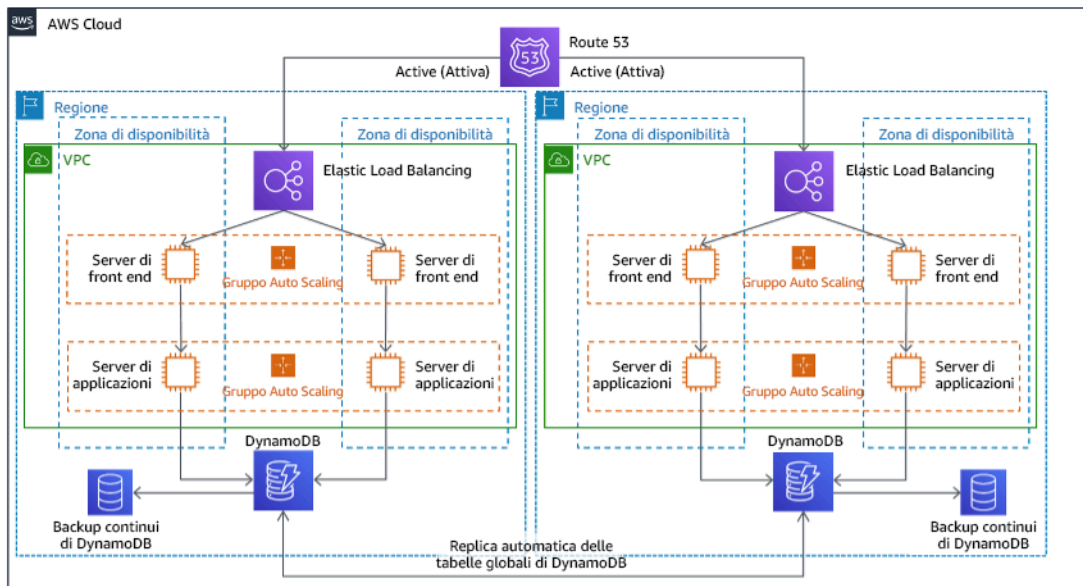


Figura 22: Architettura attivo/attivo multi-sito

Per ulteriori informazioni su questa strategia, consulta [Architettura di ripristino di emergenza su AWS, parte IV: attivo/attivo multisito](#).

AWS Elastic Disaster Recovery

Se stai prendendo in considerazione la strategia Pilot Light o di Warm Standby per il ripristino di emergenza, AWS Elastic Disaster Recovery può fornire un approccio alternativo con vantaggi ancora migliori. Elastic Disaster Recovery può offrire obiettivi RPO e RTO simili al Warm Standby, ma mantenendo l'approccio a basso costo della strategia Pilot Light. Elastic Disaster Recovery replica i dati dalla regione primaria alla regione di ripristino, usando una protezione continua dei dati per realizzare un RPO misurato in secondi e un RTO che può essere misurato in minuti. Solo le risorse necessarie per replicare i dati vengono implementate nella regione di ripristino, mantenendo i costi ridotti come nella strategia Pilot Light. Quando usi Elastic Disaster Recovery, il servizio coordina e orchestra il ripristino delle risorse di calcolo quando viene avviato come parte di un failover o di un'esercitazione.

Architettura generale di Ripristino di emergenza elastico AWS (AWS DRS)

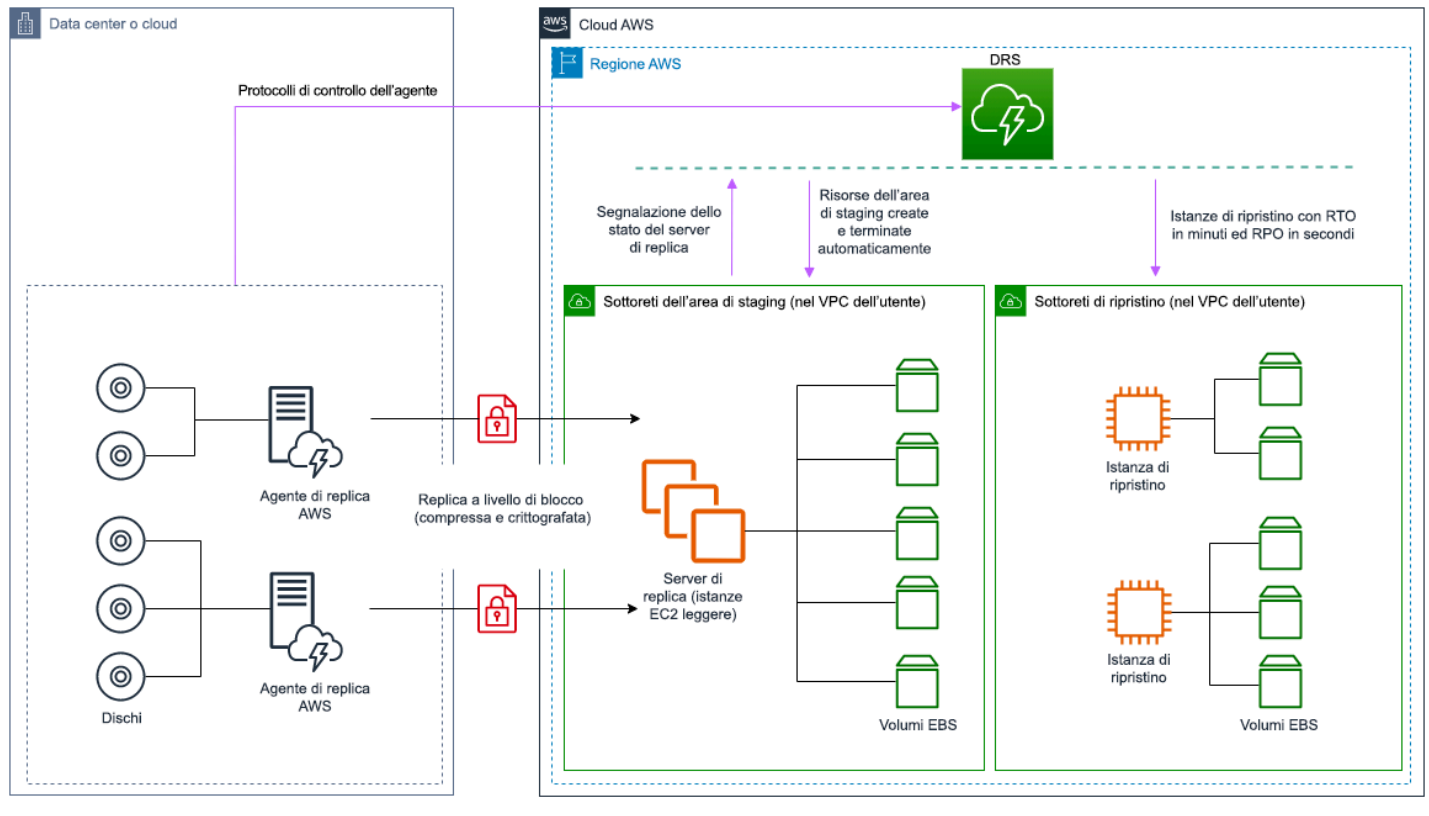


Figura 23: Architettura di AWS Elastic Disaster Recovery

Procedure aggiuntive per la protezione dei dati

Con tutte le strategie devi anche mitigare un disastro relativo ai dati. La replica continua dei dati ti proteggerà da alcuni tipi di disastri, ma non dalla corruzione o dalla distruzione dei dati, a meno che la tua soluzione non includa opzioni per il ripristino point-in-time o il controllo delle versioni dei dati archiviati. Devi anche creare un backup dei dati replicati nel sito di ripristino per creare backup point-in-time in aggiunta alle repliche.

Uso di più zone di disponibilità in una singola Regione AWS

Quando si usano più zone di disponibilità all'interno di un'unica regione, l'implementazione della strategia di ripristino di emergenza usa più elementi delle strategie precedenti. Devi innanzitutto creare un'architettura con disponibilità elevata usando più zone di disponibilità, come mostrato nella Figura 23. Questa architettura usa un approccio attivo/attivo multisito, in quanto le [istanze Amazon](#)

[EC2](#) e l'[Elastic Load Balancer](#) hanno risorse implementate in più zone di disponibilità per la gestione attiva delle richieste. L'architettura presenta anche la strategia di standby a caldo, in cui se l'istanza [Amazon RDS](#) primaria (o la zona di disponibilità stessa) restituisce un errore, l'istanza in standby viene promossa a istanza primaria.

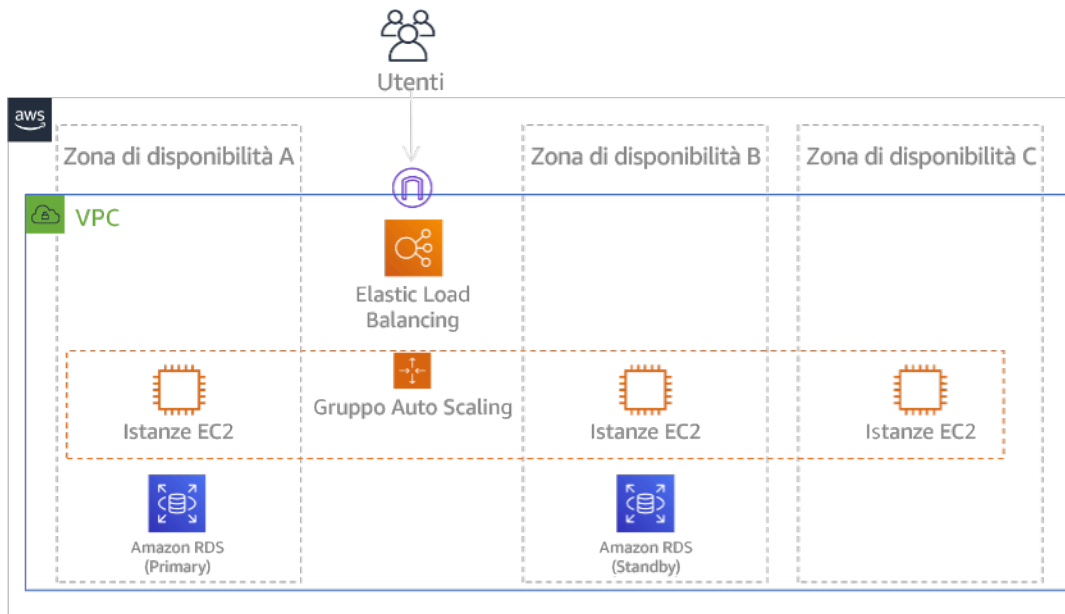


Figura 24: Architettura multi-AZ

Oltre a questa architettura HA, devi aggiungere i backup di tutti i dati richiesti per eseguire il tuo carico di lavoro. Questo aspetto è particolarmente importante per i dati limitati a un'unica zona come i [volumi Amazon EBS](#) o i [cluster Amazon Redshift](#). Se fallisce una zona di disponibilità, dovrai ripristinare i dati in un'altra zona di disponibilità. Laddove possibile, devi anche copiare i backup di dati su un'altra Regione AWS come livello di protezione aggiuntivo.

Un approccio alternativo meno comune alla singola regione, ovvero il ripristino di emergenza multi-AZ, viene descritto nel post di blog [Creazione di applicazioni altamente resilienti usando il Sistema di controllo Amazon Route 53 per il ripristino di applicazioni, parte 1: stack a regione singola](#). In questo caso la strategia adottata è quella di garantire il più possibile l'isolamento tra le zone di disponibilità, ossia come le regioni operano. Usando questa strategia alternativa puoi scegliere un approccio attivo/attivo o attivo/passivo.

Note

Alcuni carichi di lavoro hanno requisiti normativi di residenza dei dati. Se questo si applica a un carico di lavoro in una località che attualmente ha solo una Regione AWS, la multi-

regione non soddisferà i requisiti aziendali. Le strategie con più zone di disponibilità offrono una buona protezione dalla maggior parte dei disastri.

3. Valuta le risorse del tuo carico di lavoro e quale sarà la loro configurazione nella regione di ripristino prima del failover (durante la normale operatività).

Per l'infrastruttura e le risorse AWS, usa una soluzione Infrastruttura come codice (IaC), come [AWS CloudFormation](#) o strumenti di terze parti come Hashicorp Terraform. Per un'implementazione tra più account e regioni con un'unica operazione, puoi usare [AWS CloudFormation StackSets](#). Per le strategie multi-sito attivo/attivo e standby a caldo, l'infrastruttura distribuita nella tua regione di ripristino ha le stesse risorse della regione principale. Per le strategie Pilot Light e Warm Standby l'infrastruttura distribuita richiederà azioni aggiuntive per essere pronta per la produzione. Usando [parametri](#) e [logica condizionale](#) in CloudFormation, puoi controllare se uno stack implementato sia attivo o in standby con [un unico modello](#). Quando usi Elastic Disaster Recovery, il servizio replica e orchestra il ripristino delle configurazioni delle applicazioni e delle risorse di calcolo.

Tutte le strategie di ripristino di emergenza richiedono il backup delle origini dati all'interno della Regione AWS e i backup vengono quindi copiati nella regione di ripristino. [AWS Backup](#) offre una visualizzazione centralizzata in cui puoi configurare, pianificare e monitorare i backup per queste risorse. Per gli approcci Pilot Light, di Warm Standby e attivo/attivo multisito, devi anche replicare i dati dalla regione primaria alle risorse di dati nella regione di ripristino, come [Amazon Relational Database Service \(Amazon RDS\)](#), [istanze di database o tabelle Amazon DynamoDB](#). Queste risorse di dati sono pertanto attive e pronte per servire le richieste nella regione di ripristino.

Per ulteriori informazioni sul funzionamento dei servizi AWS tra regioni, consulta questa serie di blog sulla [creazione di un'applicazione in più regioni con servizi AWS](#).

4. Stabilisci e implementa le modalità con cui preparerai la tua regione al failover nel momento in cui sarà necessario (durante un evento disastroso).

Per la strategia attivo/attivo multisito, il failover significa evacuare una regione e usare le regioni attive rimanenti. In generale, tali regioni sono pronte per accettare il traffico. Per le strategie Pilot Light e di Warm Standby, le azioni di ripristino devono implementare le risorse mancanti, come le istanze EC2 nella Figura 20, insieme a risorse mancanti di altro tipo.

Per tutte le strategie precedenti potresti dover promuovere istanze di database i sola lettura a istanze di lettura/scrittura principali.

Per il backup e il ripristino, il ripristino dei dati dai backup crea risorse per tali dati, come volumi EBS, istanze DB RDS e tabelle DynamoDB. Devi anche ripristinare l'infrastruttura e distribuire il codice. Puoi usare AWS Backup per ripristinare i dati nella regione di ripristino. Consulta [REL09-BP01 Identificazione e backup di tutti i dati che richiedono un backup o riproduzione dei dati dalle origini](#) per ulteriori dettagli. La ricreazione dell'infrastruttura include la creazione di risorse come le istanze EC2, insieme a [Amazon Virtual Private Cloud \(Amazon VPC\)](#), alle sottoreti e ai gruppi di sicurezza necessari. Puoi automatizzare gran parte del processo di ripristino. Per informazioni su come fare, consulta [questo post di blog](#).

5. Stabilisci e implementa le modalità con cui reindirizzerai il traffico al failover nel momento in cui sarà necessario (durante un evento disastroso).

Questa operazione di failover può essere avviata automaticamente o manualmente. Il failover avviato automaticamente in base a controlli dell'integrità o allarmi deve essere usato con attenzione, poiché un failover non necessario (falso allarme) comporta dei costi in termini di non disponibilità e perdita dei dati. Pertanto si usa spesso il failover avviato manualmente. In questo caso, devi comunque automatizzare i passaggi del failover, in modo che l'avvio manuale si limiti al clic su un pulsante.

Esistono diverse opzioni di gestione del traffico da considerare quando si usano i servizi AWS. Un'opzione consiste nell'usare [Amazon Route 53](#). Con Amazon Route 53 puoi associare più endpoint IP in una o più Regioni AWS con un nome di dominio Route 53. Per implementare un failover avviato manualmente, puoi usare il [Sistema di controllo Amazon Route 53 per il ripristino di applicazioni](#), che fornisce un'API del piano dati a disponibilità elevata per reinstradare il traffico nella regione di ripristino. Nella fase di implementazione del failover, usa le operazioni di piano dati ed evita quelle del piano di controllo come descritto in [REL11-BP04 Fare affidamento al piano dati invece che al piano di controllo durante il ripristino](#).

Per ulteriori informazioni su questa e altre opzioni, consulta [questa sezione del whitepaper sul ripristino di emergenza](#).

6. Progetta un piano per il failback del carico di lavoro.

Si parla di failback quando un'operazione del carico di lavoro torna alla regione principale, dopo che un evento disastroso è diminuito di intensità. Il provisioning di infrastruttura e codice alla regione principale in genere segue gli stessi passaggi usati inizialmente, affidandosi all'infrastruttura come codice e alle pipeline di distribuzione del codice. La sfida del failback è il ripristino dei data store e la garanzia della loro coerenza con la regione di ripristino attiva.

Nello stato di failover i database nella regione di ripristino sono attivi e hanno dati aggiornati. L'obiettivo è eseguire una nuova sincronizzazione tra la regione di ripristino e la regione principale, per garantire il suo aggiornamento.

Alcuni servizi AWS eseguono questa operazione in automatico. Se quando usi [tabelle globali Amazon DynamoDB](#) la tabella nella regione primaria diventa non disponibile, quando torna online DynamoDB riprende la propagazione delle scritture in sospeso. Se usi il [Database globale Amazon Aurora](#) e un [failover pianificato gestito](#), viene mantenuta la topologia di replica esistente del Database globale Aurora. Pertanto, l'istanza precedente in lettura/scrittura nella regione principale diventa una replica e riceve gli aggiornamenti dalla regione di ripristino.

Nei casi in cui questo non è automatico devi ristabilire il database nella regione principale come replica del database nella regione di ripristino. In molti casi questo comporterà l'eliminazione del database principale precedente e la creazione di nuove repliche. Ad esempio, per istruzioni su come fare usando il Database globale Amazon Aurora presupponendo un failover non pianificato, consulta questo lab: [Failback di un database globale](#).

Dopo un failover, se puoi proseguire l'esecuzione nella tua regione di ripristino, valuta la possibilità di farlo nella tua regione principale. Compieresti comunque tutte le operazioni precedenti per trasformare la precedente regione principale in una regione di ripristino. Alcune organizzazioni eseguono una rotazione pianificata, scambiando periodicamente le regioni principale e di ripristino (ad esempio, ogni tre mesi).

Tutti i passaggi richiesti per failover e failback devono essere inseriti in un playbook disponibile a tutti i membri del team, sottoposto periodicamente a revisione.

Quando usi Elastic Disaster Recovery, il servizio fornirà assistenza per l'orchestrazione e l'automazione del processo di failback. Per ulteriori informazioni, consulta [Esecuzione di un failback](#).

Livello di impegno per il piano di implementazione: elevato

Risorse

Best practice correlate:

- [the section called “REL09-BP01 Identificazione e backup di tutti i dati che richiedono un backup o riproduzione dei dati dalle origini”](#)
- [the section called “REL11-BP04 Fare affidamento al piano dati invece che al piano di controllo durante il ripristino”](#)

- [the section called “REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati”](#)

Documenti correlati:

- [AWS Architecture Blog: serie sul ripristino di emergenza](#)
- [Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud \(whitepaper AWS\)](#)
- [Opzioni di ripristino di emergenza nel cloud](#)
- [Build a serverless multi-region, active-active backend solution in an hour](#)
- [Multi-region serverless backend — reloaded](#)
- [RDS: creazione di una replica di lettura in una regione AWS diversa](#)
- [Route 53: configurazione del failover DNS](#)
- [S3: replica tra regioni](#)
- [Che cos'è AWS Backup?](#)
- [Che cos'è il Sistema di controllo Route 53 per il ripristino di applicazioni?](#)
- [Ripristino di emergenza elastico AWS](#)
- [HashiCorp Terraform: Get Started - AWS](#)
- [Partner APN: partner che possono assistere con disaster recovery](#)
- [Marketplace AWS: prodotti che possono essere usati per il ripristino di emergenza](#)

Video correlati:

- [Ripristino di emergenza per carichi di lavoro su AWS](#)
- [AWS re:Invent 2018: Modelli architetturali per applicazioni attivo/attivo multi-regione \(ARC209-R2\)](#)
- [Nozioni di base sul ripristino di emergenza elastico AWS | Amazon Web Services](#)

Esempi correlati:

- [Well-Architected Lab: Ripristino di emergenza](#) – Serie di workshop che descrivono le strategie di ripristino di emergenza

REL13-BP03 Esecuzione di test sull'implementazione del ripristino di emergenza per convalidare l'implementazione

Testa regolarmente il failover nel sito di ripristino per verificare che funzioni correttamente e che sia possibile soddisfare l'RTO e l'RPO.

Anti-pattern comuni:

- Non eseguire mai failover di prova in produzione.

Vantaggi dell'adozione di questa best practice: l'esecuzione regolare di test del piano di ripristino di emergenza permette di verificare che funzionerà quando necessario e che il team è in grado di eseguire la strategia.

Livello di rischio associato alla mancata adozione di questa best practice: elevato

Guida all'implementazione

Un modello da evitare è lo sviluppo di percorsi di ripristino eseguiti raramente. Ad esempio, è possibile che si disponga di un archivio dati secondario utilizzato per query di sola lettura. Quando scrivi in un archivio dati e quello principale ha un guasto, puoi eseguire il failover verso l'archivio dati secondario. Se non testi frequentemente questo failover, è possibile che i presupposti relativi alle funzionalità dell'archivio dati secondario non siano corretti. La capacità dell'archivio dati secondario, che potrebbe essere stata sufficiente durante l'ultimo test, potrebbe non essere più in grado di tollerare il carico in questo scenario. La nostra esperienza ha dimostrato che l'unico ripristino da errore che funziona è il percorso sottoposto a frequenti test. Per questo è preferibile avere un numero ridotto di percorsi di ripristino. Puoi stabilire dei modelli di ripristino e testarli regolarmente. Se disponi di un percorso di ripristino complesso o critico, devi comunque riprodurre regolarmente il guasto specifico in produzione per convincerti che il percorso di ripristino funzioni. Nell'esempio appena discusso, è necessario eseguire il failover regolarmente in standby, indipendentemente dalle necessità.

Passaggi dell'implementazione

1. Progetta i carichi di lavoro per il ripristino. Esegui regolarmente test dei tuoi percorsi di ripristino. Il calcolo orientato al ripristino identifica le caratteristiche nei sistemi che migliorano il ripristino: isolamento e ridondanza, ripristino a livello di sistema dello stato precedente rispetto alle modifiche, capacità di fornire diagnostica, ripristino automatico, progettazione modulare e

possibilità di riavvio. Prova il percorso di ripristino per verificare di poter completare il ripristino nel tempo specificato e in base allo stato specificato. Usa i tuoi runbook durante questo ripristino per documentare i problemi e trovare le loro soluzioni prima del test successivo.

2. Per carichi di lavoro basati su Amazon EC2, usa [AWS Elastic Disaster Recovery](#) per implementare e avviare istanze di prova per la strategia di ripristino di emergenza. AWS Elastic Disaster Recovery permette di eseguire esercitazioni in modo efficiente, semplificando la preparazione a un evento di failover. Puoi anche avviare spesso le istanze usando Elastic Disaster Recovery per scopi di test ed esercitazione senza reindirizzare il traffico.

Risorse

Documenti correlati:

- [Partner APN: partner che possono assistere con disaster recovery](#)
- [AWS Architecture Blog: serie sul ripristino di emergenza](#)
- [Marketplace AWS: prodotti che possono essere usati per il ripristino di emergenza](#)
- [AWS Elastic Disaster Recovery](#)
- [Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud \(whitepaper AWS\)](#)
- [AWS Elastic Disaster Recovery – Preparazione per il failover](#)
- [Il progetto di informatica orientata al ripristino Berkeley/Stanford](#)
- [Che cos'è il Simulatore di iniezione guasti AWS?](#)

Video correlati:

- [AWS re:Invent 2018: Modelli architetturali per applicazioni attivo/attivo multi-regione](#)
- [AWS re:Invent 2019: Backup-e ripristino e soluzioni di ripristino di emergenza con AWS](#)

Esempi correlati:

- [Well-Architected Lab – Esecuzione di test per la resilienza](#)

REL13-BP04 Gestione della deviazione di configurazione nel sito o nella Regione del ripristino di emergenza

Assicurati che l'infrastruttura, i dati e la configurazione soddisfino le esigenze del sito o nella Regione del ripristino di emergenza. Ad esempio, controlla che le AMI e le quote di servizio siano aggiornate.

AWS Config monitora e registra in modo continuo le configurazioni delle risorse AWS. È in grado di rilevare le deviazioni e attivare [AWS Systems Manager Automation](#) per risolverle e attivare allarmi. AWS CloudFormation è inoltre in grado di rilevare le deviazioni negli stack distribuiti.

Anti-pattern comuni:

- Non eseguire aggiornamenti nelle sedi di ripristino, quando esegui modifiche di configurazione o di infrastruttura nelle tue sedi principali.
- Ignorare le limitazioni potenziali (ad esempio le differenze di servizio) nelle sedi di disaster recovery e principali.

Vantaggi dell'adozione di questa best practice: Assicurarsi che l'ambiente di disaster recovery sia coerente con quello esistente garantisce il ripristino completo.

Livello di rischio associato se questa best practice non fosse adottata: Medio

Guida all'implementazione

- Assicurati che le tue pipeline di distribuzione riforniscano sia i siti principali che di backup. Le pipeline per la distribuzione di applicazioni in produzione devono essere distribuite in tutte le posizioni della strategia di disaster recovery specificate, inclusi gli ambienti di sviluppo e test.
- Abilitazione di AWS Config per monitorare le potenziali posizioni di deviazione. Utilizza le regole AWS Config per creare sistemi in grado di applicare le strategie di disaster recovery e generare avvisi quando rilevano una deviazione.
 - [Correzione di risorse AWS non conformi in base alle regole di Regole di AWS Config](#)
 - [AWS Systems Manager Automation](#)
- Utilizza AWS CloudFormation per distribuire la tua infrastruttura. AWS CloudFormation è in grado di rilevare le deviazioni tra ciò che i modelli di CloudFormation specificano e ciò che viene effettivamente distribuito.
 - [AWS CloudFormation: rilevamento delle deviazioni su un intero stack CloudFormation](#)

Risorse

Documenti correlati:

- [Partner APN: partner che possono assistere con disaster recovery](#)
- [AWS Architecture Blog: Disaster Recovery Series](#)
- [AWS CloudFormation: rilevamento delle deviazioni su un intero stack CloudFormation](#)
- [Marketplace AWS: prodotti utilizzabili per il disaster recovery](#)
- [AWS Systems Manager Automation](#)
- [Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud \(whitepaper di AWS\)](#)
- [In che modo è possibile implementare una soluzione di gestione della configurazione dell'infrastruttura in AWS?](#)
- [Correzione di risorse AWS non conformi in base alle regole di Regole di AWS Config](#)

Video correlati:

- [AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications \(Modelli di architettura per applicazioni attive-attive multiregione\) \(ARC209-R2\)](#)

REL13-BP05 Automatizzazione del ripristino

Utilizza AWS o strumenti di terze parti per automatizzare il ripristino del sistema e instradare il traffico verso il sito o la Regione del ripristino di emergenza.

In base ai controlli di integrità configurati, i servizi AWS, come Elastic Load Balancing e AWS Auto Scaling, possono distribuire il carico a zone di disponibilità integre, mentre i servizi, come Amazon Route 53 e AWS Global Accelerator, instradano il carico a Regioni AWS integre. Amazon Route 53 Application Recovery Controller aiuta a gestire e coordinare il failover utilizzando i controlli di disponibilità e le funzionalità di controlli di routing. Queste funzionalità monitorano continuamente la capacità dell'applicazione di riprendersi dai guasti e permettono di controllarne il ripristino delle applicazioni su più Regioni AWS, zone di disponibilità e on-premise.

Per i carichi di lavoro su data center fisici o virtuali o cloud privati, [Ripristino di emergenza elastico AWS](#), disponibile tramite Marketplace AWS, consente alle organizzazioni di organizzare una strategia di ripristino di emergenza su AWS. CloudEndure supporta, inoltre, il ripristino di emergenza tra Regioni e zone di disponibilità in AWS.

Anti-pattern comuni:

- L'implementazione di failover e failback automatici identici può causare flapping quando si verifica un errore.

Vantaggi dell'adozione di questa best practice: Il ripristino automatico riduce i tempi di ripristino eliminando la possibilità di errori manuali.

Livello di rischio associato se questa best practice non fosse adottata: Medio

Guida all'implementazione

- Automatizzazione dei percorsi di ripristino. Per tempi di ripristino brevi, non è possibile servirsi del giudizio umano e dell'azione per scenari di disponibilità elevata. Il sistema dovrebbe ripristinarsi automaticamente in ogni situazione.
- Usa il ripristino di emergenza CloudEndure per failover e failback automatizzati. Il ripristino di emergenza CloudEndure replica in modo continuo le macchine (tra cui sistema operativo, configurazione dello stato del sistema, database, applicazioni e file) in un'area di gestione temporanea a basso costo nell'Account AWS di destinazione e nella Regione preferita. In caso di emergenza, è possibile indicare a CloudEndure Disaster Recovery di avviare automaticamente migliaia di macchine nello stato di provisioning completo in pochi minuti.
 - [Performing a Disaster Recovery Failover and Failback](#)
 - [CloudEndure Disaster Recovery](#)

Risorse

Documenti correlati:

- [Partner APN: partner che possono assistere con disaster recovery](#)
- [AWS Architecture Blog: Disaster Recovery Series](#)
- [Marketplace AWS: prodotti utilizzabili per il disaster recovery](#)
- [AWS Systems Manager Automation](#)
- [Ripristino di emergenza CloudEndure in AWS](#)
- [Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud \(whitepaper di AWS\)](#)

Video correlati:

- [AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications \(Modelli architetturali per applicazioni attive-attive su più Regioni\) \(ARC209-R2\)](#)

Esempi di implementazioni per obiettivi di disponibilità

In questa sezione, esamineremo i progetti del carico di lavoro utilizzando la distribuzione di una tipica applicazione Web che consiste in un reverse proxy, contenuto statico su Amazon S3, un server dell'applicazione e un database SQL per l'archiviazione permanente dei dati. Per ogni target di disponibilità, forniremo un'implementazione di esempio. Questo carico di lavoro potrebbe invece utilizzare container o AWS Lambda per l'elaborazione e NoSQL (ad esempio Amazon DynamoDB) per il database, ma gli approcci sono simili. In ogni scenario, illustriamo come raggiungere gli obiettivi di disponibilità attraverso la progettazione del carico di lavoro per questi argomenti:

Argomento	Per ulteriori informazioni, consulta questa sezione
Monitoraggio delle risorse	Monitoraggio delle risorse del carico di lavoro
Adattarsi alle modifiche nella domanda	Progettazione di un carico di lavoro in grado di adattarsi ai cambiamenti della domanda
Implementazione della modifica	Implementazione della modifica
Eeguire il backup dei dati	Eeguire il backup dei dati
Architettura mirata alla resilienza	Utilizzo dell'isolamento dei guasti per proteggere e il carico di lavoro Progettazione di un carico di lavoro resistente agli errori dei componenti
Testare la resilienza	Test dell'affidabilità
Pianificazione per il disaster recovery (DR)	Pianificazione per il disaster recovery (DR)

Selezione delle dipendenze

Abbiamo scelto di utilizzare Amazon EC2 per le nostre applicazioni. Mostriamo come l'utilizzo di Amazon RDS e di più zone di disponibilità migliora la disponibilità delle nostre applicazioni. Utilizzeremo Amazon Route 53 per il DNS. Quando utilizziamo più zone di disponibilità, utilizzeremo

Elastic Load Balancing. Amazon S3 viene utilizzato per i backup e i contenuti statici. Dal momento che progettiamo per raggiungere un'affidabilità più elevata, dobbiamo utilizzare servizi dotati anch'essi di una disponibilità più elevata.

Scenari a regione singola

Argomenti

- [Scenario a due 9 \(99%\)](#)
- [Scenario a tre 9 \(99,9%\)](#)
- [Scenario a quattro 9 \(99,99%\)](#)

Scenario a due 9 (99%)

Questi carichi di lavoro sono utili per l'azienda, ma se non sono disponibili rappresentano solo un inconveniente. Questo tipo di carico di lavoro può essere strumentazione interna, gestione interna delle conoscenze o monitoraggio dei progetti. Oppure può trattarsi di carichi di lavoro effettivi rivolti al cliente ma serviti da un servizio sperimentale, con un interruttore funzionale in grado di nascondere il servizio, se necessario.

Queste applicazioni possono essere distribuite con una regione e una zona di disponibilità.

Monitoraggio delle risorse

Avremo un monitoraggio semplice, indicando se la home page del servizio sta restituendo uno stato HTTP 200 (OK). Quando si verificano problemi, il nostro playbook indicherà che il log dall'istanza verrà utilizzato per stabilire la causa principale.

Adattarsi alle modifiche nella domanda

Avremo dei manuali per i guasti hardware comuni, gli aggiornamenti software urgenti e altre modifiche che causano interruzioni.

Implementazione della modifica

Utilizzeremo AWS CloudFormation per definire la nostra infrastruttura come codice e in particolare per accelerare la ricostruzione in caso di guasto.

Gli aggiornamenti del software verranno eseguiti manualmente utilizzando un runbook, con tempi di inattività richiesti per l'installazione e il riavvio del servizio. Se si verifica un problema durante la distribuzione, il runbook descrive come ripristinare la versione precedente.

Eventuali correzioni dell'errore vengono effettuate utilizzando l'analisi dei log da parte dei team operativi e di sviluppo e la correzione viene distribuita dopo aver completato e stabilito la priorità della soluzione.

Eseguire il backup dei dati

Utilizzeremo un fornitore o una soluzione di backup appositamente creata per inviare dati di backup crittografati ad Amazon S3 utilizzando un runbook. Verificheremo che i backup funzionino ripristinando i dati e garantendo la possibilità di usarli su base regolare utilizzando un runbook. Configuriamo il controllo delle versioni sui nostri oggetti Amazon S3 e rimuoveremo le autorizzazioni per l'eliminazione dei backup. Usiamo una politica del ciclo di vita del bucket Amazon S3 per archiviare o eliminare definitivamente i dati in base ai nostri requisiti.

Architettura mirata alla resilienza

I carichi di lavoro vengono distribuiti con una regione e una zona di disponibilità. Distribuiamo l'applicazione, incluso il database, in un'unica istanza.

Testare la resilienza

È prevista una pipeline di distribuzione del nuovo software, con alcuni test unitari, ma si tratta principalmente di test white-box/black-box del carico di lavoro assemblato.

Pianificazione per il disaster recovery (DR)

In caso di guasto, attendiamo il completamento dell'errore, con la possibilità di instradare le richieste a un sito Web statico utilizzando la modifica DNS tramite un runbook. Il tempo di ripristino sarà determinato dalla velocità con cui l'infrastruttura può essere distribuita e il database può essere ripristinato al backup più recente. Questa distribuzione può trovarsi nella stessa zona di disponibilità o in una zona di disponibilità diversa in caso di guasto della zona di disponibilità mediante un runbook.

Obiettivo di progettazione della disponibilità

Servono 30 minuti per comprendere e decidere di eseguire il ripristino, 10 minuti per distribuire l'intero stack in AWS CloudFormation, assumendo di distribuire in una nuova zona di disponibilità e che il database possa essere ripristinato in 30 minuti. Questo implica che ci vogliono circa 70 minuti per

ripristinare un guasto. Supponendo un guasto per trimestre, il nostro tempo di impatto stimato per l'anno è di 280 minuti, ovvero quattro ore e 40 minuti.

Ciò significa che il limite massimo di disponibilità è del 99,9%. La disponibilità effettiva dipenderà anche dal tasso reale di guasto, dalla durata dell'errore e dalla rapidità con cui ogni guasto viene di fatto ripristinato. Per questa architettura prevediamo che l'applicazione sia offline durante gli aggiornamenti (stima di 24 ore all'anno: quattro ore per update, sei volte all'anno), oltre agli eventi reali. Quindi, facendo riferimento alla tabella sulla disponibilità delle applicazioni più sopra nel whitepaper, vediamo che il nostro obiettivo di progettazione della disponibilità è del 99%.

Riepilogo

Argomento	Implementazione
Monitoraggio delle risorse	Solo controllo dello stato del sito; nessun avviso.
Adattarsi alle modifiche nella domanda	Ridimensionamento verticale tramite ridistribuzione.
Implementazione della modifica	Runbook per distribuzione e rollback.
Eseguire il backup dei dati	Runbook per backup e ripristino.
Architettura mirata alla resilienza	Ricostruzione completa; ripristino tramite backup.
Testare la resilienza	Ricostruzione completa; ripristino tramite backup.
Pianificazione per il disaster recovery (DR)	Backup crittografati, ripristino in una zona di disponibilità diversa, se necessario.

Scenario a tre 9 (99,9%)

Il prossimo obiettivo di disponibilità è per le applicazioni per le quali è importante essere altamente disponibili, ma che possono tollerare brevi periodi di indisponibilità. Questo tipo di carico di lavoro viene in genere utilizzato per le operazioni interne che hanno un effetto sui dipendenti quando non sono attivi. Questo tipo di carico di lavoro può essere utilizzato anche per sistemi rivolti al cliente ma

non costituisce un'entrata elevata per l'azienda e può tollerare tempi di ripristino o punti di ripristino più lunghi. Tali carichi di lavoro includono applicazioni amministrative per la gestione degli account o delle informazioni.

Possiamo migliorare la disponibilità per i carichi di lavoro utilizzando due zone di disponibilità per la nostra distribuzione e separando le applicazioni in livelli separati.

Monitoraggio delle risorse

Il monitoraggio verrà ampliato per avvisare della disponibilità del sito Web soprattutto controllando lo stato HTTP 200 (OK) nella homepage. Inoltre, ci saranno avvisi su ogni sostituzione di un server Web e quando si verifica un failover del database. Monitoreremo anche la disponibilità dei contenuti statici su Amazon S3 e avviseremo in caso di mancata disponibilità. I log verranno aggregati per facilitare la gestione e per aiutare nell'analisi della causa principale.

Adattarsi alle modifiche nella domanda

La scalabilità automatica è configurata per monitorare l'utilizzo della CPU sulle istanze EC2 e aggiungere o rimuovere istanze per mantenere il target CPU al 70%, ma con non meno di un'istanza EC2 per zona di disponibilità. Se i modelli di caricamento sull'istanza RDS indicano che è necessario aumentare il dimensionamento, il tipo di istanza verrà modificato durante una finestra di manutenzione.

Implementazione della modifica

Le tecnologie di distribuzione dell'infrastruttura rimangono le stesse dello scenario precedente.

La distribuzione di nuovo software ha una scadenza fissa ogni due o quattro settimane. Gli aggiornamenti del software saranno automatizzati, senza utilizzare modelli di distribuzione canary o blue/green, ma piuttosto utilizzando la sostituzione in loco. La decisione di eseguire il rollback verrà presa utilizzando il runbook.

Avremo dei playbook per stabilire la causa principale dei problemi. Dopo che la causa principale è stata identificata, la correzione dell'errore sarà identificata da una combinazione di team operativi e di sviluppo. La correzione verrà distribuita dopo aver sviluppato la correzione.

Eseguire il backup dei dati

Il backup e il ripristino possono essere eseguiti utilizzando Amazon RDS. Verrà eseguito regolarmente utilizzando un runbook per garantire che possiamo soddisfare i requisiti di ripristino.

Architettura mirata alla resilienza

Possiamo migliorare la disponibilità per le applicazioni utilizzando due zone di disponibilità per la nostra distribuzione e separando le applicazioni in livelli separati. Utilizzeremo servizi che funzionano su più zone di disponibilità, ad esempio Elastic Load Balancing, Auto Scaling e Amazon RDS Multi-AZ con storage crittografato tramite AWS Key Management Service. Ciò garantirà la tolleranza ai guasti a livello di risorsa e a livello di zona di disponibilità.

Il sistema di bilanciamento del carico indirizzerà il traffico solo verso istanze di applicazioni integre. Il controllo dello stato deve essere sul piano dati/sul livello dell'applicazione che indica la capacità dell'applicazione sull'istanza. Questa verifica non dovrebbe essere sul piano di controllo. Un URL di controllo dello stato per l'applicazione Web sarà presente e configurato per l'uso da parte del sistema di bilanciamento del carico e di Auto Scaling, in modo che le istanze con esito negativo vengano rimosse e sostituite. Amazon RDS gestirà il motore di database attivo affinché sia disponibile nella seconda zona di disponibilità se l'istanza ha esito negativo nella zona di disponibilità primaria, quindi effettuerà il ripristino con la stessa resilienza.

Dopo avere separato i livelli, è possibile utilizzare modelli di resilienza distribuiti per aumentare l'affidabilità in modo che l'applicazione possa essere ancora disponibile anche quando il database è temporaneamente non disponibile durante un failover della zona di disponibilità.

Testare la resilienza

I test funzionali vengono eseguiti come nello scenario precedente. Non testiamo le funzionalità di correzione automatica di ELB, scalabilità automatica o failover RDS.

Avremo dei playbook per problemi di database comuni, incidenti relativi alla sicurezza e distribuzioni non riuscite.

Pianificazione per il disaster recovery (DR)

Esistono runbook per il ripristino totale del carico di lavoro e report comuni. Il ripristino utilizza i backup archiviati nella stessa regione del carico di lavoro.

Obiettivo di progettazione della disponibilità

Partiamo dal presupposto che almeno alcuni guasti richiederanno una decisione manuale per eseguire il ripristino. Tuttavia, con una maggiore automazione in questo scenario, supponiamo che solo due eventi all'anno richiedano questa decisione. Dedichiamo 30 minuti per decidere di eseguire

il ripristino e supponiamo che il ripristino sia completato entro 30 minuti. Ciò implica 60 minuti per il ripristino del guasto. Supponendo due incidenti all'anno, il nostro tempo di impatto stimato per l'anno è di 120 minuti.

Ciò significa che il limite massimo di disponibilità è del 99,95%. La disponibilità effettiva dipenderà anche dal tasso reale di errore, dalla durata del guasto e dalla rapidità con cui ciascun componente viene ripristinato nella pratica. Per questa architettura è necessario che l'applicazione sia brevemente offline per gli aggiornamenti, ma questi aggiornamenti sono automatizzati. Stimiamo 150 minuti all'anno per questo: 15 minuti per update, 10 volte all'anno. Ciò aggiunge fino a 270 minuti all'anno quando il servizio non è disponibile, quindi il nostro obiettivo di progettazione della disponibilità è pari al 99,9%.

Riepilogo

Argomento	Implementazione
Monitoraggio delle risorse	Solo controllo dello stato del sito; avvisi inviati quando inattivo.
Adattarsi alle modifiche nella domanda	ELB per il livello di applicazione Web e di scalabilità automatica; dimensionamento di RDS con Multi-AZ.
Implementazione della modifica	Distribuzione automatica con sostituzione diretta e runbook per il rollback.
Eseguire il backup dei dati	Backup automatici tramite RDS per soddisfare RPO e runbook per il ripristino.
Architettura mirata alla resilienza	Scalabilità automatica per fornire un livello di applicazione e Web autorigeneranti; RDS con Multi-AZ.
Testare la resilienza	ELB e applicazione autorigeneranti; RDS con Multi-AZ; nessun test esplicito.
Pianificazione per il disaster recovery (DR)	Backup crittografati tramite RDS nella stessa Regione AWS.

Scenario a quattro 9 (99,99%)

Questo obiettivo di disponibilità per le applicazioni richiede che l'applicazione sia altamente disponibile e tollerante ai guasti dei componenti. L'applicazione deve essere in grado di assorbire i guasti senza la necessità di ottenere risorse aggiuntive. Questo obiettivo di disponibilità è per le applicazioni mission critical che sono fonte di entrate principali o significative per un'azienda, come un sito di e-commerce, un servizio web business-to-business o un sito di contenuti/media ad alto traffico.

Possiamo migliorare ulteriormente la disponibilità utilizzando un'architettura che sarà staticamente stabile all'interno della regione. Questo obiettivo di disponibilità non richiede una modifica del piano di controllo nel comportamento del nostro carico di lavoro per tollerare il guasto. Ad esempio, dovrebbe esserci una capacità sufficiente per resistere alla perdita di una zona di disponibilità. Non è necessario richiedere aggiornamenti al Amazon Route 53 DNS. Non dovremmo aver bisogno di creare alcuna nuova infrastruttura, che si tratti di creare o modificare un bucket S3, creare nuove policy IAM (o modificare le policy) o modificare le configurazioni delle attività di Amazon ECS.

Monitoraggio delle risorse

Il monitoraggio includerà parametri di successo e avvisi quando si verificano problemi. Inoltre, ci saranno avvisi su ogni sostituzione di un server Web, quando si verifica un failover del database e quando si verifica un errore di zona di disponibilità.

Adattarsi alle modifiche nella domanda

Utilizzeremo Amazon Aurora come RDS, che consente la scalabilità automatica delle repliche di lettura. Per queste applicazioni, la progettazione per la disponibilità in lettura rispetto alla disponibilità in scrittura dei contenuti principali è anche una decisione essenziale per l'architettura. Aurora può anche aumentare automaticamente lo storage in base alle esigenze con incrementi di 10 GB, fino a 64 TB.

Implementazione della modifica

Distribuiremo gli aggiornamenti utilizzando le distribuzioni Canary o blue/green in ciascuna zona di isolamento singolarmente. Le distribuzioni sono completamente automatizzate, incluso un rollback se gli indicatori KPI indicano un problema.

Esistono runbook per requisiti di reporting rigorosi e monitoraggio delle prestazioni. Se le operazioni riuscite tendono a non raggiungere gli obiettivi di prestazione o disponibilità, verrà utilizzato un

manuale per stabilire cosa sta causando la tendenza. Esistono dei manuali per le modalità di errore non scoperte e gli incidenti di sicurezza. Esistono anche playbook per stabilire la causa principale dei guasti. Ci impegneremo anche con AWS Support per l'offerta di Infrastructure Event Management.

Il team che crea e fa funzionare il sito Web identificherà la correzione per qualsiasi guasto imprevisto e darà la priorità alla correzione da distribuire dopo che è stata implementata.

Eseguire il backup dei dati

Il backup e il ripristino possono essere eseguiti utilizzando Amazon RDS. Verrà eseguito regolarmente utilizzando un runbook per garantire che possiamo soddisfare i requisiti di ripristino.

Architettura mirata alla resilienza

Consigliamo tre zone di disponibilità per questo approccio. Utilizzando una distribuzione con tre zone di disponibilità, ciascuna zona di disponibilità ha una capacità statica del 50% di picco. Potrebbero essere utilizzate due zone di disponibilità, ma il costo della capacità staticamente stabile sarebbe maggiore perché entrambe le zone di disponibilità dovrebbero avere il 100% della capacità di picco. Aggiungeremo Amazon CloudFront per fornire la memorizzazione geografica nella cache, oltre a richiedere una riduzione sul piano dati della nostra applicazione.

Utilizzeremo Amazon Aurora come RDS e distribuiremo repliche di lettura in tutte e tre le zone.

L'applicazione verrà creata utilizzando i modelli di resilienza software/applicazione in tutti i livelli.

Testare la resilienza

La pipeline di distribuzione avrà una suite di test completa, inclusi test di prestazioni, carico e inserimento degli errori.

Eserciteremo costantemente le nostre procedure di recupero degli errori durante i giorni di attività, utilizzando i runbook per assicurarci di poter eseguire le attività e non deviare dalle procedure. Il team che costruisce il sito Web gestisce anche il sito Web.

Pianificazione per il disaster recovery (DR)

Esistono runbook per il ripristino totale del carico di lavoro e report comuni. Il ripristino utilizza i backup archiviati nella stessa regione del carico di lavoro. Le procedure di ripristino vengono regolarmente esercitate come parte delle giornate di gioco.

Obiettivo di progettazione della disponibilità

Partiamo dal presupposto che almeno alcuni guasti richiederanno una decisione manuale per eseguire il ripristino, tuttavia con una maggiore automazione in questo scenario ipotizziamo che solo due eventi l'anno richiedano questa decisione e le azioni di ripristino saranno rapide. Dedichiamo 10 minuti per decidere di eseguire il ripristino e supponiamo che il ripristino sia completato entro cinque minuti. Ciò implica 15 minuti per il ripristino del guasto. Supponendo due guasti all'anno, il nostro tempo di impatto stimato per l'anno è di 30 minuti.

Ciò significa che il limite massimo di disponibilità è del 99,99%. La disponibilità effettiva dipenderà anche dal tasso reale di errore, dalla durata del guasto e dalla rapidità con cui ciascun componente viene ripristinato nella pratica. Per questa architettura ipotizziamo che l'applicazione sia costantemente online durante gli aggiornamenti. Sulla base di questo, il nostro obiettivo di progettazione della disponibilità è del 99,99%.

Riepilogo

Argomento	Implementazione
Monitoraggio delle risorse	Controlli di integrità a tutti i livelli e sugli indicatori KPI; avvisi inviati quando vengono attivati gli allarmi configurati; avvisi su tutti i guasti. Le riunioni operative sono rigorose per rilevare le tendenze e raggiungere gli obiettivi di progettazione.
Adattarsi alle modifiche nella domanda	ELB per il livello applicativo di scalabilità automatica e Web; storage di scalabilità automatica e repliche di lettura in più zone per Aurora RDS.
Implementazione della modifica	Distribuzione automatizzata tramite canary o blue/green e rollback automatico quando KPI o avvisi indicano problemi non rilevati nell'applicazione. Le distribuzioni vengono effettuate per zona di isolamento.
Eseguire il backup dei dati	Backup automatizzati tramite RDS per soddisfare RPO e ripristino automatico che

Argomento	Implementazione
	viene praticato regolarmente in una giornata di gioco.
Architettura mirata alla resilienza	Implementate zone di isolamento degli errori per l'applicazione; ridimensionamento automatico per fornire livelli di applicazione e web autorigeneranti; RDS con Multi-AZ.
Testare la resilienza	La verifica dei guasti delle componenti e della zona di isolamento è nella pipeline ed effettuata regolarmente con il personale operativo in una giornata di gioco; esistono dei manuali per diagnosticare problemi sconosciuti; ed esiste un processo di analisi della causa principale.
Pianificazione per il disaster recovery (DR)	Backup crittografati tramite RDS nella stessa Regione AWS effettuati in una giornata di gioco.

Scenari multi-regione

L'implementazione della nostra applicazione in più regioni AWS aumenterà i costi operativi, in parte perché isoliamo le regioni per mantenere la loro autonomia. Seguire questo approccio dovrebbe essere una decisione molto ponderata. Detto questo, le regioni offrono un confine di isolamento forte e facciamo del nostro meglio per evitare guasti correlati tra le regioni. L'uso di più aree ti consentirà di avere un maggiore controllo sui tempi di ripristino in caso di un grave errore di dipendenza in un servizio AWS regionale. In questa sezione, discuteremo di vari modelli di implementazione e della loro tipica disponibilità.

Argomenti

- [Scenario a tre 9 e ½ \(99,95%\) con un tempo di ripristino compreso tra 5 e 30 minuti](#)
- [Scenario a cinque 9 \(99,999%\) o superiore con un tempo di ripristino inferiore a 1 minuto](#)

Scenario a tre 9 e ½ (99,95%) con un tempo di ripristino compreso tra 5 e 30 minuti

Questo obiettivo di disponibilità per le applicazioni richiede tempi di inattività estremamente brevi e una perdita di dati molto ridotta per periodi specifici. Le applicazioni con questo obiettivo di disponibilità includono applicazioni nei settori: bancario, investimenti, servizi di emergenza e acquisizione dei dati. Queste applicazioni hanno tempi di ripristino e punti di ripristino molto brevi.

È possibile migliorare ulteriormente i tempi di ripristino utilizzando un approccio Approccio warm standby tra due regioni AWS. Distribuiremo l'intero carico di lavoro in entrambe le regioni, con il nostro sito passivo ridimensionato e mantenendo la coerenza finale di tutti i dati. Entrambe le distribuzioni saranno staticamente stabili all'interno delle rispettive regioni. Le applicazioni devono essere create utilizzando i modelli di resilienza del sistema distribuito. Dovremo creare un componente di instradamento leggero che monitori lo stato del carico di lavoro e, se necessario, possa essere configurato per instradare il traffico verso la regione passiva.

Monitoraggio delle risorse

Inoltre, ci saranno avvisi per ogni sostituzione di un server Web, quando si verifica un failover del database e quando si verifica un errore di regione. Monitoreremo anche la disponibilità dei contenuti statici su Amazon S3 e avviseremo in caso di mancata disponibilità. I log verranno aggregati per facilitare la gestione e per aiutare nell'analisi della causa principale in ciascuna regione.

Il componente di instradamento monitora sia lo stato della nostra applicazione sia le dipendenze hard regionali che abbiamo.

Adattarsi alle modifiche nella domanda

Uguale allo scenario a quattro 9.

Implementazione della modifica

La distribuzione di nuovo software ha una scadenza fissa ogni due o quattro settimane. Gli aggiornamenti del software saranno automatizzati, utilizzando modelli di distribuzione canary o blue/green.

Esistono runbook per quando si verifica il failover della Regione, per problemi comuni dei clienti che si verificano durante tali eventi e per report comuni.

Avremo manuali per problemi di database comuni, incidenti relativi alla sicurezza, distribuzioni non riuscite, problemi impreveduti dei clienti nel failover della Regione e per stabilire la causa principale dei problemi. Dopo che la causa principale è stata identificata, la correzione dell'errore sarà identificata da una combinazione di team operativi e di sviluppo e distribuita quando la correzione è sviluppata.

Ci impegneremo anche con AWS Support per l'offerta di Infrastructure Event Management.

Eseguire il backup dei dati

Come per lo scenario a quattro 9, usiamo backup RDS automatici e la funzione di controllo delle versioni di S3. I dati vengono replicati automaticamente e in modo asincrono dal cluster Aurora RDS nella regione attiva alle repliche di lettura tra regioni nella regione passiva. La replica tra regioni S3 viene utilizzata per spostare automaticamente e in modo asincrono i dati dalla regione attiva a quella passiva.

Architettura mirata alla resilienza

Come nello scenario a quattro 9, inoltre è possibile il failover regionale. Questa operazione viene gestita manualmente. Durante il failover, indirizzeremo le richieste a un sito Web statico utilizzando il failover DNS fino al ripristino nella seconda regione.

Testare la resilienza

Come per lo scenario a quattro 9 più una convalida dell'architettura attraverso i game day utilizzando i runbook. Inoltre, la correzione RCA ha la priorità rispetto al rilascio delle funzionalità per l'implementazione e la distribuzione immediate

Pianificazione per il disaster recovery (DR)

Il failover regionale viene gestito manualmente. Tutti i dati vengono replicati in modo asincrono. L'infrastruttura in warm standby viene dimensionata. Questa può essere automatizzata utilizzando un flusso di lavoro eseguito su AWS Step Functions. Anche AWS Systems Manager (SSM) può essere d'aiuto con questa automazione, poiché è possibile creare documenti SSM che aggiornano i gruppi Auto Scaling e dimensionano le istanze.

Obiettivo di progettazione della disponibilità

Partiamo dal presupposto che almeno alcuni guasti richiederanno una decisione manuale per eseguire il ripristino, tuttavia, con una maggiore automazione, in questo scenario ipotizziamo che solo due eventi l'anno richiedano questa decisione. Impieghiamo 20 minuti per decidere di eseguire

il ripristino e supponiamo che questo sia completato entro 10 minuti. Questo implica che ci vogliono circa 30 minuti per eseguire il ripristino da un guasto. Supponendo due incidenti all'anno, il nostro tempo di impatto stimato per l'anno è di 60 minuti.

Ciò significa che il limite massimo di disponibilità è del 99,95%. La disponibilità effettiva dipenderà anche dal tasso reale di errore, dalla durata del guasto e dalla rapidità con cui ciascun componente viene ripristinato nella pratica. Per questa architettura ipotizziamo che l'applicazione sia costantemente online durante gli aggiornamenti. Sulla base di questo, il nostro obiettivo di progettazione della disponibilità è pari al 99,95%.

Riepilogo

Argomento	Implementazione
Monitoraggio delle risorse	Controlli di integrità a tutti i livelli, compresi i DNS a livello di regione AWS, e sugli indicatori KPI; avvisi inviati quando vengono attivati gli allarmi configurati; allerta su tutti i guasti. Le riunioni operative sono rigorose per rilevare le tendenze e raggiungere gli obiettivi di progettazione.
Adattarsi alle modifiche nella domanda	ELB per il livello applicativo di scalabilità automatica e Web; storage di scalabilità automatica e repliche di lettura in più zone nelle regioni attive e passive per Aurora RDS. Dati e infrastruttura sincronizzati tra regioni AWS per una stabilità statica.
Implementazione della modifica	Distribuzione automatizzata tramite canary o blue/green e rollback automatico quando i KPI o gli avvisi indicano problemi non rilevati nell'applicazione, le distribuzioni vengono effettuate in una zona di isolamento in una Regione AWS alla volta.
Eseguire il backup dei dati	Backup automatizzati in ciascuna Regione AWS tramite RDS per soddisfare RPO e

Argomento	Implementazione
	<p>ripristino automatico effettuato regolarmente in una giornata di gioco. I dati di Aurora RDS e S3 vengono replicati automaticamente e in modo asincrono da una regione attiva a una regione passiva.</p>
<p>Architettura mirata alla resilienza</p>	<p>Scalabilità automatica per fornire un livello di applicazione e Web autorigeneranti; RDS con Multi-AZ; il failover regionale viene gestito manualmente con il sito statico mostrato durante il failover.</p>
<p>Testare la resilienza</p>	<p>La verifica dei guasti delle componenti e della zona di isolamento è nella pipeline ed effettuata regolarmente con il personale operativo in una giornata di gioco; esistono dei manuali per diagnosticare problemi sconosciuti; ed esiste un processo di analisi della causa principale, con percorsi di comunicazione per individuare il problema e come questo è stato corretto o prevenuto. La correzione RCA ha la priorità rispetto al rilascio delle funzionalità per l'implementazione e la distribuzione immediate.</p>
<p>Pianificazione per il disaster recovery (DR)</p>	<p>Warm Standby distribuito in un'altra regione. L'infrastruttura viene dimensionata utilizzando flussi di lavoro eseguiti utilizzando AWS Step Functions o documenti AWS Systems Manager. Backup crittografati tramite RDS. Repliche di lettura tra regioni tra due regioni AWS. Replica tra regioni di asset statici in Amazon S3. Il ripristino è nella Regione AWS attiva corrente, viene effettuato in una giornata di gioco ed è coordinato con AWS.</p>

Scenario a cinque 9 (99,999%) o superiore con un tempo di ripristino inferiore a 1 minuto

Questo obiettivo di disponibilità per le applicazioni richiede tempi di inattività estremamente brevi e una perdita di dati molto ridotta per periodi specifici. Le applicazioni che potrebbero avere questo obiettivo di disponibilità includono, ad esempio, alcune applicazioni bancarie, di investimento, finanziarie, governative e di business cruciali che costituiscono le attività principali di un'azienda che genera entrate estremamente elevate. L'obiettivo è di avere archivi di dati fortemente coerenti e completa ridondanza a tutti i livelli. Abbiamo selezionato un archivio dati basato su SQL. Tuttavia, in alcuni scenari, troveremo difficile ottenere un RPO molto basso. Se riesci a partizionare i tuoi dati è possibile che non ci siano perdite di dati. Ciò potrebbe richiedere l'aggiunta di logica e di latenza nell'applicazione per garantire la coerenza dei dati tra le posizioni geografiche, nonché la capacità di spostare o copiare i dati tra le partizioni. L'esecuzione di questo partizionamento potrebbe essere più semplice se si utilizza un database NoSQL.

Possiamo migliorare ulteriormente la disponibilità utilizzando un approccio Attivo-attivo approccio su più regioni AWS. Il carico di lavoro verrà distribuito in tutte le regioni desiderate che sono staticamente stabili tra regioni (in modo che le regioni rimanenti possano gestire il carico con la perdita di una regione). Un instradamento indirizza il traffico verso le posizioni geografiche integre e modifica automaticamente la destinazione quando una posizione non è integra, interrompendo anche temporaneamente i livelli di replica dei dati. Amazon Route 53 offre controlli di integrità a intervalli di 10 secondi e offre anche TTL sui set di record fino a un secondo.

Monitoraggio delle risorse

Come per lo scenario a 3½ 9, e inoltre vengono emessi avvisi quando una regione viene rilevata come non integra e il traffico viene allontanato da essa.

Adattarsi alle modifiche nella domanda

Uguale allo scenario a 3 9 e ½.

Implementazione della modifica

La pipeline di distribuzione avrà una suite di test completa, inclusi test di prestazioni, carico e inserimento degli errori. Distribuiremo gli aggiornamenti utilizzando le distribuzioni canary o blue/green in una zona di isolamento alla volta, in una singola regione, prima di passare a un'altra. Durante la distribuzione, le versioni precedenti continueranno a essere in esecuzione su altre istanze per facilitare un rollback più veloce. Queste sono completamente automatizzate, incluso un rollback

se gli indicatori KPI indicano un problema. Il monitoraggio includerà parametri di successo e avvisi quando si verificano problemi.

Esistono runbook per requisiti di reporting rigorosi e monitoraggio delle prestazioni. Se le operazioni riuscite tendono a non raggiungere gli obiettivi di prestazione o disponibilità, verrà utilizzato un manuale per stabilire cosa sta causando la tendenza. Esistono dei manuali per le modalità di errore non scoperte e gli incidenti di sicurezza. Esistono anche playbook per stabilire la causa principale dei guasti.

Il team che costruisce il sito Web gestisce anche il sito Web. Quel team identificherà la correzione per qualsiasi guasto imprevisto e darà la priorità alla correzione da distribuire dopo che è stata implementata. Ci impegneremo anche con AWS Support per l'offerta di Infrastructure Event Management.

Eseguire il backup dei dati

Uguale allo scenario a 3 9 e ½.

Architettura mirata alla resilienza

L'applicazione verrà creata utilizzando i modelli di resilienza software/applicazione. È possibile che siano necessari molti altri livelli di routing per implementare la disponibilità necessaria. La complessità di questa implementazione aggiuntiva non deve essere sottovalutata. L'applicazione verrà implementata nelle zone di isolamento degli errori di distribuzione e partizionata e distribuita in modo tale che anche un evento su tutta la regione non influirà su tutti i clienti.

Testare la resilienza

Eserciteremo costantemente le nostre procedure di recupero degli errori durante i giorni di attività, utilizzando i runbook per assicurarci di poter eseguire le attività e non deviare dalle procedure.

Pianificazione per il disaster recovery (DR)

Attiva-attiva e distribuzione multi-regione con infrastruttura completa del carico di lavoro e dati in più regioni. Utilizzando una strategia globale di lettura locale e di scrittura, una regione è il database master per tutte le scritture e i dati vengono replicati per le letture in altre regioni. Se la regione database master ha esito negativo, sarà necessario promuovere un nuovo database. La strategia di lettura locale e di scrittura prevede l'assegnazione di utenti a una regione di origine in cui vengono gestite le scritture DB. Ciò consente agli utenti di leggere o scrivere da qualsiasi regione, ma richiede una logica complessa per gestire potenziali conflitti di dati tra scritture in regioni diverse.

Quando una regione viene rilevata come non integra, il livello di instradamento instrada automaticamente il traffico verso le regioni integre rimanenti. Non è richiesto alcun intervento manuale.

Gli archivi di dati devono essere replicati tra le regioni in modo da poter risolvere potenziali conflitti. Sarà necessario creare strumenti e processi automatizzati per copiare o spostare i dati tra le partizioni per motivi di latenza e per bilanciare richieste o quantità di dati in ciascuna partizione. Anche la risoluzione dei conflitti di dati richiederà runbook operativi aggiuntivi.

Obiettivo di progettazione della disponibilità

Partiamo dal presupposto che vengono effettuati ingenti investimenti per automatizzare tutto il ripristino e che il ripristino può essere completato in un minuto. Non prevediamo ripristini attivati manualmente, ma fino a un'azione di ripristino automatizzata al trimestre. Ciò implica quattro minuti all'anno per il ripristino. Ipotizziamo che l'applicazione sia costantemente online durante gli aggiornamenti. Sulla base di questo, il nostro obiettivo di progettazione della disponibilità è pari al 99,999%.

Riepilogo

Argomento	Implementazione
Monitoraggio delle risorse	Controlli di integrità a tutti i livelli, compresi i DNS a livello di regione AWS, e sugli indicatori KPI; avvisi inviati quando vengono attivati gli allarmi configurati; allerta su tutti i guasti. Le riunioni operative sono rigorose per rilevare le tendenze e raggiungere gli obiettivi di progettazione.
Adattarsi alle modifiche nella domanda	ELB per il livello applicativo di scalabilità automatica e Web; storage di scalabilità automatica e repliche di lettura in più zone nelle regioni attive e passive per Aurora RDS. Dati e infrastruttura sincronizzati tra regioni AWS per una stabilità statica.
Implementazione della modifica	Distribuzione automatizzata tramite canary o blue/green e rollback automatico quando i

Argomento	Implementazione
	KPI o gli avvisi indicano problemi non rilevati nell'applicazione, le distribuzioni vengono effettuate in una zona di isolamento in una Regione AWS alla volta.
Eeguire il backup dei dati	Backup automatizzati in ciascuna Regione AWS tramite RDS per soddisfare RPO e ripristino automatico effettuato regolarmente in una giornata di gioco. I dati di Aurora RDS e S3 vengono replicati automaticamente e in modo asincrono da una regione attiva a una regione passiva.
Architettura mirata alla resilienza	Implementate zone di isolamento degli errori per l'applicazione; auto scaling per fornire livelli di applicazione e web autorigeneranti; RDS con Multi-AZ, failover regionale automatizzato.
Testare la resilienza	La verifica dei guasti delle componenti e della zona di isolamento è nella pipeline ed effettuata regolarmente con il personale operativo in una giornata di gioco; esistono dei manuali per diagnosticare problemi sconosciuti; ed esiste un processo di analisi della causa principale, con percorsi di comunicazione per individuare il problema e come questo è stato corretto o prevenuto. La correzione RCA ha la priorità rispetto al rilascio delle funzionalità per l'implementazione e la distribuzione immediate.

Argomento	Implementazione
Pianificazione per il disaster recovery (DR)	Configurazione attiva-attiva distribuita in almeno due regioni. L'infrastruttura è completamente ridimensionata e staticamente stabile tra le regioni. I dati vengono partizionati e sincronizzati tra regioni. Backup crittografati tramite RDS. Il guasto di una regione viene praticato in una giornata di gioco ed è coordinato con AWS. Durante il ripristino, potrebbe essere necessario promuovere un nuovo master del database.

Risorse

Documentazione

- [Amazon Builders' Library](#) - Come Amazon crea e gestisce software
- [Centro di progettazione AWS](#)

I corsi

- [I corsi sull'affidabilità di AWS Well-Architected](#)

Collegamenti esterni

- Modello di accodamento adattivo: [Fail at Scale \(errori su vasta scala\)](#)
- [Oltre la disponibilità: comprendere e migliorare la resilienza dei sistemi distribuiti su AWS](#)

Libri

- Robert S. Hammer "[Modelli per software tollerante ai guasti](#)"
- Andrew Tanenbaum e Marten van Steen "[Sistemi distribuiti: principi e paradigmi](#)"

Conclusione

Che tu non abbia alcuna esperienza di argomenti quali disponibilità e affidabilità o che tu sia un esperto in cerca di approfondimenti per massimizzare la disponibilità del tuo carico di lavoro mission critical, speriamo che questo whitepaper abbia stimolato il tuo modo di pensare, ti abbia offerto nuove idee o suggerito nuove possibili domande. Ci auguriamo che ciò ti porti a comprendere in modo più approfondito il giusto livello di disponibilità in base alle esigenze della tua azienda e come progettare l'affidabilità necessaria a raggiungerla. Ti invitiamo a trarre vantaggio dai consigli di progettazione, operativi e orientati al ripristino offerti qui, nonché dalla conoscenza e dall'esperienza degli AWS Solution Architects. Ci piacerebbe avere tue notizie, in particolare sulle tue storie di successo che hanno raggiunto livelli elevati di disponibilità su AWS. Contatta il team responsabile del tuo account o seleziona [Contattaci sul nostro sito Web](#).

Collaboratori

I collaboratori di questo documento includono:

- Seth Eliot, Principal Developer Advocate, Amazon Web Services
- Mahanth Jayadeva, Solutions Architect – Well-Architected, Amazon Web Services
- Amulya Sharma, Principal Solutions Architect, Amazon Web Services
- Jason DiDomenico, Senior Solutions Architect – Cloud Foundations, Amazon Web Services
- Marcin Bednarz, Principal Solutions Architect, Amazon Web Services
- Tyler Applebaum, Senior Solutions Architect, Amazon Web Services
- Rodney Lester, Principal Solutions Architect – App Modernization, Amazon Web Services
- Joe Chapman, Senior Solutions Architect, Amazon Web Services
- Adrian Hornsby, Principal System Development Engineer, Amazon Web Services
- Kevin Miller, Vice President – S3, Amazon Web Services
- Shannon Richards, Principal Technical Program Manager, Amazon Web Services
- Laurent Domb, Chief Technologist - Fed Fin, Amazon Web Services
- Kevin Schwarz, Sr. Solutions Architect, Amazon Web Services
- Rob Martell, Principal Cloud Resilience Architect, Amazon Web Services
- Priyam Reddy, Senior Solutions Architect Manager DR, Amazon Web Services
- Jeff Ferris, Principal Technologist, Amazon Web Services
- Matias Battaglia, Senior Solutions Architect, Amazon Web Services

Approfondimenti

Per ulteriori informazioni, consulta:

- [Framework AWS Well-Architected](#)
- [Centro di progettazione AWS](#)

Revisioni del documento

Per ricevere una notifica sugli aggiornamenti di questo whitepaper, iscriviti al feed RSS.

Modifica	Descrizione	Data
Whitepaper aggiornato	Best practice aggiornate con nuova guida all'implementazione.	June 27, 2024
Linee guida sulle best practice aggiornate	Le best practice sono state aggiornate con nuove linee guida nelle seguenti aree: Progettazione di interazioni in un sistema distribuito per prevenire guasti , Progettazione di interazioni in un sistema distribuito per mitigare o affrontare gli errori , Monitoraggio delle risorse del carico di lavoro , Progettazione di un carico di lavoro in grado di adattarsi ai cambiamenti della domanda , Implementazione della modifica e Test dell'affidabilità .	December 6, 2023
Linee guida sulle best practice aggiornate	Le best practice sono state aggiornate con nuove linee guida nelle seguenti aree: Monitoraggio delle risorse del carico di lavoro e Progettazione di un carico di lavoro resistente agli errori dei componenti .	October 3, 2023

Linee guida sulle best practice aggiornate	Le best practice sono state aggiornate con nuove linee guida nelle seguenti aree: Progettazione dell'architettura del servizio di carico di lavoro , Progettazione di interazioni in un sistema distribuito per mitigare o affrontare gli errori e Monitoraggio delle risorse del carico di lavoro .	July 13, 2023
Aggiornamento di minore entità	Rimuovere linguaggio non inclusivo.	April 13, 2023
Aggiornamenti per il nuovo canone	Best practice aggiornate con prontuario e nuove best practice aggiunte.	April 10, 2023
Whitepaper aggiornato	Best practice aggiornate con nuova guida all'implementazione.	December 15, 2022
Aggiornamenti di minore entità	Numeri delle figure aggiornati e piccole modifiche in tutto il documento.	November 17, 2022
Whitepaper aggiornato	Ampliamento delle best practice e aggiunta dei piani di miglioramento.	October 20, 2022
Whitepaper aggiornato	Aggiunte due nuove best practice al Principio di affidabilità nelle sezioni Utilizzo dell'isolamento dei guasti per proteggere il carico di lavoro e Progettazione di un carico di lavoro resistente agli errori dei componenti.	May 5, 2022

<u>Aggiornamento di minore entità</u>	Aggiunta del pilastro della sostenibilità all'introduzione.	December 2, 2021
<u>Whitepaper aggiornato</u>	Aggiornamento delle linee guida del ripristino di emergenza per includere Route 53 Application Recovery Controller. Aggiunta di riferimenti a DevOps Guru. Aggiornamento di diversi collegamenti alle Risorse e altre modifiche editoriali minori.	October 26, 2021
<u>Aggiornamento di minore entità</u>	Aggiunta di informazioni su AWS Fault Injection Service (AWS FIS).	March 15, 2021
<u>Aggiornamento di minore entità</u>	Aggiornamento minore del testo.	January 4, 2021

[Whitepaper aggiornato](#)

Appendice A aggiornata per modificare l'obiettivo di progettazione della disponibilità per Amazon SQS, Amazon SNS e Amazon MQ; riordinare le righe della tabella per migliorare la ricerca; migliorare e la spiegazione delle differenze tra disponibilità e ripristino di emergenza e come entrambi contribuiscono alla resilienza; espandere la copertura di architetture su più Regioni (per la disponibilità) e di strategie su più Regioni (per il ripristino di emergenza); adeguare il documento di riferimento all'ultima versione; ampliare i calcoli di disponibilità per includere quelli basati sulle richieste e delle scelte rapide; migliorare la descrizione per le Giornate di gioco

December 7, 2020

[Aggiornamento di minore entità](#)

Appendice A aggiornata per aggiungere l'obiettivo di progettazione della disponibilità per AWS Lambda

October 27, 2020

[Aggiornamento di minore entità](#)

Appendice A aggiornata per aggiungere l'obiettivo di progettazione della disponibilità per AWS Global Accelerator

July 24, 2020

[Aggiornamenti per il nuovo canone](#)

Aggiornamenti sostanziali e contenuti nuovi/revisionati, tra cui: aggiunta della sezione di best practice "Architettura del carico di lavoro", riorganizzazione delle best practice nelle sezioni Gestione delle modifiche e Gestione dei guasti, risorse aggiornate per includere le risorse e i servizi AWS più recenti come AWS Global Accelerator, AWS Service Quotas e AWS Transit Gateway, aggiunta/aggiornamento delle definizioni di affidabilità, disponibilità e resilienza, whitepaper più allineato a AWS Well-Architected Tool domande e best practice utilizzato per le revisioni di Well-Architect, riordinamento dei principi di progettazione, spostando Ripristino automatico dagli errori prima di Collaudo delle procedure di ripristino, diagrammi e formati aggiornati per equazioni, rimosse le sezioni Servizi chiave e integrati invece riferimenti a servizi AWS chiave nelle best practice.

July 8, 2020

[Aggiornamento di minore entità](#)

Correzione di un link danneggiato

October 1, 2019

Whitepaper aggiornato	Appendice A aggiornata	April 1, 2019
Whitepaper aggiornato	Aggiunta di consigli specifici sulla rete AWS Direct Connect e di obiettivi di progettazione del servizio aggiuntivi	September 1, 2018
Whitepaper aggiornato	Aggiunte le sezioni Principi di progettazione e Gestione dei limiti. Collegamenti aggiornati, rimossa ambiguità della terminologia upstream/downstream e aggiunti riferimenti espliciti ai restanti argomenti del principio di base di affidabilità negli scenari di disponibilità.	June 1, 2018
Whitepaper aggiornato	È stata modificata la soluzione DynamoDB tra regioni in tabelle globali DynamoDB. Aggiunta degli obiettivi di progettazione del servizio	March 1, 2018
Aggiornamenti di minore entità	Correzione minore al calcolo della disponibilità per includere la disponibilità dell'applicazione	December 1, 2017
Whitepaper aggiornato	Aggiornamento per fornire indicazioni su progetti ad alta disponibilità, inclusi concetti, best practice ed esempi di implementazione.	November 1, 2017
Pubblicazione originale	Pubblicazione del Principio dell'affidabilità – Framework AWS Well-Architected.	November 1, 2016