Framework AWS Well-Architected

Pilastro dell'affidabilità

Pilastro dell'affidabilità: Framework AWS Well-Architected

Table of Contents

Pilastro dell'affidabilità: Framework AWS Well-Architected

Data di pubblicazione: 6 novembre 2024 (Revisioni del documento)

Questo whitepaper tratta del pilastro 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

Il 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. Il Framework fornisce un modo per misurare in modo coerente le architetture rispetto alle best practice e identificare le aree da migliorare. Riteniamo che avere un carico di lavoro ben progettato aumenti notevolmente la probabilità di successo aziendale.

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

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

Il presente 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 on-premises tradizionali a causa di singoli punti di errore, mancanza di automazione e di elasticità. Adottando le prassi del presente documento, creerai architetture con solide basi, un'architettura resiliente, una gestione coerente delle modifiche e processi di ripristino dei guasti comprovati.

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

di architetture caratterizzate dall'affidabilità in un ambiente cloud. Il documento include dettagli di implementazione di alto livello e modelli architetturali, nonché riferimenti a risorse aggiuntive.

Affidabilità

Il pilastro dell'affidabilità comprende la capacità di un carico di lavoro di eseguire la funzione attesa in modo corretto e coerente quando previsto. Ciò comprende la possibilità di utilizzare e testare il carico di lavoro per tutto il ciclo di vita. Il presente 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

La resilienza è una responsabilità condivisa tra te e AWS. È importante comprendere, nell'ambito della resilienza, il funzionamento del disaster recovery (DR) e della disponibilità in questo modello condiviso.

Responsabilità AWS: resilienza del cloud

AWS è responsabile della protezione dell'infrastruttura di esecuzione di tutti i servizi offerti nel Cloud AWS. Questa infrastruttura comprende l'hardware, il software, la rete e le strutture che gestiscono i servizi Cloud AWS. AWS compie sforzi commercialmente ragionevoli per rendere disponibili tali servizi Cloud AWS, garantendo che la loro disponibilità soddisfi o superi gli accordi sul livello di servizio (SLA) di AWS.

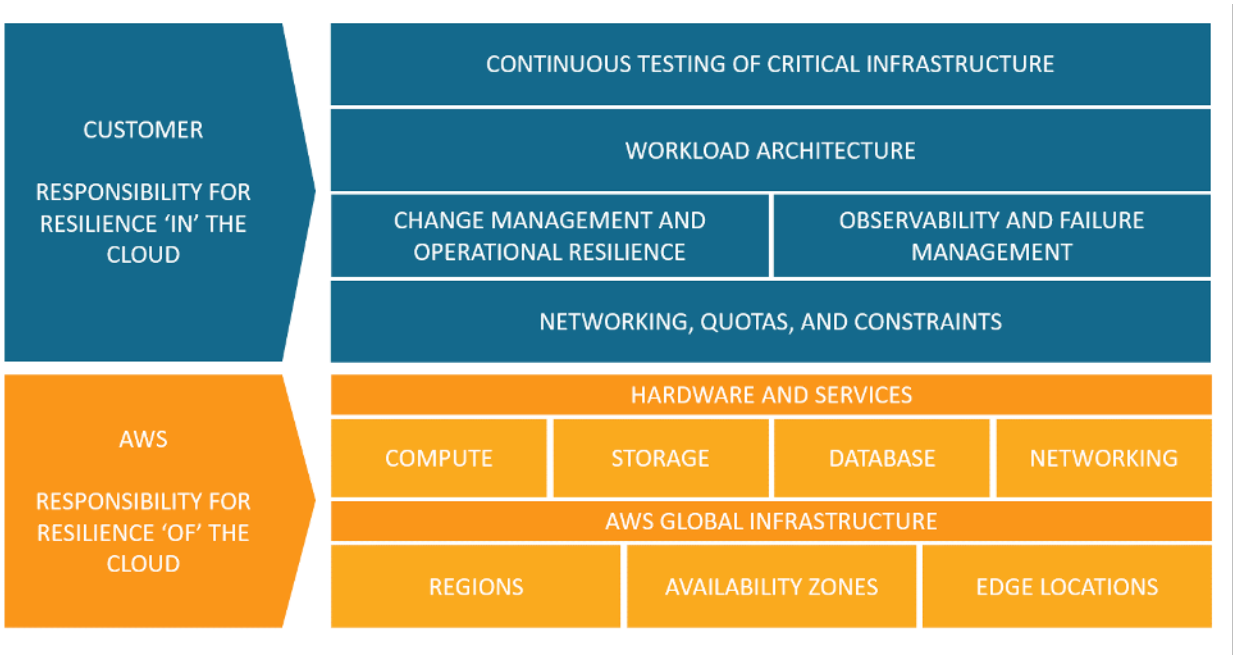
L'infrastruttura cloud globale di AWS è progettata per consentire ai clienti di creare architetture di carichi di lavoro altamente resilienti. Ogni Regione AWS è completamente isolata e comprende diverse zone di disponibilità, ossia partizioni completamente isolate della nostra infrastruttura. Le zone 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 una rete ad alto throughput e bassa latenza tra le zone. Tutto il traffico tra zone è crittografato. Le prestazioni di rete sono adeguate per l'esecuzione della replica sincrona tra zone. In caso di partizione di

un'applicazione 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 del cloud

La tua responsabilità è determinata dai servizi Cloud AWS che scegli. La scelta definisce l'entità delle attività di configurazione che devi eseguire nell'ambito delle tue responsabilità nell'ambito della 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 delle istanze Amazon EC2 in più sedi (come le zone di disponibilità AWS), dell'implementazione della riparazione automatica tramite servizi come Auto Scaling e dell'utilizzo delle best practice per un'architettura resiliente per carichi di lavoro per le applicazioni installate sulle 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 disaster recovery 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 disaster recovery 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 disaster recovery se la regione attiva non può più gestire le richieste.



Responsabilità per la resilienza all'interno 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 maggiori dettagli su ciascun servizio in particolare, consulta la documentazione AWS.

Reti, quote e vincoli

- La sezione Fondamenti illustra le best practice per quest'area del modello di responsabilità condivisa.
- Pianifica la tua architettura con un margine di scalabilità adeguato e analizza le quote di servizio, nonché i vincoli dei servizi inclusi, in base agli aumenti previsti delle richieste di carico, 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 comprende le modalità di introduzione e gestione delle modifiche nell'ambiente. L'implementazione delle 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, ivi comprese metriche tecniche e aziendali, notifiche, automazione e analisi.

- I carichi di lavoro nel cloud devono adattarsi ai cambiamenti nella domanda in senso di riduzione orizzontale in risposta a riduzioni o fluttuazioni nell'utilizzo.

Osservabilità e gestione dei guasti

- L'osservazione dei guasti attraverso il monitoraggio è necessaria per automatizzare la correzione in modo che i carichi di lavoro possano resistere ai guasti dei componenti.
- La gestione dei guasti richiede il backup dei dati, l'applicazione delle best practice per consentire al carico di lavoro di resistere ai guasti dei componenti e la pianificazione del disaster recovery.

Architettura del carico di lavoro

- L'architettura del carico di lavoro include la progettazione di servizi in base ai domini aziendali, l'applicazione della SOA e la progettazione di sistemi distribuiti per prevenire i guasti e l'integrazione di funzionalità come limitazione (della larghezza di banda della rete), nuovi tentativi, gestione delle code, timeout e leve di emergenza.
- Affidati a soluzioni AWS comprovate, ad Amazon Builders' Library e a modelli serverless per allinearti alle best practice e avviare subito le implementazioni.
- Apporta miglioramenti continui per scomporre il sistema in servizi distribuiti per una scalabilità e un'innovazione più rapide. Utilizza le indicazioni relative ai microservizi AWS e le opzioni di servizio gestito per semplificare e accelerare la tua capacità di introdurre modifiche e innovazioni.

Esecuzione continua di test dell'infrastruttura critica

- Testare l'affidabilità significa eseguire test a livello funzionale, prestazionale e di caos, nonché adottare l'analisi degli incidenti e le pratiche delle giornate di gioco per acquisire competenze nella risoluzione di problemi non ben 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 utili per aumentare l'affidabilità. Tieni presente quanto segue quando si discute delle best practice:

- Ripristino automatico in caso di guasto: monitorando un carico di lavoro per gli indicatori chiave di prestazioni (KPI), puoi avviare l'automazione in caso di violazione 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-premises, spesso vengono eseguiti test per dimostrare che il carico di lavoro funziona in uno scenario specifico. I test non vengono in genere utilizzati per convalidare le strategie di ripristino. Nel cloud, puoi testare il modo in cui il carico di lavoro incorre nell'errore e convalidare le procedure di ripristino. Puoi utilizzare l'automazione per simulare diversi errori o ricreare scenari che in precedenza hanno portato a errori. Questo approccio presenta percorsi di errore che è possibile testare e correggere prima che si verifichi uno scenario di errore reale, riducendo così il rischio.
- Scalare a livello orizzontale per aumentare la disponibilità dei carichi di lavoro aggregati: sostituisci una risorsa grande con più risorse piccole per ridurre l'impatto di un singolo guasto sul carico di lavoro complessivo. Distribuisci le richieste tra più risorse di dimensioni inferiori per garantire che non condividano un punto di errore comune.
- Smetti di fare ipotesi sulla capacità: una causa comune di guasti nei carichi di lavoro on-premises è la saturazione delle risorse, quando le richieste assegnate a un carico di lavoro superano la capacità di quel carico di lavoro (questo è spesso l'obiettivo di 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 limiti, ma alcune quote possono essere controllate e altre possono essere gestite (consulta Gestione di Service Quotas e vincoli).
- Gestione del cambiamento tramite l''automazione: le modifiche all'infrastruttura andrebbero apportate utilizzando l'automazione. Le modifiche da gestire includono quelle all'automazione, che possono quindi essere monitorate e revisionate.

Definizioni

Il presente 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 dei guasti

Per garantire l'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 i guasti. Il carico di lavoro deve gestire le variazioni nella domanda o nei requisiti e deve essere progettato per rilevare il guasto e applicare autonomamente le correzioni in automatico.

Argomenti

Resilienza e 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 a seguito di interruzioni dell'infrastruttura o del servizio, acquisire in modo dinamico le risorse di calcolo per soddisfare la domanda e mitigare le interruzioni, quali configurazioni errate o problemi di rete transitori.

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 revisioni sulla 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 sono illustrati nei rispettivi pilastri 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 una metrica usata comunemente sia per misurare in termini quantitativi la resilienza sia come obiettivo target della resilienza.

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

Disponibile per l'uso significa che la funzione concordata viene eseguita correttamente quando occorre.

Questa percentuale si calcola su un periodo di tempo, ad esempio un mese, un anno o 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. Ecco la definizione di disponibilità:

$$Availability = \frac{Available\ for\ Use\ Time}{Total\ Time}$$

- la disponibilità è una percentuale del tempo di attività (come il 99,9%) per un periodo di tempo (di norma 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 del servizio pianificati (ad esempio, manutenzione pianificata) dal tempo totale nella formula del tempo totale. Tuttavia, questo approccio non è consigliato, poiché gli utenti probabilmente vorranno usare il tuo servizio in quei momenti.

Ecco una tabella relativa agli 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 presenta esempi dei tipi di applicazioni che di solito vediamo a ogni livello di disponibilità. In questo documento, ci riferiamo a questi valori.

Disponibilità	Indisponibilità massima (per anno)	Categorie dell'applicazione
99%	3 giorni 15 ore	Elaborazione batch, estrazione dati, trasferimento e caricamento di lavori
99,9%	8 ore 45 minuti	Strumenti interni come knowledge management, monitoraggio dei progetti
99,95%	4 ore 22 minuti	Commercio online, POS
99,99%	52 minuti	Carichi di lavoro di video e trasmissione
99,999%	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 non andate a buon fine e quelle andate a buon fine, invece del "tempo disponibile per l'utilizzo". In questo caso, è possibile usare il seguente calcolo:

$$Availability = \frac{Successful\ Responses}{Valid\ Requests}$$

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 tali periodi. In caso di assenza di richieste in un dato periodo, viene conteggiato come disponibile al 100% per il periodo in questione.

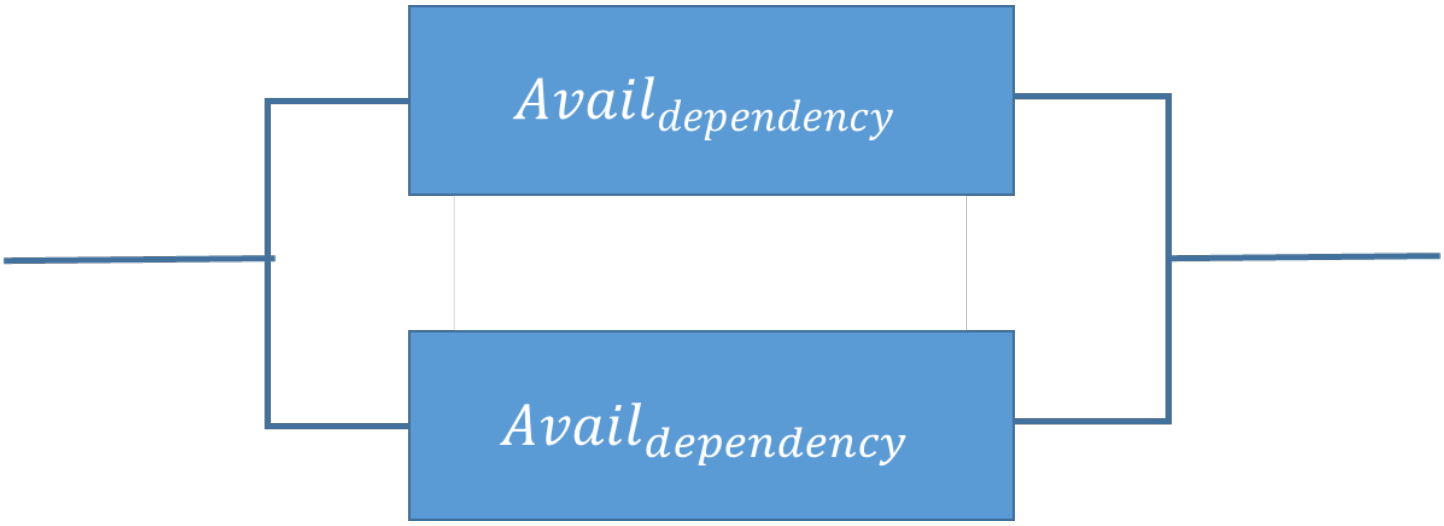
Calcolo della disponibilità con forti dipendenze. Molti sistemi presentano forti dipendenze con altri sistemi, nei quali un'interruzione in un sistema dipendente si traduce direttamente in un'interruzione del sistema di richiamo. Ciò si contrappone a una dipendenza leggera, in cui l'applicazione compensa un guasto del sistema dipendente. In presenza di tali forti dipendenze, la disponibilità del sistema di richiamo è il prodotto delle disponibilità dei sistemi dipendenti. Ad esempio, se disponi di un sistema progettato per una disponibilità del 99,99% con 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:

$Avail_{invok} \times Avail_{dep1} \times Avail_{dep2} = Avail_{workload}$

99,99% × 99,99% × 99,99% = 99,97%

Pertanto, è importante conoscere le tue dipendenze e i loro obiettivi di progettazione in termini di 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 si calcola come 100% meno il prodotto delle percentuali di guasto dei componenti. Ad esempio, se un sistema utilizza due componenti indipendenti, ciascuno con una disponibilità del 99,9%, la disponibilità effettiva di tale dipendenza è del 99,9999%:



$Avail_{effective} = Avail_{MAX} - ((100\% - Avail_{dependency}) \times (100\% - Avail_{dependency}))$

99,9999% = 100% − (0,1%×0,1%)

Calcolo abbreviato: se le disponibilità di tutti i componenti nel tuo calcolo consistono solamente di nove, allora puoi sommare il numero di nove per ottenere una 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 in termini di disponibilità per molti servizi AWS. Tuttavia, nei casi in cui esse non siano disponibili (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). È possibile stabilire una stima della disponibilità mediante:

$$Avail_{EST} = \frac{MTBF}{MTBF + MTTR}$$

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

Per ulteriori dettagli, consulta Availability and Beyond: Understanding and improving the resilience of distributed systems on AWS, utile per calcolare la tua disponibilità.

Costi per la disponibilità. La progettazione di applicazioni per livelli più elevati di disponibilità comporta in genere un aumento dei costi, quindi è opportuno identificare le reali esigenze in termini di disponibilità prima di iniziare la progettazione dell'applicazione. Elevati livelli di disponibilità impongono requisiti più severi per test e convalida in scenari di guasto completo. Questi richiedono l'automazione per il ripristino da tutti i tipi di guasti, oltre alla realizzazione e al test di tutti gli aspetti delle operazioni di sistema in modo simile, secondo gli stessi standard. Ad esempio, l'aggiunta o la rimozione di capacità, l'implementazione o il rollback del software aggiornato o le modifiche alla configurazione o la migrazione dei dati di sistema devono essere condotti nel rispetto dell'obiettivo di disponibilità desiderato. Oltre ai costi per lo sviluppo del software, con livelli di disponibilità molto elevati, l'innovazione ne risente a causa della necessità di procedere più lentamente nell'implementazione dei sistemi. Il suggerimento, pertanto, è di essere rigorosi nell'applicazione degli standard e nel considerare l'obiettivo di disponibilità adeguato 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 ad alta disponibilità riguarda la selezione delle dipendenze. A questi obiettivi più elevati, il set di software o servizi che è possibile scegliere come dipendenze diminuisce in base a quali di questi servizi hanno beneficiato di ingenti investimenti discussi 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 poiché questi ultimi sono più facili da valutare, testare e automatizzare e presentano un potenziale ridotto di interazioni imprevedibili con funzionalità incluse, ma inutilizzate.

Obiettivi di disaster recovery (DR)

Oltre agli obiettivi di disponibilità, la strategia di resilienza deve includere obiettivi di disaster recovery (DR) basati su strategie per ripristinare il carico di lavoro in caso di un evento di emergenza. Il disaster recovery 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 guasti di componenti, picchi di carico e bug di software.

Obiettivo del tempo di ripristino (RTO) definito dall'organizzazione. L'RTO è il ritardo massimo accettabile tra l'interruzione del servizio e il relativo ripristino. Questo determina ciò che viene considerato un intervallo di tempo accettabile in caso di indisponibilità del servizio.

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 si considera una perdita di dati accettabile tra l'ultimo punto di ripristino e l'interruzione del servizio.

Relazione tra RPO, RTO ed 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 su diversi eventi che influiscono sulla disponibilità in un periodo di tempo, mentre l'RTO è un obiettivo, o un valore massimo consentito, per un singolo evento che influisce sulla disponibilità.

Approfondimento sulle esigenze in termini di disponibilità

Di solito si pensa alla disponibilità di un'applicazione come a un singolo obiettivo per l'applicazione nel suo insieme. Tuttavia, a un'analisi più attenta, spesso scopriamo che alcuni aspetti di un'applicazione o di un servizio presentano 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 a quelle che modificano 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 valutarne i requisiti in termini di disponibilità. Il vantaggio di questo approccio è concentrare gli sforzi (e le spese) sulla disponibilità in base alle esigenze specifiche, piuttosto che progettare l'intero sistema in base ai requisiti più rigidi.

Raccomandazione

Valuta in modo critico gli aspetti unici delle tue applicazioni e, dove necessario, differenzia gli obiettivi di progettazione della disponibilità e del disaster recovery per riflettere le esigenze della tua azienda.

All'interno di AWS, dividiamo di solito i servizi in "piano dati" e "piano di controllo (control-plane)". Il piano dati è responsabile della fornitura del servizio in tempo reale mentre i piani di controllo (control-plane) consentono di configurare l'ambiente. Ad esempio, le istanze Amazon EC2, i database Amazon RDS e le operazioni di lettura/scrittura su tabelle di Amazon DynamoDB sono tutte operazioni sul piano 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 (control-plane). Sebbene elevati livelli di disponibilità siano importanti per tutte queste funzionalità, i piani dati hanno in genere obiettivi di progettazione di disponibilità più elevati rispetto ai piani di controllo (control-plane). Pertanto, i carichi di lavoro con requisiti di alta disponibilità dovrebbero evitare dipendenze di runtime su operazioni sul piano di controllo (control-plane).

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 si adottano le opportune iniziative di lavoro per progettare il sistema. AWS vanta una grande un'esperienza nella progettazione di applicazioni con una serie di obiettivi di progettazione della disponibilità, inclusi servizi con una disponibilità del 99,999% o superiore. AWS I Solution Architects (SA) possono aiutarti a effettuare una progettazione adeguata 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, ma in modo continuo per perfezionare la progettazione man mano che acquisisci esperienza operativa, impari da eventi reali e superi guasti di diversi tipi. È quindi possibile applicare l'iniziativa di lavoro opportuna per migliorare l'implementazione.

Le esigenze di disponibilità necessarie per un carico di lavoro devono essere allineate a esigenze e criticità aziendali. Definendo innanzitutto il framework di criticità aziendale con RTO, RPO e disponibilità definiti, è possibile valutare ciascun carico di lavoro. Tale approccio richiede che le persone coinvolte nell'implementazione del carico di lavoro conoscano framework e impatto del loro carico di lavoro 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, occorre stabilire i requisiti fondamentali che influenzano l'affidabilità. Ad esempio, è necessario disporre di una larghezza di banda della rete sufficiente verso il data center.

In un ambiente on-premises, questi requisiti possono causare tempi di consegna lunghi a causa delle dipendenze e pertanto devono essere incorporati durante la pianificazione iniziale. Tuttavia, grazie a AWS, 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 la libertà 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

Per le architetture di carichi di lavoro basate sul cloud, esistono quote di servizio (chiamate anche 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 lo spazio di archiviazione 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-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 Assicurarsi che esista un intervallo sufficiente tra le quote correnti e l'utilizzo massimo per consentire il failover

REL01-BP01 Conoscenza delle quote e dei 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 prevenire il degrado o l'interruzione del servizio Account AWS implementando linee guida appropriate per il monitoraggio delle metriche chiave, le revisioni dell'infrastruttura e le misure correttive dell'automazione per verificare che non vengano raggiunte le quote e i vincoli di servizio che potrebbero causare il degrado o l'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 procedano al dimensionamento automatico 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 dell'adozione di questa best practice: il monitoraggio e la gestione automatizzata di quote di servizio e vincoli di risorse consentono di ridurre in modo proattivo i guasti. Le modifiche 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 AWS servizio che ti aiuta a gestire le quote per oltre 250 AWS servizi da un'unica posizione. Oltre a cercare i valori delle quote, puoi anche richiedere e tenere traccia degli aumenti delle quote dalla console Service Quotas o utilizzando il. AWS SDK AWS Trusted Advisor offre un controllo delle quote di servizio che mostra l'utilizzo e le quote per alcuni aspetti di alcuni servizi. Le quote di servizio predefinite per servizio si trovano anche nella AWS documentazione del rispettivo servizio (ad esempio, vedi Amazon VPC Quotas).

Alcuni limiti di servizio, come i limiti di velocità per le limitazioni di velocità, APIs vengono impostati all'interno dello stesso Amazon API Gateway configurando un piano di utilizzo. Alcuni limiti impostati come configurazione sui rispettivi servizi includono ProvisionedIOPS, Amazon RDS storage allocated e Amazon EBS Volume Allocations. Amazon Elastic Compute Cloud dispone di un proprio pannello di controllo sulle restrizioni dei servizi 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 dell'applicazione e non sono adattabili alle tue esigenze, contattaci per verificare se esistono soluzioni Supporto di mitigazione.

Le quote di servizio possono essere specifiche per ogni regione o di natura globale. L'utilizzo AWS di un servizio che raggiunge la quota prevista non funzionerà come previsto in condizioni di utilizzo normale e potrebbe causare interruzioni o deterioramenti del servizio. Ad esempio, una quota di servizio limita il numero di EC2 istanze Amazon DL utilizzate in una regione. Tale limite può essere raggiunto durante un evento di scalabilità del traffico utilizzando i gruppi ASG Auto Scaling ().

Le quote di servizio per ogni account devono essere valutate in modo regolare per determinare i limiti di servizio opportuni per quell'account. Queste quote di servizio fungono da guardrail operativi, per evitare di fornire accidentalmente più risorse di quelle necessarie. Servono anche a limitare i tassi di richiesta sulle API operazioni 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 dalla stessa. Questi potrebbero essere la capacità di archiviazione (ad esempio, gp2 ha un limite di dimensione compreso tra 1 GB e 16 TB) o la velocità effettiva del disco. È essenziale che il vincolo di un tipo di risorsa sia progettato e valutato in modo costante per l'utilizzo che potrebbe raggiungere il suo limite. In caso di raggiungimento inaspettato di un vincolo, può verificarsi il degrado o l'interruzione delle applicazioni o dei servizi dell'account.

Se c'è un caso d'uso in cui le quote di servizio influiscono sulle prestazioni di un'applicazione e non possono essere adattate alle esigenze richieste, contattateci Supporto per verificare se esistono delle mitigazioni. Per ulteriori dettagli sull'adeguamento delle quote fisse, consulta REL01-BP03 Adattamento di quote e vincoli di servizio fissi mediante l'architettura.

Esistono numerosi AWS servizi e strumenti per aiutare a monitorare e gestire Service Quotas. Sfrutta il servizio e gli strumenti per fornire controlli automatizzati 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 CDKs offre metodi programmatici per gestire e monitorare automaticamente i livelli e l'utilizzo delle quote di servizio.

Passaggi dell'implementazione

Per Service Quotas:

- Rivedi AWS Service Quotas.
- Per conoscere le quote di servizio esistenti, determina i servizi (come IAM Access Analyzer) da utilizzare. Esistono circa 250 AWS servizi controllati da quote di servizio. Quindi stabilisci il nome della quota di servizio specifica utilizzabile all'interno di ogni account e regione. Esistono circa 3000 nomi di quote di servizio per regione.
- Amplia questa analisi delle quote AWS Config per trovare tutte le AWS risorse utilizzate nel tuo. Account AWS
- Usa AWS CloudFormation i dati per determinare le AWS risorse utilizzate. Guarda le risorse che sono state create con AWS Management Console o con il `list-stack-resources` AWS CLI comando. È anche possibile vedere le risorse configurate da implementare 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 a livello di programmazione da Trusted Advisor e Service Quotas.
- Stabilisci un metodo di monitoraggio automatizzato (consulta REL01-BP02 Gestisci le quote di servizio tra account e regioni e REL01-BP04 Monitoraggio e gestione delle quote) per ricevere avvisi e informazioni se le quote di servizio sono vicine al limite o lo hanno superato.
- Stabilisci un metodo automatizzato e programmatico per verificare se una quota di servizio ha subito modifiche in una regione ma non in altre nello stesso account (consulta REL01-BP02 Gestisci le quote di servizio tra account e regioni e REL01-BP04 Monitoraggio e gestione delle quote).
- Automatizza la scansione dei log e delle metriche delle applicazioni per determinare la presenza di errori di quota o di vincoli di servizio. In presenza di errori, invia gli allarmi al sistema di monitoraggio.
- Stabilisci procedure di progettazione per calcolare la modifica richiesta della quota (consulta REL01-BP05 Automazione della gestione delle quote) una volta individuata la necessità di quote più elevate per servizi specifici.
- Crea un flusso di lavoro di provisioning e di approvazione per richiedere modifiche alla quota di servizio, che 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 di servizio prima di fornire e utilizzare nuovi AWS servizi prima di implementarli in ambienti di produzione o caricati. (ad esempio, account per il test di carico).

Per i vincoli dei servizi:

- Stabilisci metodi di monitoraggio e metriche per avvisi in caso di avvicinamento da parte delle risorse ai relativi limiti. Sfruttalo in CloudWatch modo appropriato per le metriche o il monitoraggio dei log.
- Stabilisci soglie di allarme per ciascuna risorsa con 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 procedure e processi esistenti.

Risorse

Best practice correlate:

- REL01-BP02 Gestisci le quote di servizio tra 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 Assicurarsi che esista un intervallo sufficiente tra le quote correnti e l'utilizzo massimo per consentire il failover
- REL03-BP01 Scegli come segmentare il tuo 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-BP04 Test della resilienza tramite l'utilizzo dell'ingegneria del caos

Documenti correlati:

- AWS Il pilastro dell'affidabilità di Well-Architected Framework: disponibilità
- AWS Service Quotas (precedentemente denominate limiti di servizio)
- AWS Trusted Advisor Controlli relativi alle migliori pratiche (vedere la sezione Limiti del servizio)
- AWS limita il monitoraggio delle AWS risposte
- Limiti EC2 del servizio Amazon
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Guida per l'utente di Service Quotas
- Quota Monitor per AWS

- AWS Limiti di isolamento dei guasti
- Availability with redundancy
- AWS per i dati
- Cos'è l'integrazione continua?
- Cos'è la distribuzione continua?
- APNPartner: partner che possono aiutare nella gestione della configurazione
- Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS
- Gestione e monitoraggio della limitazione dei carichi di API lavoro
- Visualizza i AWS Trusted Advisor consigli su larga scala con AWS Organizations
- Automatizzazione degli aumenti dei limiti di servizio e del supporto aziendale con AWS Control Tower

Video correlati:

- AWS Live re:InForce 2019 - Service Quotas
- Visualizzazione e gestione delle quote per AWS i servizi tramite Service Quotas
- AWS IAMDimostrazione di Quotas

Strumenti correlati:

- CodeGuru Revisore Amazon
- 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 Gestisci le quote di servizio tra account e regioni

Se utilizzi più account o regioni, assicurati di richiedere le quote opportune in tutti gli ambienti di esecuzione dei carichi di lavoro di produzione.

Risultato desiderato: servizi e applicazioni non dovrebbero essere influenzati dall'esaurimento della quota di servizio per le configurazioni che si estendono su account o regioni o che presentano progetti di resilienza che utilizzano il failover di zona, regione o account.

Anti-pattern comuni:

- Si consente l'aumento dell'utilizzo delle risorse in una regione di isolamento senza alcun meccanismo per mantenere la capacità nelle altre.
- Impostazione manualmente tutte le quote in modo indipendente nelle regioni di isolamento.
- Mancata valutazione dell'effetto delle architetture di resilienza (come quelle attive o passive) nelle future esigenze di quote durante un degrado nella regione non primaria.
- Mancata valutazione regolare delle quote e applicazione dele modifiche necessarie in ogni regione e account in cui viene gestito il carico di lavoro.
- Mancato utilizzo dei modelli di richiesta di quote per la richiesta di aumenti in più regioni e account.
- Mancato aggiornamento delle quote dei servizi, perché si pensa erroneamente che l'aumento delle quote abbia implicazioni di costo, come le richieste di prenotazione di calcolo.

Vantaggi dell'adozione di questa best practice: verifica della capacità di gestire il carico corrente nelle regioni o negli account secondari in caso di indisponibilità 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

Il monitoraggio delle quote di servizio avviene per account. Salvo diversa indicazione, ogni quota è specifica. Regione AWS Oltre agli ambienti di produzione, gestisci anche le quote in tutti gli ambienti non di produzione applicabili, in modo che test e 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 un aumento dei carichi di lavoro in tutte le regioni dovuto all'implementazione di progetti che utilizzano approcci attivo/attivo, attivo/passivo con standby a caldo, attivo/passivo con standby a

freddo e attivo/passivo con Pilot Light, è essenziale conoscere tutti i livelli di quota di regione e account. 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.

Riconcilia tutte le differenze di quota di servizio tra le diverse regioni (regione attiva o passiva) e crea 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 delle giornate di gioco o table top potrebbero non riuscire a trovare le differenze nelle quote di servizio tra le regioni e a mantenere i limiti corretti.

La deviazione della quota di servizio, la condizione in cui la modifica dei limiti della quota di servizio per una determinata quota denominata avviene in una regione e non in tutte le regioni, è un fattore molto importante da monitorare e valutare. Si dovrebbe prendere in considerazione la possibilità di modificare la quota nelle regioni con traffico o potenzialmente in grado di trasportare traffico.

- Seleziona account e regioni pertinenti in base ai tuoi requisiti di servizio, latenza, normativi e 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. Confronta questi valori per individuare le differenze.

Passaggi dell'implementazione

- Rivedi i valori di 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 in modo corretto nelle regioni secondarie in caso di guasto nella regione primaria.
- Valuta in modo automatizzato 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, aggiorna i modelli di quote di servizio 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 Conoscenza delle quote e dei 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 Assicurarsi che esista un intervallo sufficiente tra le quote correnti e l'utilizzo massimo per consentire il failover
- REL03-BP01 Scegli come segmentare il tuo 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-BP04 Test della resilienza tramite l'utilizzo dell'ingegneria del caos

Documenti correlati:

- AWS Il pilastro dell'affidabilità di Well-Architected Framework: disponibilità
- AWS Service Quotas (precedentemente denominate limiti di servizio)
- AWS Trusted Advisor Controlli relativi alle migliori pratiche (vedere la sezione Limiti del servizio)
- AWS limita il monitoraggio delle AWS risposte
- Limiti EC2 del servizio Amazon
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Guida per l'utente di Service Quotas
- Quota Monitor per AWS

- AWS Limiti di isolamento dei guasti
- Availability with redundancy
- AWS per i dati
- Cos'è l'integrazione continua?
- Cos'è la distribuzione continua?
- APNPartner: partner che possono aiutare nella gestione della configurazione
- Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS
- Gestione e monitoraggio della limitazione dei carichi di API lavoro
- Visualizza i AWS Trusted Advisor consigli su larga scala con AWS Organizations
- Automatizzazione degli aumenti dei limiti di servizio e del supporto aziendale con AWS Control Tower

Video correlati:

- AWS Live re:INforce 2019 - Service Quotas
- Visualizzazione e gestione delle quote per AWS i servizi tramite Service Quotas
- AWS IAMDemo di Quotas

Servizi correlati:

- CodeGuru Revisore Amazon
- 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 con attenzione quote di servizio, vincoli del servizio e limiti delle risorse fisiche che non possono essere modificati. Progetta architetture per applicazioni e servizi in modo da impedire che questi limiti pregiudichino l'affidabilità.

Alcuni esempi includono la larghezza di banda della rete, le dimensioni di payload dell'invocazione di funzioni serverless, il tasso di espansione della limitazione (della larghezza di banda della rete) per un gateway API e le connessioni utente simultanee a un database.

Risultato desiderato: funzionamento dell'applicazione o del servizio come previsto in condizioni di traffico normale e intenso. 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 non scalabile o modificabile in caso di superamento delle quote di servizio fisse. Ad esempio, dimensioni dei payload SQS di 256 KB.
- Mancata progettazione e implementazione dell'osservabilità per monitorare e inviare avvisi circa le soglie per le quote di servizio a rischio durante eventi di traffico elevato.

Vantaggi dell'adozione di questa best practice: verifica del funzionamento dell'applicazione con tutti i livelli di carico dei servizi previsti senza interruzioni o deterioramenti.

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

Guida all'implementazione

Diversamente dalle risorse e dalle quote di servizio flessibili, sostituibili con unità di capacità maggiori, le quote di servizio fisse in AWS non possono essere modificate. Di conseguenza, occorre verificare tutti i servizi AWS di questo tipo per identificare i possibili limiti fissi di capacità in caso di relativo utilizzo per la progettazione di un'applicazione.

I limiti fissi vengono mostrati nella console di Service Quotas. Se le colonne visualizzano `ADJUSTABLE = No`, il servizio ha un limite fisso. I limiti fissi vengono mostrati anche in alcune pagine di configurazione delle risorse. Ad esempio, Lambda presenta 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 di un'esecuzione di Lambda superiore a 15 minuti. Se il codice potrebbe restare in esecuzione oltre questo limite della quota di servizio, devi prendere in considerazione tecnologie o progettazioni alternative. In caso di raggiungimento del limite dopo l'implementazione nell'ambiente di produzione, l'applicazione sarà soggetta a errori o interruzioni fino alla correzione. A differenza dalle quote flessibili, non esiste alcun metodo per modificare i limiti, anche in caso di eventi di emergenza con livello di gravità 1.

Una volta implementata l'applicazione in un ambiente di test, occorre adottare una strategia per determinare se l'eventuale probabilità di raggiungere i limiti fissi. I test di stress, di carico e di caos devono fare parte del piano di test iniziale.

Passaggi dell'implementazione

- Esamina l'elenco completo dei servizi AWS utilizzabili nella fase di progettazione dell'applicazione.
- Esamina i limiti di quota flessibili e fissi per tutti i servizi. Non tutti i limiti vengono mostrati nella console di Service Quotas. Alcuni servizi indicano tali limiti in posizioni alternative.
- 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 disaster recovery. Orienta il processo di identificazione del sistema distribuito corretto per il carico di lavoro in base a tali fattori commerciali e tecnologici.
- 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 Conoscenza delle quote e dei vincoli di servizio
- REL01-BP02 Gestisci le quote di servizio tra account e regioni
- REL01-BP04 Monitoraggio e gestione delle quote
- REL01-BP05 Automazione della gestione delle quote
- REL01-BP06 Assicurarsi che esista un intervallo sufficiente tra le quote correnti e l'utilizzo massimo per consentire il failover
- REL03-BP01 Scegli come segmentare il tuo 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-BP04 Test della resilienza tramite l'utilizzo dell'ingegneria del caos

Documenti correlati:

- Pilastro dell'affidabilità di AWS Well-Architected Framework: disponibilità
- AWS Service Quotas (precedentemente definite restrizioni dei servizi)
- AWS Trusted Advisor Best Practice Checks (consulta la sezione Service Limits)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Guida per l'utente di Service Quotas
- Quota Monitor for AWS
- Limiti di isolamento dei guasti di AWS
- Availability with redundancy
- AWS for Data
- Cos'è l'integrazione continua?
- Cos'è la distribuzione continua?
- Partner APN: partner per la gestione della configurazione
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS

- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Video correlati:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

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 opportuno per una crescita pianificata dell'utilizzo.

Risultato desiderato: implementazione di sistemi attivi e automatizzati per la gestione e il monitoraggio. Queste soluzioni operative indicano che le soglie di utilizzo delle quote stanno per

essere raggiunte. Il 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 degli allarmi solo per i servizi supportati da Service AWS Quotas e non monitoraggio di 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 multiregioni.

Vantaggi derivanti dall'adozione di questa best practice: il monitoraggio automatico delle AWS Service Quotas e il monitoraggio dell'utilizzo rispetto a tali quote ti consentiranno di vedere quando ti stai avvicinando a un limite di quota. Puoi usare questi dati di monitoraggio per limitare eventuali errori dovuti all'esaurimento della quota.

Livello di rischio associato se questa best practice non fosse adottata: 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. Questi allarmi possono essere richiamati da, funzioni AWS Config Lambda, Amazon CloudWatch o da. AWS Trusted Advisor Puoi anche utilizzare i filtri metrici sui CloudWatch registri per cercare ed estrarre modelli nei log per determinare se l'utilizzo si avvicina alle soglie di quota.

Passaggi dell'implementazione

Per il monitoraggio:

- Acquisisci informazioni sull'attuale consumo di risorse, ad esempio bucket o istanze. Utilizza API le operazioni di servizio, come Amazon EC2 `DescribeInstances`API, per raccogliere l'attuale consumo di risorse.
- Acquisisci le attuali quote essenziali e valide per i servizi usando:
  - AWS Service Quotas
  - AWS Trusted Advisor
  - AWS documentazione
  - AWS pagine specifiche del servizio
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- Utilizza AWS Service Quotas, un AWS servizio che ti aiuta a gestire le quote per oltre 250 AWS servizi da un'unica posizione.
- Utilizza i limiti Trusted Advisor di servizio per monitorare i tuoi attuali limiti di servizio a varie soglie.
- Utilizza 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:

- Automatizzato: imposta una regola AWS Config personalizzata per scansionare le quote di servizio tra le regioni e confrontarle per individuare le differenze.
- Automatica: configura una funzione Lambda personalizzata per analizzare le quote di servizio tra regioni e confrontarle per individuare le differenze.
- Manuale: scansiona la quota dei servizi tramite AWS CLI o AWS Console per scansionare le quote di servizio tra le regioni e confrontarle per individuare eventuali differenze. API Segnala le differenze.
- In caso di individuazione di 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 Conoscenza delle quote e dei vincoli di servizio
- REL01-BP02 Gestisci le quote di servizio tra 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 Assicurarsi che esista un intervallo sufficiente tra le quote correnti e l'utilizzo massimo per consentire il failover
- REL03-BP01 Scegli come segmentare il tuo 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-BP04 Test della resilienza tramite l'utilizzo dell'ingegneria del caos

Documenti correlati:

- AWS Il pilastro dell'affidabilità di Well-Architected Framework: disponibilità
- AWS Service Quotas (precedentemente denominate limiti di servizio)
- AWS Trusted Advisor Controlli relativi alle migliori pratiche (vedere la sezione Limiti del servizio)
- AWS limita il monitoraggio delle AWS risposte
- Limiti EC2 del servizio Amazon
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Guida per l'utente di Service Quotas
- Quota Monitor per AWS
- AWS Limiti di isolamento dei guasti
- Availability with redundancy
- AWS per i dati
- Cos'è l'integrazione continua?
- Cos'è la distribuzione continua?

- APNPartner: partner che possono aiutare nella gestione della configurazione
- Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS
- Gestione e monitoraggio della limitazione dei carichi di API lavoro
- Visualizza i AWS Trusted Advisor consigli su larga scala con AWS Organizations
- Automatizzazione degli aumenti dei limiti di servizio e del supporto aziendale con AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Video correlati:

- AWS Live re:InForce 2019 - Service Quotas
- Visualizzazione e gestione delle quote per AWS i servizi tramite Service Quotas
- AWS IAMDimostrazione di Quotas
- AWS re:Invent 2018: Chiudere i circuiti e aprire le menti: 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

Service Quotas, chiamate anche limiti nei servizi AWS, sono i valori massimi per le risorse dell'Account AWS. Ogni servizio AWS definisce un set di quote e i relativi valori predefiniti. Per fornire

al carico di lavoro l'accesso a tutte le risorse necessarie, potrebbe essere necessario aumentare i valori di Service Quotas.

L'aumento del consumo delle risorse AWS da parte del carico di lavoro può minacciare la stabilità del carico di lavoro e avere un impatto sull'esperienza dell'utente in caso di superamento delle quote. Implementa strumenti che segnalano quando il carico di lavoro si avvicina ai limiti e valuta la possibilità di creare automaticamente richieste di aumento delle quote.

Risultato desiderato: le quote sono configurate in modo appropriato per i carichi di lavoro in esecuzione in ciascun Account AWS e Regione.

Anti-pattern comuni:

- Non riesci a considerare e regolare le quote in modo appropriato per soddisfare i requisiti del carico di lavoro.
- Tieni traccia delle quote e dell'utilizzo mediante metodi che possono diventare obsoleti, come ad esempio i fogli di calcolo.
- Aggiorni i limiti di servizio solo in base a pianificazioni periodiche.
- L'organizzazione non dispone di processi operativi per rivedere le quote esistenti e richiedere aumenti di Service Quotas quando necessario.

Vantaggi dell'adozione di questa best practice:

- Maggiore resilienza del carico di lavoro: eviti gli errori causati dal superamento delle quote di risorse AWS.
- Disaster recovery semplificato: puoi riutilizzare i meccanismi di gestione automatica delle quote creati nella Regione primaria durante la configurazione del disaster recovery in un'altra Regione AWS.

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

Guida all'implementazione

Visualizza le quote correnti e tieni traccia del consumo di quote in corso attraverso meccanismi quali la console AWS Service Quotas, AWS Command Line Interface (AWS CLI) e gli AWS SDK. Inoltre, puoi integrare i database di gestione della configurazione (CMDB) e i sistemi di gestione dei servizi IT (ITSM) con le API di AWS Service Quotas.

Genera avvisi automatici se l'utilizzo delle quote raggiunge le soglie definite e definisci un processo per presentare richieste di aumento della quota quando ricevi avvisi. Se il carico di lavoro sottostante è critico per l'azienda, puoi automatizzare le richieste di aumento della quota, ma esegui il test dell'automazione con cautela per evitare il rischio di un'azione incontrollata, come un ciclo di feedback della crescita.

Gli aumenti di quota più piccoli vengono spesso approvati automaticamente. È possibile che per le richieste di quote più grandi sia richiesta l'elaborazione manuale a livello di Supporto AWS e che il tempo richiesto per la revisione e l'elaborazione sia maggiore. Prevedi un tempo aggiuntivo per l'elaborazione di più richieste o richieste di grandi incrementi.

Passaggi dell'implementazione

- Implementa il monitoraggio automatico di Service Quotas e invia avvisi se la crescita dell'utilizzo delle risorse del carico di lavoro si avvicina ai limiti delle quote. Ad esempio, Monitoraggio delle quote per AWS può fornire il monitoraggio automatico di Service Quotas. Questo strumento si integra con AWS Organizations e si distribuisce utilizzando Cloudformation StackSets in modo che i nuovi account vengano monitorati automaticamente al momento della creazione.
- Utilizza funzionalità come i modelli di richiesta Service Quotas o AWS Control Tower per semplificare la configurazione di Service Quotas per nuovi account.
- Crea dashboard dell'utilizzo attuale di Service Quotas in tutti gli Account AWS e le Regioni e fai riferimenti ad esse, se necessario, per evitare di superare le quote. La Dashboard Trusted Advisor Organizational (TAO), che fa parte delle Dashboard di cloud intelligence, permette di iniziare rapidamente a usare una dashboard di questo tipo.
- Tieni traccia delle richieste di aumento dei limiti di servizio. Consolidated Insights from Multiple Accounts (CIMA) può fornire una visione a livello di organizzazione di tutte le richieste.
- Verifica la generazione di avvisi e l'automazione delle richieste di aumento della quota impostando soglie di quota più basse negli account non di produzione. Non eseguire questi test in un account di produzione.

Risorse

Best practice correlate:

- OPS10-BP07 Automazione delle risposte agli eventi

Documenti correlati:

- Partner APN: partner per la gestione della configurazione
- Marketplace AWS: prodotti CMDB per il monitoraggio delle restrizioni
- AWS Service Quotas (precedentemente definite restrizioni dei servizi)
- AWS Trusted Advisor Best Practice Checks (consulta la sezione Service Limits)
- Quota Monitor Solution on AWS - AWS Solution
- What is Service Quotas?
- What is Service Quotas request templates?

Video correlati:

- AWS Live re:Inforce 2019 - Service Quotas
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Strumenti correlati:

- Quota Monitor for AWS

REL01-BP06 Assicurarsi che esista un intervallo sufficiente tra le quote correnti e l'utilizzo massimo per consentire il failover

Il presente articolo illustra come mantenere lo spazio tra la quota di risorse e l'utilizzo e i relativi vantaggi per la tua organizzazione. Una volta terminato l'utilizzo di una risorsa, la quota di utilizzo può continuare a essere conteggiata per tale risorsa, con possibile conseguenza di una risorsa in errore o inaccessibile. Evita tale errore nelle risorse verificando che le quote tengano conto della sovrapposizione di risorse in errore o inaccessibili e della rispettiva sostituzione. Prendi in considerazioni casi come errori della rete, errori della zona di disponibilità o errori della regione durante il calcolo di questo divario.

Risultato desiderato: è possibile coprire piccoli o grandi errori nelle risorse o nell'accessibilità delle risorse entro le 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 della stabilità statica.
- Calcolo della quota totale necessaria per ogni regione senza tenere conto delle potenziali risorse inaccessibili.
- Non sono stati presi in considerazione i limiti di isolamento dagli errori di AWS servizio per alcuni servizi e i loro potenziali modelli di utilizzo anomali.

Vantaggi dell'adozione di questa best practice: in caso di eventi di interruzione del servizio che influiscono sulla disponibilità dell'applicazione, utilizza il cloud per implementare strategie di ripristino da tali eventi. Un esempio di strategia consiste nella creazione di risorse aggiuntive per sostituire quelle inaccessibili e soddisfare le condizioni di failover senza esaurire il limite del servizio.

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

Guida all'implementazione

Nel valutare un limite di quota, tieni conto dei casi di failover che possono verificarsi a causa di un peggioramento della situazione. Considera i casi di failover seguenti:

- Un file interrotto o inaccessibile. VPC
- Sottorete inaccessibile.
- Zona di disponibilità degradata che influisce sull'accessibilità delle risorse.
- Diversi instradamenti di rete o punti di ingresso e uscita bloccati o modificati.
- Impatto di una regione degradata sull'accessibilità delle risorse.
- Errore in un sottoinsieme di risorse in una regione o in una zona di disponibilità.

La decisione relativa all'avvio del failover è unica per ogni situazione, in quanto l'impatto aziendale può variare. Gestisci la pianificazione della capacità delle risorse nella posizione di failover e le quote delle risorse prima di decidere di effettuare il failover di un'applicazione o di un servizio.

Prendi in considerazione i picchi di attività più elevati del normale nell'esame delle quote per ciascun servizio. Questi picchi potrebbero essere correlati a risorse ancora attive ma inaccessibili a causa di reti o autorizzazioni. Le risorse attive non terminate vengono conteggiate rispetto al limite di quota del servizio.

Passaggi dell'implementazione

- Mantieni uno spazio 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. Tieni conto di modelli di implementazione tipici, requisiti di disponibilità e crescita dei consumi.
- Richiedi aumenti delle quote, se necessario. Prevedi un tempo di attesa per la richiesta di aumento della quota.
- Determina i requisiti di affidabilità, noti anche come numero di 9.
- Analizza i potenziali scenari di errore, come la perdita di un componente, di una zona di disponibilità o di una regione.
- Stabilisci la metodologia di implementazione (ad esempio, canary, blu/verde, rosso/nero e rolling).
- Includi un buffer appropriato rispetto al limite della quota attuale. Un esempio di buffer potrebbe essere del 15%.
- Includi calcoli per la stabilità statica (zonale e regionale) laddove appropriato.
- Pianifica la crescita dei consumi e monitora i trend di consumo.
- 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 l'utilizzo di prenotazioni della capacità on demand per pianificare la capacità in anticipo rispetto a qualsiasi failover. Si tratta di una strategia utile da implementare per le pianificazioni aziendali 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 Conoscenza delle quote e dei vincoli di servizio
- REL01-BP02 Gestisci le quote di servizio tra 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 Scegli come segmentare il tuo 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-BP04 Test della resilienza tramite l'utilizzo dell'ingegneria del caos

Documenti correlati:

- AWS Il pilastro dell'affidabilità di Well-Architected Framework: disponibilità
- AWS Service Quotas (precedentemente denominate limiti di servizio)
- AWS Trusted Advisor Controlli relativi alle migliori pratiche (vedere la sezione Limiti del servizio)
- AWS limita il monitoraggio delle AWS risposte
- Limiti EC2 del servizio Amazon
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Guida per l'utente di Service Quotas
- Quota Monitor per AWS
- AWS Limiti di isolamento dei guasti
- Availability with redundancy
- AWS per i dati
- Cos'è l'integrazione continua?
- Cos'è la distribuzione continua?
- APNPartner: partner che possono aiutare nella gestione della configurazione
- Gestione del ciclo di vita dell'account in ambienti SaaS account-per-tenant su AWS
- Gestione e monitoraggio della limitazione dei carichi di API lavoro
- Visualizza i AWS Trusted Advisor consigli su larga scala con AWS Organizations
- Automatizzazione degli aumenti dei limiti di servizio e del supporto aziendale con AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Video correlati:

- AWS Live re:INforce 2019 - Service Quotas
- Visualizzazione e gestione delle quote per AWS i servizi tramite Service Quotas

- AWS IAMDimostrazione di Quotas
- AWS re:Invent 2018: Circuiti 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 accessibili pubblicamente sia privati) e, possibilmente, l'infrastruttura del data center esistente. I piani devono includere considerazioni di rete, ad esempio connettività intrasistema e intersistema, gestione di indirizzi IP e 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 rispetto a 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 isolata e privata dell'AWS Cloud, dove è possibile avviare 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 Fornisci connettività ridondante tra reti private nel cloud e ambienti on-premise

- REL02-BP03 Garantire che l'allocazione della sottorete IP tenga conto dell'espansione e della disponibilità
- REL02-BP04 Preferire topologie hub-and-spoke rispetto a mesh da-molti-a-molti
- REL02-BP05 Applica intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi di 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 tuo 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: la pianificazione, la realizzazione e la messa in funzione di una connettività di rete altamente disponibile per i tuoi endpoint pubblici è fondamentale. Se il carico di lavoro diventa irraggiungibile a causa della perdita di connettività, il sistema apparirà ai clienti come non funzionante, 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, a un'architettura resiliente per il carico di lavoro stesso, puoi offrire ai clienti la disponibilità e il livello di servizio migliori possibili.

AWS Global Accelerator, Amazon CloudFront, Gateway Amazon API, funzione URL AWS Lambda, API AWS AppSync ed Elastic Load Balancing (ELB) forniscono tutti endpoint pubblici a elevata disponibilità. Amazon Route 53 fornisce un servizio DNS ad alta disponibilità per la risoluzione dei nomi di dominio, così da verificare la possibilità di risolvere gli indirizzi degli endpoint pubblici.

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 throughput 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 l'uso di 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 ridurre la disponibilità per gli utenti.

Vantaggi dell'adozione di questa best practice: la progettazione pensata per una connettività di rete a elevata disponibilità e resilienza garantisce l'accessibilità e la disponibilità del carico di lavoro agli utenti.

Livello di rischio associato se questa best practice non fosse adottata: 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. Utilizza un sistema dei nomi di dominio (DNS) altamente scalabile e disponibile, 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 l'elevata disponibilità effettiva del carico di lavoro, utilizza Elastic Load Balancing (ELB). Amazon Route 53 consente di indirizzare il traffico verso ELB, che lo distribuisce alle istanze di calcolo di destinazione. Puoi anche usare Gateway Amazon API insieme a AWS Lambda per una soluzione serverless. I clienti possono anche eseguire carichi di lavoro in più Regioni AWS. Grazie a un pattern attivo/attivo multisito, il carico di lavoro può servire il traffico proveniente da più regioni. Con un pattern attivo/passivo multisito, il carico di lavoro serve il traffico proveniente dalla regione attiva, mentre nella regione secondaria avviene la replica dei dati, che diventano attivi in caso di

guasto nella regione primaria. I controlli dell'integrità di Route 53 consentono dunque di controllare il failover DNS da qualsiasi endpoint in una regione primaria a un endpoint in una regione secondaria, verificando la raggiungibilità e la disponibilità del carico di lavoro 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 si migliora anche la disponibilità dell'applicazione poiché il carico dei contenuti viene spostato dai server alle 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 così 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 permettono il reindirizzamento del traffico degli utenti a endpoint integri che offrono le prestazioni e la disponibilità migliori agli utenti. Inoltre, AWS Global Accelerator presenta una progettazione di isolamento degli errori che usa due indirizzi IPv4 statici gestiti da zone di rete indipendenti, migliorando la disponibilità delle applicazioni.

Per proteggere i clienti dagli attacchi DDoS, AWS offre AWS Shield Standard. Shield Standard si attiva in automatico e protegge dagli attacchi comuni all'infrastruttura (livello 3 e 4) come i flood SYN/UDP e gli attacchi di riflessione in modo da supportare l'elevata disponibilità delle applicazioni su 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 prendere in considerazione l'uso di 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 DNS a elevata disponibilità: Amazon Route 53 è un servizio Web di sistema dei nomi di dominio (DNS) altamente scalabile e disponibile. Route 53 collega le richieste degli utenti alle applicazioni Internet eseguite su AWS oppure on-premises. Per ulteriori informazioni, consulta configuring Amazon Route 53 as your DNS service.
2. Configura controlli dell'integrità: quando usi Route 53, verifica che solo le destinazioni integre siano risolvibili. Inizia con la creazione dei controlli dell'integrità di Route 53 e la configurazione del 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 Amazon Route 53 DNS
3. Connessione del servizio DNS agli endpoint.
    a. In caso di utilizzo di Elastic Load Balancing come target per il tuo traffico, crea un record di alias mediante Amazon Route 53 che punti all'endpoint regionale del tuo sistema bilanciatore del carico. Durante la creazione del record di alias, imposta l'opzione Valutazione dello stato target su Sì.
    b. In caso di utilizzo di API Gateway, per i carichi di lavoro serverless o le API private, usa Route 53 per indirizzare il traffico verso l'API Gateway.
4. Opta per una rete di distribuzione di contenuti (CDN).
    a. Per la distribuzione di contenuti mediante posizioni edge più vicine all'utente, esamina il modo in cui CloudFront distribuisce i contenuti.
    b. Inizia partendo con una distribuzione CloudFront semplice. 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. Come funziona la memorizzazione nella cache con le posizioni edge di CloudFront
        ii. Aumento della percentuale di richieste eseguite direttamente dalle cache CloudFront (percentuale di riscontri nella cache)
        iii. Utilizzo dello scudo di origine Amazon CloudFront
        iv. Ottimizzazione dell'elevata disponibilità con il failover di origine 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 una conoscenza più approfondita, scopri come funziona AWS WAF e

quando sarà tutto pronto per implementare le protezioni dai flood HTTP POST e GET a livello dell'applicazione, consulta Getting started with AWS WAF. Puoi anche utilizzare AWS WAF con CloudFront. Consulta la documentazione su come funziona AWS WAF con le 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 accesso a Internet in esecuzione su Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator e Amazon Route 53, puoi prendere in considerazione AWS Shield Advanced ed esaminare gli esempi di architetture resilienti agli attacchi DDoS. Per proteggere carico di lavoro ed endpoint pubblici dagli attacchi DDoS, consulta Getting started with AWS Shield Advanced.

Risorse

Best practice correlate:

- REL10-BP01 Implementazione del carico di lavoro in diversi luoghi
- REL11-BP04 Affidati al piano dati e non 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?
- What is Amazon CloudFront?
- What is Amazon Route 53?
- Cos'è l'Elastic Load Balancing?
- Network Connectivity capability - Establishing Your Cloud Foundations
- What is Amazon API Gateway?
- What are AWS WAF, AWS Shield, and AWS Firewall Manager?
- What is Amazon Application Recovery Controller?

- Configure custom health checks for DNS failover

Video correlati:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2020: Global traffic management with Amazon Route 53
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure
- AWS re:Invent 2022 - Dive deep on networking infrastructure

Esempi correlati:

- Disaster Recovery with Amazon Application Recovery Controller (ARC)
- Workshop AWS Global Accelerator

REL02-BP02 Fornisci connettività ridondante tra reti private nel cloud e 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.
- Si utilizza solo uno o più VPN tunnel che terminano nella stessa zona di disponibilità.
- Ti affidi a uno ISP per la VPN connettività, che può portare a guasti completi durante ISP le interruzioni.
- Non implementate protocolli di routing dinamici come quelli BGP fondamentali per reindirizzare il traffico durante le interruzioni della rete.
- Ignorate i limiti di larghezza di banda dei VPN tunnel e sopravvalutate le loro capacità di backup.

Vantaggi dell'adozione di questa best practice: implementando una connettività ridondante tra il tuo ambiente cloud e l'ambiente aziendale oppure on-premises, puoi garantire l'affidabilità delle comunicazioni dei servizi dipendenti tra due ambienti.

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 locale a AWS, è possibile ottenere la massima resilienza della rete (SLAdel 99,99%) utilizzando connessioni separate che terminano su dispositivi distinti in più di una posizione locale e più di una posizione. AWS Direct Connect Questa topologia offre resilienza ai guasti dei dispositivi, ai problemi di connettività e alle interruzioni complete della posizione. In alternativa, è possibile ottenere un'elevata resilienza (SLAdel 99,9%) utilizzando due connessioni individuali a più sedi (ciascuna sede locale connessa a un'unica posizione Direct Connect). Questo approccio offre protezione dalle interruzioni della connettività causate da interruzioni della fibra o guasti dei dispositivi e aiuta a mitigare le interruzioni complete della posizione. Il AWS Direct Connect Resiliency Toolkit può aiutarti a progettare la topologia. AWS Direct Connect

Puoi anche prendere in considerazione l'idea di AWS Site-to-Site VPN terminare con un backup economico sulla tua connessione principale. AWS Transit Gateway AWS Direct Connect Questa configurazione consente il routing multipath (ECMP) a parità di costo su più VPN tunnel, con una velocità di trasmissione fino a 50 Gbps, anche se ogni tunnel è limitato a 1,25 Gbps. VPN È importante notare, tuttavia, che questa è ancora la scelta più efficace per ridurre al minimo le interruzioni di rete e AWS Direct Connect fornire una connettività stabile.

Quando utilizzi VPNs Internet per connettere l'ambiente cloud al data center locale, configura due VPN tunnel come parte di un'unica connessione. site-to-site VPN 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 più connessioni Internet da vari provider di servizi Internet (ISPs) presso la tua sede locale per evitare interruzioni complete della VPN connettività dovute a una singola interruzione. ISP La scelta ISPs con routing e infrastrutture diverse, in particolare quelle con percorsi fisici separati verso gli AWS endpoint, offre un'elevata disponibilità di connettività.

Oltre alla ridondanza fisica con più AWS Direct Connect connessioni e più VPN tunnel (o una combinazione di entrambi), è fondamentale anche l'implementazione del routing dinamico del Border Gateway Protocol (BGP). Dynamic BGP fornisce il reindirizzamento automatico del traffico da un percorso all'altro in base alle condizioni di rete in tempo reale e alle politiche 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

- Acquisizione di connettività ad alta disponibilità tra e AWS l'ambiente locale.
  - Utilizza più AWS Direct Connect connessioni o VPN tunnel tra reti private distribuite separatamente.
  - Utilizza più AWS Direct Connect sedi per un'elevata disponibilità.
  - Se ne utilizzi più Regioni AWS, crea ridondanza in almeno due di esse.
- AWS Transit GatewayUtilizzatelo, quando possibile, per terminare la VPN connessione.
- Valuta i Marketplace AWS dispositivi a cui terminare VPNs o estendere la WAN porta SD- AWS. Se utilizzi Marketplace AWS appliance, implementa istanze ridondanti per un'elevata disponibilità in diverse zone di disponibilità.
- Fornisci una connessione ridondante all'ambiente on-premises.
  - Potresti aver bisogno di connessioni ridondanti a più connessioni per soddisfare le tue esigenze di disponibilità Regioni AWS .
  - Usa l'AWS Direct Connect Resiliency Toolkit per iniziare.

Risorse

Documenti correlati:

- AWS Direct Connect Raccomandazioni sulla resilienza
- Utilizzo di Site-to-Site VPN connessioni ridondanti per fornire il failover
- Politiche e comunità di routing BGP
- Configurazioni attive/attive e attive/passive in AWS Direct Connect
- APNPartner: partner che possono aiutarti a pianificare la tua rete
- Marketplace AWS per l'infrastruttura di rete
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Creazione di un'infrastruttura multirete scalabile e sicura VPC AWS
- Utilizzo di connessioni ridondanti Site-to-Site VPN per fornire il failover
- Utilizzo del AWS Direct Connect Resiliency Toolkit per iniziare
- VPCEndpoint e servizi per gli VPC endpoint ()AWS PrivateLink
- Che cos'è AmazonVPC?
- What is a transit gateway?

- Che cos'è AWS Site-to-Site VPN?
- Working with Direct Connect gateways

Video correlati:

- AWS re:Invent 2018: VPC design avanzato e nuove funzionalità per Amazon VPC
- AWS re:Invent 2019: architetture di riferimento per molti AWS Transit Gateway VPCs

REL02-BP03 Garantire che l'allocazione della sottorete IP tenga conto dell'espansione e della disponibilità

Gli intervalli di indirizzi VPC IP di Amazon devono essere sufficientemente ampi da soddisfare i requisiti dei carichi di lavoro, incluso il calcolo delle future espansioni e dell'allocazione degli indirizzi IP alle sottoreti nelle zone di disponibilità. Ciò include sistemi di bilanciamento del carico, istanze e applicazioni basate su contenitori. EC2

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 del RFC 1918) devono essere assegnati per ciascuno. VPC Nell'ambito di questo processo, soddisfa i seguenti requisiti:

- Consenti lo spazio degli indirizzi IP per più di uno VPC per regione.
- All'interno di aVPC, consenti spazio per più sottoreti in modo da poter coprire più zone di disponibilità.
- Valuta la possibilità di lasciare dello spazio inutilizzato nei CIDR blocchi VPC per future espansioni.
- Assicurati che sia disponibile uno spazio di indirizzi IP per soddisfare le esigenze di eventuali flotte transitorie di EC2 istanze Amazon che potresti utilizzare, ad esempio flotte Spot per l'apprendimento automatico, cluster Amazon o cluster Amazon EMR Redshift. Un'analoga considerazione dovrebbe essere data ai cluster Kubernetes, come Amazon Elastic Kubernetes Service (EKSAmazon), poiché a ogni pod Kubernetes viene assegnato un indirizzo routabile dal blocco per impostazione predefinita. VPC CIDR
- Tieni presente che i primi quattro indirizzi IP e l'ultimo indirizzo IP in ogni blocco di sottorete sono riservati e non disponibili per l'uso. CIDR
- Tieni presente che il VPC CIDR blocco iniziale assegnato al tuo VPC non può essere modificato o eliminato, ma puoi aggiungere ulteriori blocchi non sovrapposti CIDR a. VPC La sottorete IPv4 CIDRs non può essere modificata, tuttavia sì. IPv6 CIDRs

- Il VPC CIDR blocco più grande possibile è /16 e il più piccolo è /28.
- Prendi in considerazione altre reti connesse (VPClocali o altri provider di servizi cloud) e assicurati che lo spazio degli indirizzi IP non si sovrapponga. Per ulteriori informazioni, consulta REL02-BP05 Applica intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi di 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 considerare la crescita futura, con conseguenti CIDR blocchi troppo piccoli che richiedono una riconfigurazione, con conseguenti potenziali tempi di inattività.
- Stima erronea del numero di indirizzi IP utilizzabili da un bilanciatore del carico elastico.
- Distribuzione di numerosi bilanciatori del carico a traffico elevato nelle stesse sottoreti.
- Utilizzo di meccanismi di dimensionamento automatico senza monitorare il consumo di indirizzi IP.
- Definire CIDR intervalli eccessivamente ampi ben oltre le aspettative di crescita future, il che può comportare difficoltà nel 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 nell'aumentare verticalmente.

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

Guida all'implementazione

Pianifica la tua 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 le aree Account AWS e le regioni pertinenti in base ai requisiti di servizio, di latenza, normativi e di disaster recovery (DR).
- Identifica le tue esigenze per le VPC implementazioni regionali.
- Identifica la dimensione di. VPCs
  - Determina se intendi implementare la connettività multipla. VPC
    - What Is a Transit Gateway?

    - Connettività multipla a regione singola VPC
- Stabilisci se hai bisogno della segregazione delle reti a causa di requisiti normativi.
- Crea VPCs con CIDR blocchi di dimensioni adeguate per soddisfare le tue esigenze attuali e future.
    - Se avete proiezioni di crescita sconosciute, potreste optare per CIDR blocchi più grandi per ridurre il potenziale di future riconfigurazioni.
- Prendi in considerazione l'utilizzo dell'IPv6indirizzamento per le sottoreti come parte di un dual-stack. VPC IPv6è ideale per essere utilizzato in sottoreti private contenenti flotte di istanze o contenitori temporanei che altrimenti richiederebbero un gran numero di indirizzi. IPv4

Risorse

Best practice Well-Architected correlate:

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

Documenti correlati:

- APNPartner: partner che possono aiutarti a pianificare la tua rete
- Marketplace AWS per l'infrastruttura di rete
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity
- Connettività multipla a regione singola VPC
- Che cos'è AmazonVPC?
- IPv6su AWS
- IPv6sulle architetture di riferimento
- Amazon Elastic Kubernetes Service lancia il supporto IPv6
- Consigli per i tuoi sistemi Classic Load Balancer VPC
- Subnet delle zone di disponibilità - Application Load Balancer
- Zone di disponibilità - Network Load Balancer

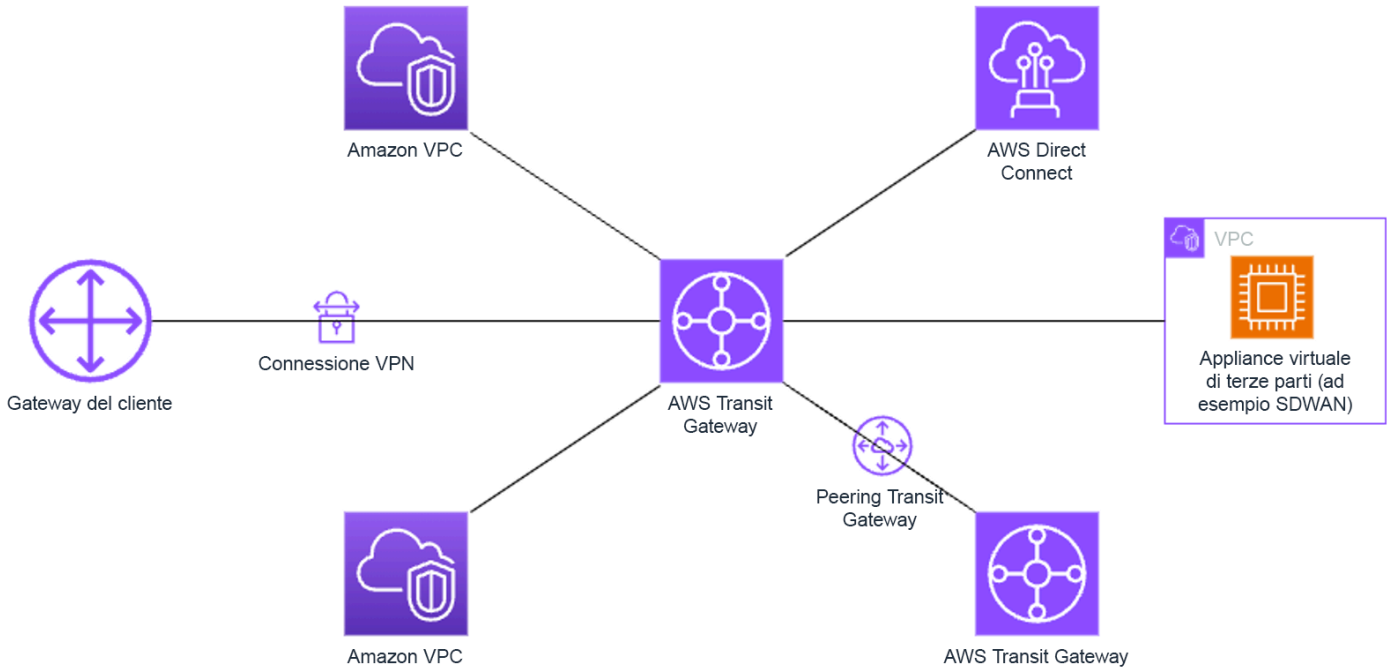
Video correlati:

- AWS re:Invent 2018: VPC design avanzato e nuove funzionalità per Amazon VPC (03) NET3
- AWS re:Invent 2019: architetture di AWS Transit Gateway riferimento per molti (06-R1) VPCs NET4
- AWS re:Invent 2023: pronti per il futuro? AWS Progettare reti per la crescita e la flessibilità (0) NET31

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 on-premises, 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. Funge 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.



Risultato desiderato: hai connesso i cloud privati virtuali (VPC) e le reti on-premises tramite un hub centrale. Configuri le connessioni peering tramite l'hub, che funge da router cloud a scalabilità elevata. L'instradamento è semplificato perché non è necessario lavorare con complesse relazioni di peering. Il traffico tra le reti è crittografato ed è possibile isolare le reti.

Anti-pattern comuni:

- Crei regole di peering di rete complesse.
- Fornisci instradamenti tra reti che non devono comunicare tra loro (ad esempio, carichi di lavoro separati che non hanno interdipendenze).
- La governance dell'istanza dell'hub è inefficace.

Vantaggi dell'adozione di questa best practice: con l'aumento del numero di reti connesse, la gestione e l'espansione della connettività mesh diventa sempre più impegnativa. Un'architettura mesh introduce ulteriori sfide, come componenti aggiuntivi dell'infrastruttura, requisiti di configurazione e considerazioni sull'implementazione. La mesh introduce inoltre costi operativi aggiuntivi per gestire e monitorare i componenti del piano dati e del piano di controllo (control-plane). Devi pensare a come fornire disponibilità elevata dell'architettura mesh, monitorare l'integrità e le prestazioni della mesh e gestire gli aggiornamenti dei componenti della mesh.

Un modello hub-and-spoke, invece, stabilisce un instradamento centralizzato del traffico su più reti. Consente un approccio più semplice alla gestione e al monitoraggio dei componenti del piano dati e del piano di controllo (control-plane).

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

Guida all'implementazione

Crea un account dei Servizi di rete se non esiste. Posiziona l'hub nell'account dei Servizi di rete dell'organizzazione. Questo approccio consente ai tecnici della rete di gestire centralmente l'hub.

L'hub del modello hub-and-spoke funge da router virtuale per il traffico che scorre tra i cloud privati virtuali (VPC) e le reti on-premises. Questo approccio riduce la complessità della rete e facilita la risoluzione dei problemi di rete.

Considera la progettazione della rete, inclusi i VPC, AWS Direct Connect e le connessioni VPN sito-sito che desideri interconnettere.

Considera l'utilizzo di una sottorete separata per ciascun collegamento VPC del gateway di transito. Per ogni sottorete, utilizza un piccolo CIDR (ad esempio /28), in modo da avere più spazio di indirizzi

per le risorse di elaborazione. Inoltre, crea una lista di controllo degli accessi alla rete e associala a tutte le sottoreti collegate all'hub. Mantieni aperta la lista di controllo degli accessi di rete in entrata e in uscita.

Progetta e implementa le tabelle di routing in modo che gli instradamenti siano forniti solo tra le reti che devono comunicare. Ometti gli instradamenti tra reti che non devono comunicare tra loro (ad esempio, tra carichi di lavoro separati che non hanno interdipendenze).

Passaggi dell'implementazione

1. Pianifica la rete. Determina le reti che desideri connettere e verifica che non condividano intervalli CIDR sovrapposti.
2. Crea un AWS Transit Gateway e collega i VPC.
3. Se necessario, crea connessioni VPN o gateway Direct Connect e associali a Transit Gateway.
4. Definisci come viene instradato il traffico tra i VPC connessi e altre connessioni tramite la configurazione delle tabelle di routing di Transit Gateway.
5. Usa Amazon CloudWatch per monitorare e modificare come necessario le configurazioni per l'ottimizzazione delle prestazioni e dei costi.

Risorse

Best practice correlate:

- REL02-BP03 Verifica che l'allocazione delle sottoreti IP consenta l'espansione e la disponibilità
- REL02-BP05 Applicazione di intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi con indirizzi privati a cui sono connessi

Documenti correlati:

- What Is a Transit Gateway?
- Transit gateway design best practices
- Realizzazione di un'infrastruttura di reti AWS multi-VPC sicura e scalabile
- Building a global network using AWS Transit Gateway Inter-Region peering
- Opzioni di connettività di Amazon Virtual Private Cloud
- Partner APN: partner per la pianificazione della rete
- Marketplace AWS per l'infrastruttura di rete

Video correlati:

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities

Workshop correlati:

- Workshop su AWS Transit Gateway

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

Gli intervalli di indirizzi IP di ciascuno di voi non VPCs devono sovrapporsi in caso di peering, connessione tramite Transit Gateway o connessione tramite rete. VPN Evita i conflitti di indirizzi IP tra ambienti a VPC e locali o con altri provider di servizi cloud che utilizzi. Bisogna inoltre disporre di una soluzione per allocare gli intervalli di indirizzi IP privati quando necessario. Un sistema di gestione degli indirizzi IP (IPAM) può aiutare ad automatizzare questa operazione.

Risultato desiderato:

- Nessun conflitto di intervalli di indirizzi IP tra VPCs ambienti locali 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:

- Utilizzate lo stesso intervallo di IP che avete VPC in sede, nella rete aziendale o in altri provider di servizi cloud
- Non si tiene traccia degli intervalli di IP VPCs utilizzati per distribuire i carichi di lavoro.
- Ricorso a processi manuali di gestione degli indirizzi IP, come i fogli di calcolo.
- CIDRBlocchi sovradimensionati o sottodimensionati, con conseguente spreco di indirizzi IP o spazio di indirizzamento insufficiente per il carico di lavoro.

Vantaggi dell'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 se questa best practice non fosse adottata: medio

Guida all'implementazione

Utilizza un programmaIPAM, come Amazon VPC IP Address Manager, per monitorare e gestire il tuo CIDR utilizzo. Diversi IPAMs sono disponibili anche presso Marketplace AWS. Valuta il tuo potenziale utilizzo AWS, aggiungi CIDR intervalli a quelli esistenti VPCs e crea VPCs per consentire una crescita pianificata dell'utilizzo.

Passaggi dell'implementazione

- Rileva CIDR il consumo di corrente (ad esempio, VPCs e le sottoreti).
  - Utilizza API le operazioni di servizio per raccogliere il consumo correnteCIDR.
  - Usa Amazon VPC IP Address Manager per scoprire le risorse.
- Misura l'utilizzo attuale delle sottoreti.
  - Utilizza API le operazioni di servizio per raccogliere sottoreti per ogni VPC regione.
  - Usa Amazon VPC IP Address Manager per scoprire 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 migrare verso una nuova gamma di indirizzi o prendere in considerazione l'utilizzo di tecniche come il NATgateway privato o AWS PrivateLinkse devi connettere gli intervalli sovrapposti.

Risorse

Best practice correlate:

- Protezione delle reti

Documenti correlati:

- APNPartner: partner che possono aiutarti a pianificare la tua rete
- Marketplace AWS per l'infrastruttura di rete
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity

- Connecting Networks with Overlapping IP Ranges
- Che cos'è AmazonVPC?
- Che cos'èIPAM?

Video correlati:

- AWS re:Invent 2023 - VPC Design avanzati e nuove funzionalità
- AWS re:Invent 2019: architetture di riferimento per molti AWS Transit Gateway VPCs
- AWS re:Invent 2023 - Pronti per il futuro? Designing networks for growth and flexibility
- AWS re:Invent 2021 - {New Launch} Gestisci i tuoi indirizzi IP su larga scala 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 del Framework Well-Architected. Per l'affidabilità, è necessario seguire modelli specifici.

Le sezioni seguenti illustrano le best practice da utilizzare con questi modelli per l'affidabilità.

Argomenti

Progettazione dell'architettura del servizio di carico di lavoro

Crea 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 da integrarle rapidamente 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 presentano 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 richiamati centinaia di microservizi per creare porzioni discrete della pagina. Sebbene ci siano alcuni servizi che devono essere disponibili per fornire prezzo e 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 Scegli come segmentare il tuo carico di lavoro
- REL03-BP02 Crea servizi incentrati su domini e funzionalità aziendali specifici
- REL03-BP03 Fornire contratti di assistenza per API

REL03-BP01 Scegli come segmentare il tuo carico di lavoro

La segmentazione del carico di lavoro è importante nel determinare i requisiti di resilienza dell'applicazione. L'architettura monolitica va evitata, se possibile. Valuta invece con particolare attenzione quali componenti dell'applicazione possono essere suddivisi in microservizi. A seconda dei requisiti dell'applicazione, questa potrebbe finire per essere una combinazione di un'architettura orientata ai servizi () con microservizi, ove possibile. SOA I carichi di lavoro stateless sono più idonei all'implementazione come microservizi.

Risultato desiderato: i carichi di lavoro devono essere supportabili, scalabili e caratterizzati dal maggiore accoppiamento debole 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 scalare le risorse. Durante la rifattorizzazione di un monolito esistente, 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 guasto verificatosi in un componente genera un guasto 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.

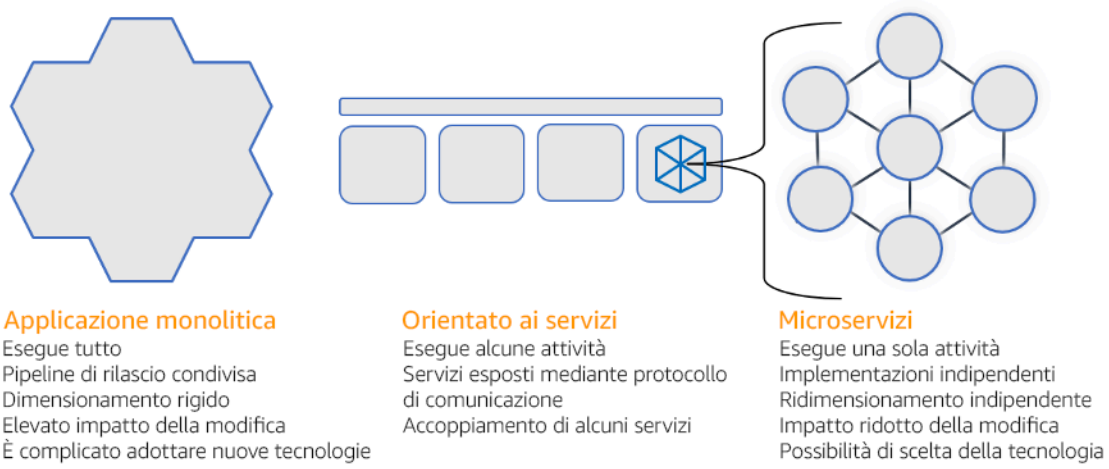
- Responsabilità ben definite per i team che supportano il carico di lavoro.

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

Guida all'implementazione

Scegli il tipo di architettura in base al tipo di segmentazione del carico di lavoro. Scegliete un'SOAarchitettura a microservizi (o, in alcuni rari casi, un'architettura monolitica). Anche se scegli di iniziare con un'architettura monolitica, devi assicurarti che sia modulare e che alla fine possa evolversi verso i nostri microservizi man mano che il prodotto cresce con l'SOAadozione da parte degli utenti. SOAe i microservizi offrono rispettivamente una segmentazione più ridotta, che è preferibile in un'architettura moderna, scalabile e affidabile, ma ci sono dei compromessi da considerare, soprattutto quando si implementa 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 l'implementazione di più componenti di indipendenza.



Architettura monolitica, orientata ai servizi e di microservizi

Passaggi dell'implementazione

- Determina l'architettura più opportuna per rifattorizzare o creare l'applicazione. SOAe i microservizi offrono rispettivamente una segmentazione più piccola, preferibile come architettura moderna,

scalabile e affidabile. SOApuò essere un buon compromesso per ottenere una segmentazione più piccola evitando al contempo alcune delle complessità dei microservizi. Per ulteriori dettagli, consulta Microservice Trade-Offs.

- 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 Implementazione dei microservizi su. AWS
- Valuta l'idea di attenerti al modello Strangler Fig per rifattorizzare un monolite in componenti più piccoli. Ciò comporta la sostituzione graduale di componenti applicativi specifici con nuove applicazioni e servizi. AWS Migration Hub Refactor Spaces funge da punto di partenza per procedere a rifattorizzare in modo incrementale. Per ulteriori dettagli, consulta Seamlessly migrate on-premises legacy workloads using a strangler pattern.
- L'implementazione dei microservizi può richiedere un meccanismo di rilevamento dei servizi per consentire a questi servizi distribuiti di comunicare tra loro. AWS App Meshpuò essere utilizzato con architetture orientate ai servizi per fornire l'individuazione e l'accesso affidabili ai servizi. AWS Cloud Mappuò essere utilizzato anche per l'individuazione dinamica dei servizi. DNS
- Se stai migrando da un monolite a, Amazon SOA MQ può aiutarti a colmare il divario come bus di 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 business unit, schema di accesso o struttura dei dati. A questo punto del processo di refactoring, dovresti scegliere di procedere con un tipo di database relazionale o non relazionale (No). SQL Per maggiori dettagli, consulta From to No. SQL SQL

Livello di impegno per il piano di implementazione: elevato

Risorse

Best practice correlate:

- REL03-BP02 Crea servizi incentrati su domini e funzionalità aziendali specifici

Documenti correlati:

- Amazon API Gateway: configurazione di un REST API utilizzo di Open API
- Cosa si intende per SOA (architettura orientata ai servizi)?
- Bounded Context (un modello centrale in Domain-Driven Design)

- Implementazione di microservizi su AWS
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- Microservizi attivi AWS
- Che cos'è AWS App Mesh?

Esempi correlati:

- Iterative App Modernization Workshop

Video correlati:

- Offrire l'eccellenza con i microservizi attivi AWS

REL03-BP02 Crea servizi incentrati su domini e funzionalità aziendali specifici

Le architetture orientate ai servizi (SOA) definiscono servizi con funzioni ben delineate definite in base alle 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 limitati e i progetti di sistema si spostano verso architetture a microservizi. SOA In caso di rifattorizzazione dei monoliti, 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:

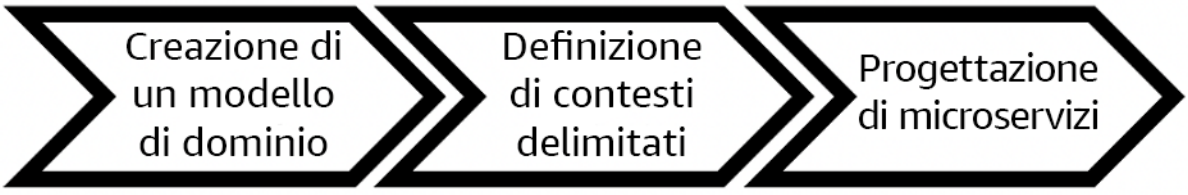
- I team sono formati su domini tecnici specifici come UI e UX, middleware (software intermediario) o database anziché su domini aziendali specifici.
- Le applicazioni coprono le responsabilità di dominio. I servizi che coprono contesti delimitati possono essere più difficili da gestire, richiedere maggiori 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: elevato

Guida all'implementazione

La progettazione basata sul dominio (DDD) è l'approccio fondamentale per la progettazione e la creazione di software in base 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.



Progettazione basata sul dominio

Passaggi dell'implementazione

- I team possono organizzare workshop di event storming per identificare rapidamente eventi, comandi, aggregati e domini in un formato leggero simile a quello delle note adesive.
- Una volta create le entità e le funzioni di dominio in un contesto di dominio, puoi suddividere il dominio in servizi mediante il contesto delimitato, che raggruppa entità con funzionalità e attributi simili. 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.
- La scomposizione dei monoliti in microservizi delinea i modelli per rifattorizzare i microservizi. L'utilizzo di modelli per la decomposizione in base a capacità aziendale, sottodominio o transazione si allinea bene agli approcci basati sul dominio.
- Le tecniche tattiche, come il bubble context, consentono di introdurre applicazioni esistenti o legacy senza riscritture iniziali e impegni completi DDD in tal senso. DDD In un approccio basato sul contesto delle bolle, si crea un contesto ristretto e delimitato mediante un livello di mappatura e coordinamento dei servizi o il livello anticorruzione, 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 sfruttare i AWS servizi per implementare la loro 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 mantenere i servizi concentrati sulla risoluzione dei problemi aziendali.
- Seleziona i servizi AWS ideali per i requisiti del tuo dominio aziendale e l'architettura dei 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 in 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 AWS i servizi aiutano i team a concentrarsi sulle funzionalità del dominio e a migliorare la qualità del software.

Risorse

Best practice correlate:

- REL03-BP01 Scegli come segmentare il tuo carico di lavoro
- REL03-BP03 Fornire contratti di assistenza per API

Documenti correlati:

- AWS Microservizi
- Implementazione di microservizi su AWS
- How to break a Monolith into Microservices
- Come iniziare DDD quando sei circondato da sistemi legacy
- Domain-Driven Design: Tackling Complexity in the Heart of Software
- Costruire architetture esagonali su AWS
- Decomposing monoliths into microservices
- Event Storming
- Messages Between Bounded Contexts
- Microservices
- Sviluppo basato su test
- Sviluppo basato sul comportamento

Esempi correlati:

- Progettazione di microservizi nativi per il cloud su (da/) AWS DDD EventStormingWorkshop

Strumenti correlati:

- Cloud AWS Database
- Serverless attivo AWS
- Contenitori presso AWS

REL03-BP03 Fornire contratti di assistenza per API

I contratti di assistenza sono accordi documentati tra API produttori e consumatori definiti in una definizione leggibile da una macchinaAPI. Una strategia di controllo delle versioni contrattuali consente ai consumatori di continuare a utilizzare le applicazioni esistenti API e di migrare le proprie applicazioni a una versione più recente quando sono pronte. API L'implementazione da parte del produttore può avvenire in qualsiasi momento, purché il processo sia conforme al contratto. I team di assistenza possono utilizzare lo stack tecnologico di loro scelta per soddisfare il contratto. API

Risultato desiderato: le applicazioni create con architetture orientate ai servizi o ai microservizi sono in grado di funzionare in modo indipendente pur avendo una dipendenza di runtime integrata. Le modifiche apportate a un API consumatore o a un produttore non interrompono la stabilità dell'intero sistema quando entrambe le parti seguono un contratto comune. API I componenti che comunicano tramite servizio APIs possono eseguire rilasci funzionali indipendenti, aggiornamenti alle dipendenze di runtime o eseguire il failover su un sito di disaster recovery (DR) con un impatto reciproco minimo o nullo. Inoltre, i servizi discreti sono in grado di scalare in modo indipendente assorbendo la richiesta di risorse senza che gli altri servizi debbano ridurre orizzontalmente di conseguenza.

Anti-pattern comuni:

- Creazione di servizi APIs senza schemi fortemente tipizzati. Ciò comporta APIs che non può essere utilizzato per generare API associazioni e payload che non possono essere convalidati programmaticamente.
- Non adottare una strategia di controllo delle versioni, che costringerebbe gli API utenti ad aggiornare e rilasciare o fallire quando i contratti di assistenza si evolvono.
- 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 API contratti per sviluppare casi di test e API implementazioni fittizie per consentire test indipendenti dei componenti del servizio.

Vantaggi derivanti dall'adozione di questa best practice: i sistemi distribuiti composti da componenti che comunicano tramite contratti di API assistenza possono migliorare l'affidabilità. Gli sviluppatori

possono individuare potenziali problemi nelle prime fasi del processo di sviluppo grazie al controllo del tipo durante la compilazione per verificare che le richieste e le risposte siano conformi al API contratto e che i campi obbligatori siano presenti. APIi contratti forniscono una chiara interfaccia di autodocumentazione APIs e forniscono una migliore interoperabilità tra diversi sistemi e linguaggi di programmazione.

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

Guida all'implementazione

Dopo aver identificato i domini aziendali e determinato la segmentazione del carico di lavoro, puoi sviluppare il tuo servizio. APIs Innanzitutto, definisci contratti di assistenza leggibili automaticamente e poi implementa una strategia di controllo delle APIs versioni. API Quando sei pronto per integrare i servizi tramite protocolli comuni come REST GraphQL o eventi asincroni, puoi incorporare AWS servizi nella tua architettura per integrare i componenti con contratti fortemente tipizzati. API

AWS APIservizi per contratti di assistenza

Incorpora AWS servizi tra cui Amazon API Gateway e Amazon EventBridge nella tua architettura per utilizzare i contratti di API servizio nella tua applicazione. AWS AppSync Amazon API Gateway ti aiuta a integrarti con AWS servizi nativi diretti e altri servizi Web. APIGateway supporta le APIspecifiche e il controllo delle versioni Open. AWS AppSync è un endpoint GraphQL gestito che puoi configurare 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 i tuoi eventi.

Passaggi dell'implementazione

- Innanzitutto, definisci un contratto per il tuo. API Un contratto esprimerà le capacità di un e API definirà oggetti di dati e campi fortemente tipizzati per l'APIinput e l'output.
- Quando esegui la configurazione APIs in API Gateway, puoi importare ed esportare API le specifiche aperte per i tuoi endpoint.
  - L'importazione di una API definizione aperta semplifica la creazione della propria definizione API e può essere integrata con l' AWS infrastruttura come strumenti di codice come la e. AWS Serverless Application ModelAWS Cloud Development Kit (AWS CDK)
  - L'esportazione di una API definizione semplifica l'integrazione con gli strumenti di API test e fornisce ai consumatori di servizi una specifica di integrazione.

- Puoi definire e gestire GraphQL APIs AWS AppSync definendo un file di schema GraphQL per generare l'interfaccia del contratto e semplificare l'interazione con REST modelli complessi, più tabelle di database o servizi legacy.
- AWS Amplifyi progetti integrati AWS AppSync generano file di JavaScript query fortemente tipizzati da utilizzare nella tua applicazione e una libreria client AWS AppSync GraphQL per le tabelle Amazon DynamoDB.
- Quando utilizzi gli eventi di servizio di Amazon EventBridge, gli eventi aderiscono a schemi già esistenti nel registro degli schemi o definiti con Open API Spec. Con uno schema definito nel registro, puoi anche generare associazioni client dal contratto dello schema per integrare il codice con gli eventi.
- Estendere o modificare il tuo. API L'estensione di un API è un'opzione più semplice quando si aggiungono campi che possono essere configurati con campi opzionali o valori predefiniti per i campi obbligatori.
  - JSONi contratti basati su protocolli come REST GraphQL possono essere adatti per l'estensione del contratto.
  - XMLi contratti basati su protocolli, ad esempio, SOAP dovrebbero essere testati con i consumatori di servizi per determinare la fattibilità dell'estensione del contratto.
- Quando si effettua il versionamento di unAPI, è consigliabile implementare il controllo delle versioni proxy, in cui viene utilizzata una facciata per supportare le versioni in modo che la logica possa essere mantenuta in un'unica base di codice.
  - Con API Gateway puoi utilizzare le mappature di richiesta e risposta per semplificare l'assorbimento delle modifiche contrattuali, stabilendo una facciata per fornire valori predefiniti per nuovi campi o per eliminare i campi rimossi da una richiesta o risposta. Con questo approccio, il servizio sottostante può avere un'unica base di codice.

Risorse

Best practice correlate:

- REL03-BP01 Scegli come segmentare il tuo carico di lavoro
- REL03-BP02 Crea servizi incentrati su domini e funzionalità aziendali specifici
- REL04-BP02 Implementare dipendenze liberamente accoppiate
- REL05-BP03 Controlla e limita le chiamate di nuovo tentativo
- REL05-BP05 Imposta i timeout dei client

Documenti correlati:

- Che cos'è una API (interfaccia di programmazione delle applicazioni)?
- Implementazione di microservizi su AWS
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- Microservizi attivi AWS
- Utilizzo delle estensioni API Gateway to Open API
- Apri API - Specificazione
- GraphQL: schemi e tipi
- Associazioni di EventBridge codice Amazon

Esempi correlati:

- Amazon API Gateway: configurazione di un REST API utilizzo di Open API
- Da Amazon API Gateway all'applicazione Amazon CRUD DynamoDB tramite Open API
- Modelli di integrazione delle applicazioni moderni nell'era senza server: API Gateway Service Integration
- Implementazione del controllo delle versioni API Gateway basato su header con Amazon CloudFront
- AWS AppSync: Building a client application

Video correlati:

- Utilizzo di Open API in AWS SAM per gestire Gateway API

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 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 di prevenire gli errori e migliorare il tempo medio tra guasti (MTBF).

Best practice

- REL04-BP01 Identificazione del tipo di sistema distribuito da cui si dipende
- REL04-BP02 Implementare dipendenze liberamente accoppiate
- REL04-BP03 Fai un lavoro costante
- REL04-BP04 Rendere idempotenti le operazioni di mutazione

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: progettazione di un carico di lavoro in grado di interagire in modo efficace con 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 effettuano chiamate reciproche 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: non è insolito che un determinato carico di lavoro implementi uno o più stili di comunicazione tra sincroni, asincroni 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) in termini 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 così da verificare che la tua dipendenza fornisca un servizio ai livelli richiesti dal 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 gamma di sfide in grado di far aumentare complessità dell'architettura, carico operativo e costi. Le sfide più comuni includono latenza, interruzioni della rete, perdita dei dati, scalabilità e ritardo nella replica dei dati.
- Implementa una gestione e una creazione di log affidabili degli errori per risolvere i problemi quando si verificano quelli legati alle dipendenze.

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 ulteriori 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, la guida per gli sviluppatori AWS Lambda illustra come gestire tentativi ed errori nel richiamare le funzioni Lambda.
- Imposta i timeout per le chiamate delle dipendenze da parte del carico di lavoro. Questa tecnica evita di aspettare troppo a lungo o all'infinito una risposta. Per un'utile discussione su questo argomento, 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 sincrona

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 relativa dipendenza, inviando e ricevendo messaggi tramite un broker di messaggi. Lo streaming di eventi consente al carico di lavoro e alla relativa dipendenza di utilizzare un servizio di streaming per la pubblicazione e l'abbonamento a eventi, forniti come flussi continui di dati, da elaborare 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 sono in grado di elaborare messaggi ad alta priorità prima di quelli normali. Nello streaming di eventi, tutti i messaggi presentano la stessa priorità.
  - Consumo di messaggi: i broker di messaggi garantiscono la ricezione del messaggio da parte dei consumatori. Gli utenti che utilizzano lo streaming di eventi devono tenere traccia dell'ultimo messaggio letto.
  - Ordinamento dei messaggi: utilizzando la messaggistica, la ricezione dei messaggi nell'ordine esatto di invio non è garantita, salvo in caso di un approccio first-in-first-out (FIFO). Lo streaming di eventi mantiene sempre l'ordine in cui i dati sono stati prodotti.
  - Eliminazione dei messaggi: con la messaggistica, il consumatore deve eliminare il messaggio dopo la relativa elaborazione. 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 procede a richiamare una funzione Lambda in modo asincrono, Lambda inserisce la richiesta in una coda e restituisce una risposta di esito positivo senza ulteriori informazioni. Al termine dell'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 sottolineare che i servizi di messaggistica o streaming recapiteranno di nuovo un messaggio in caso di errore di rete o mancata ricezione della 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. La documentazione AWS Lambda illustra come gestire gli errori di 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. Amazon CloudWatch ti consente di monitorare i servizi di messaggistica e streaming di eventi. Puoi anche dotare di strumenti il tuo carico di lavoro con AWS X-Ray per ottenere informazioni utili rapidamente per la risoluzione dei problemi.

Dipendenza dal 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, come Amazon Simple Storage Service (Amazon S3), Amazon Elastic File System (Amazon EFS) e Amazon FSx per Lustre, in modo che il tuo carico di lavoro possa leggere e scrivere file su larga scala.
- Se il tuo carico di lavoro deve richiamare più processi batch, puoi sfruttare AWS Step Functions per semplificare l'orchestrazione dei processi batch eseguiti in AWS o on-premises. Questo progetto di esempio mostra 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
- Messaggi AWS
- Cosa sono i flussi di dati?
- AWS Lambda Developer Guide: Asynchronous invocation
- Amazon Simple Queue Service FAQ: FIFO queues

- 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
- Esempi AWS su GitHub: AWS Step functions Complex Orchestrator App
- Guida per l'utente AWS Batch: 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 Service (Amazon S3)
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx per Lustre
- AWS Step Functions
- AWS Batch

REL04-BP02 Implementare dipendenze liberamente accoppiate

Le dipendenze come sistemi di accodamento, sistemi di streaming, flussi di lavoro e bilanciatori del carico sono 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.

Il disaccoppiamento delle dipendenze, come i sistemi di coda, quelli di streaming e i flussi di lavoro, favorisce la riduzione al minimo dell'impatto sul sistema di modifiche o guasti. Tale separazione isola il comportamento di un componente dall'impatto sugli altri dipendenti dallo stesso, migliorando resilienza e 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. Garantisce inoltre una resilienza granulare a livello di componente in cui è possibile aumentare orizzontalmente 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' point-to-pointinterazione diretta, ma di solito attraverso un livello di storage intermedio durevole, come una SQS coda Amazon, una piattaforma di dati di streaming come Amazon Kinesis o. AWS Step Functions

Figura 4: dipendenze come sistemi di accodamento e bilanciatori del carico con accoppiamento debole

Amazon mette in SQS coda e AWS Step Functions sono solo due modi per aggiungere uno strato intermedio per l'accoppiamento libero. Le architetture basate sugli eventi possono anche essere create utilizzando Cloud AWS Amazon EventBridge, che può astrarre i clienti (produttori di eventi) dai servizi su cui fanno affidamento (consumatori di eventi). Amazon Simple Notification Service (AmazonSNS) è una soluzione efficace quando è necessaria una messaggistica basata su push ad alta velocità. many-to-many Utilizzando SNS gli argomenti di Amazon, i tuoi sistemi di pubblicazione possono inviare messaggi a un gran numero di endpoint di abbonati 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: riduzione al minimo l'area della superficie in caso di guasto a livello di componente, supportando così diagnostica e risoluzione dei problemi, grazie all'implementazione di

dipendenze con accoppiamento debole. 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 aumentare orizzontalmente 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.
- Richiamo diretto APIs tra livelli di carico di lavoro senza possibilità 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 di 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 se questa best practice non fosse adottata: elevato

Guida all'implementazione

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

- Crea architetture basate sugli eventi: EventBridgeAmazon ti aiuta a creare architetture basate sugli eventi liberamente accoppiate e distribuite.
- Implementazione di code in sistemi distribuiti: puoi utilizzare Amazon Simple Queue Service SQS (Amazon) per integrare e disaccoppiare sistemi distribuiti.
- Containerizza i componenti come microservizi: i microservizi consentono ai team di creare applicazioni composte da piccoli componenti indipendenti che comunicano in modo ben definito.

APIs Amazon Elastic Container Service (AmazonECS) e Amazon Elastic Kubernetes Service (EKSAmazon) possono aiutarti a iniziare a usare i container più velocemente.

- Gestisci i flussi di lavoro con Step Functions: Step Functions ti aiuta a coordinare più AWS servizi in flussi di lavoro flessibili.
- Sfrutta le architetture di messaggistica publish-subscribe (pub/sub): Amazon Simple Notification Service (Amazon SNS) fornisce il recapito 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 sugli eventi con Amazon EventBridge
  - AWS re:Invent 2022 - Progettazione di integrazioni basate sugli eventi con Amazon EventBridge
- I sistemi di messaggistica distribuiti sono composti da tre parti principali che devono essere implementate per un'architettura basata su code. Includono componenti del sistema distribuito, la coda utilizzata per il disaccoppiamento (distribuita sui SQS server Amazon) 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 SQS server Amazon per motivi di ridondanza.
  - SQSArchitettura Amazon di base
  - Send Messages Between Distributed Applications with 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 su AWS
  - Let's Architect! Architecting microservices with containers
- Con AWS Step Functions puoi creare applicazioni distribuite, automatizzare i processi, orchestrare microservizi, tra le altre cose. L'orchestrazione di più componenti in un flusso di lavoro automatizzato consente di disaccoppiare le dipendenze nell'applicazione.
  - Crea un flusso di lavoro serverless con e AWS Step FunctionsAWS Lambda
  - Guida introduttiva con AWS Step Functions

Risorse

Documenti correlati:

- AmazonEC2: garantire l'idempotenza
- The Amazon Builders' Library: difficoltà dei sistemi distribuiti
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- Che cos'è Amazon EventBridge?
- What Is Amazon Simple Queue Service?
- Break up with your monolith
- Orchestra i microservizi basati sulle code con Amazon AWS Step Functions SQS
- SQSArchitettura Amazon di base
- Queue-Based Architecture

Video correlati:

- AWS New York Summit 2019: introduzione alle architetture basate sugli eventi e ad Amazon (05) EventBridge MAD2
- AWS re:Invent 2018: Close Loops and Opening Minds: Come prendere il controllo di sistemi, grandi e piccoli ARC337 (include accoppiamento libero, lavoro costante, stabilità statica)
- AWS re:Invent 2019: Passaggio ad architetture basate sugli eventi (08) SVS3
- AWS re:Invent 2019: applicazioni scalabili senza server basate su eventi con Amazon e Lambda SQS
- AWS re:Invent 2022 - Progettazione di integrazioni basate sugli eventi con Amazon EventBridge
- AWS re:Invent 2017: Approfondimento e best practice su Elastic Load Balancing

REL04-BP03 Fai un lavoro costante

I sistemi possono presentare guasti 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 dell'integrità esegue un lavoro costante con modifiche rapide e di piccole dimensioni.

Ad esempio, se il sistema di controllo dell'integrità 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 dell'integrità sarebbe sovraccarico nel tentativo di aggiornare i sistemi di notifica e comunicare lo stato con i client. Pertanto, il sistema di controllo dell'integrità dovrebbe inviare ogni volta lo snapshot completo dello stato attuale. 100.000 stati di integrità del server, ciascuno rappresentato da un bit, equivarrebbero a un payload di soli 12,5 KB. Indipendentemente dal fatto che non ci siano server guasti, o che lo siano tutti, il sistema di controllo dell'integrità 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: basso

Guida all'implementazione

- Esegui un lavoro costante in modo che i sistemi non presentino guasti quando si verificano cambiamenti rapidi e significativi nel carico.
- Implementazione di dipendenze con accoppiamento debole Le dipendenze come sistemi di accodamento, sistemi di streaming, flussi di lavoro e bilanciatori del carico sono 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: Chiudere i circuiti e aprire le menti: come prendere il controllo dei sistemi, grandi e piccoli (include un 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:

- AmazonEC2: 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: introduzione alle architetture basate sugli eventi e ad Amazon (05) EventBridge MAD2
- AWS re:Invent 2018: Chiudere i circuiti e aprire le menti: come assumere il controllo di sistemi, grandi e piccoli (include un lavoro costante) ARC337
- AWS re:Invent 2018: Close Loops and Opening Minds: Come prendere il controllo dei sistemi, grandi e piccoli ARC337 (include accoppiamento libero, lavoro costante, stabilità statica)
- AWS re:Invent 2019: Passaggio ad architetture basate sugli eventi (08) SVS3

REL04-BP04 Rendere idempotenti le operazioni di mutazione

Un servizio idempotente promette che ogni richiesta venga elaborata esattamente una volta, in modo che effettuare più richieste identiche abbia lo stesso effetto di una singola richiesta. Questo rende più facile per un client implementare i tentativi senza temere che una richiesta venga erroneamente elaborata più volte. A tal fine, i client possono inviare richieste API con un token di idempotenza, che viene utilizzato ogni volta che la richiesta viene ripetuta. 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, anche se lo stato sottostante del sistema è cambiato.

In un sistema distribuito, è relativamente semplice eseguire un'azione al massimo una volta (il client effettua una sola richiesta) o almeno una volta (continuando a richiedere fino a quando il client non riceve una conferma di successo). È più difficile garantire che un'azione venga eseguita esattamente una volta, in modo che più richieste identiche abbiano lo stesso effetto di una singola richiesta. Utilizzando i token di idempotenza nelle API, i servizi possono ricevere una richiesta di mutazione una o più volte senza la necessità di creare record duplicati o effetti collaterali.

Risultato desiderato: hai un approccio coerente, ben documentato e ampiamente adottato per garantire l'idempotenza in tutti i componenti e servizi.

Anti-pattern comuni:

- Applichi l'idempotenza indiscriminatamente, anche quando non è necessaria.
- Introduci una logica troppo complessa per implementare l'idempotenza.
- Utilizzi i timestamp come chiavi per l'idempotenza. Questo può causare imprecisioni a causa del disallineamento dell'orologio o a causa di più client che utilizzano gli stessi timestamp per applicare le modifiche.

- Archivi interi payload per l'idempotenza. In questo approccio, salvi i payload completi dei dati per ogni richiesta e li sovrascrivi a ogni nuova richiesta. Questo può ridurre le prestazioni e influenzare la scalabilità.
- Generi le chiavi in modo incoerente tra i vari servizi. Senza chiavi coerenti, i servizi potrebbero non riconoscere le richieste duplicate, con conseguenti risultati indesiderati.

Vantaggi dell'adozione di questa best practice:

- Maggiore scalabilità: il sistema può gestire tentativi e richieste duplicate senza dover eseguire una logica aggiuntiva o una complessa gestione dello stato.
- Maggiore affidabilità: l'idempotenza aiuta i servizi a gestire più richieste identiche in modo coerente, riducendo il rischio di effetti collaterali indesiderati o di record duplicati. Questo aspetto è particolarmente importante nei sistemi distribuiti, dove gli errori di rete e i tentativi sono frequenti.
- Miglioramento della coerenza dei dati: poiché la stessa richiesta produce la stessa risposta, l'idempotenza aiuta a mantenere la coerenza dei dati nei sistemi distribuiti. Questo è essenziale per mantenere l'integrità delle transazioni e delle operazioni.
- Gestione degli errori: i token di idempotenza rendono più semplice la gestione degli errori. Se un client non riceve una risposta a causa di un problema, può tranquillamente inviare nuovamente la richiesta con lo stesso token di idempotenza.
- Trasparenza operativa: l'idempotenza consente di migliorare il monitoraggio e la registrazione. I servizi possono registrare le richieste con i propri token di idempotenza, il che facilita il monitoraggio e il debug dei problemi.
- Contratto API semplificato: può semplificare il contratto tra i sistemi lato client e lato server e ridurre il timore di un'elaborazione errata dei dati.

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

Guida all'implementazione

In un sistema distribuito, eseguire un'azione al massimo una volta (il client effettua una sola richiesta) o almeno una volta (il client continua a richiedere finché non viene confermato l'esito positivo) è relativamente semplice. Tuttavia, è difficile implementare esattamente una volta il comportamento. A tal fine, i client devono generare e fornire un token di idempotenza per ogni richiesta.

Utilizzando i token di idempotenza, un servizio può distinguere tra nuove richieste e richieste ripetute. Quando un servizio riceve una richiesta con un token di idempotenza, controlla se il token è già stato

utilizzato. Se il token è stato utilizzato, il servizio recupera e restituisce la risposta memorizzata. Se il token è nuovo, il servizio elabora la richiesta, memorizza la risposta insieme al token e restituisce la risposta. Questo meccanismo rende tutte le risposte idempotenti, migliorando l'affidabilità e la coerenza del sistema distribuito.

Anche l'idempotenza è un comportamento importante delle architetture basate su eventi. Queste architetture sono in genere supportate da una coda di messaggi come Amazon SQS, Amazon MQ, Amazon Kinesis Streams o Amazon Managed Streaming per Apache Kafka (MSK). In alcune circostanze, un messaggio pubblicato una sola volta può essere accidentalmente recapitato più di una volta. Quando un publisher genera e include i token di idempotenza nei messaggi, richiede che l'elaborazione di qualsiasi messaggio duplicato ricevuto non comporti un'azione ripetuta per lo stesso messaggio. I consumatori devono tenere traccia di ogni token ricevuto e ignorare i messaggi che contengono token duplicati.

I servizi e i consumatori devono anche passare il token di idempotenza ricevuto a tutti i servizi a valle che vengono chiamati. Ogni servizio a valle della catena di elaborazione ha la stessa responsabilità di assicurarsi che l'idempotenza sia implementata per evitare l'effetto collaterale di elaborare un messaggio più di una volta.

Passaggi dell'implementazione

1. Identifica le operazioni idempotenti

   Determina quali operazioni richiedono idempotenza. Questi includono in genere i metodi HTTP POST, PUT e DELETE e le operazioni di inserimento, aggiornamento o eliminazione del database. Le operazioni che non mutano lo stato, come le query di sola lettura, di solito non richiedono idempotenza, a meno che non abbiano effetti collaterali.

2. Utilizza identificatori univoci

   Includi un token univoco in ogni richiesta di operazione idempotente inviata dal mittente, direttamente nella richiesta o come parte dei relativi metadati (ad esempio, un'intestazione HTTP). Ciò consente al destinatario di riconoscere e gestire richieste o operazioni duplicate. Gli identificatori comunemente usati per i token includono Universally Unique Identifier (UUID) e K-Sortable Unique Identifier (KSUID).

3. Monitora e gestisci lo stato

   Mantieni lo stato di ogni operazione o richiesta nel carico di lavoro. Ciò può essere ottenuto memorizzando il token di idempotenza e lo stato corrispondente (come in attesa, completato o non

riuscito) in un database, una cache o un altro archivio persistente. Queste informazioni sullo stato consentono al carico di lavoro di identificare e gestire richieste o operazioni duplicate.

Mantieni la coerenza e l'atomicità utilizzando, se necessario, meccanismi di controllo della simultaneità appropriati, come blocchi, transazioni o controlli ottimistici della simultaneità. Questo include il processo di registrazione del token idempotente e l'esecuzione di tutte le operazioni di mutazione associate al servizio della richiesta. Questo aiuta a prevenire le condizioni di gara e verifica che le operazioni idempotenti vengano eseguite correttamente.

Rimuovi periodicamente i vecchi token di idempotenza dal datastore per gestire l'archiviazione e le prestazioni. Se il sistema di archiviazione lo supporta, prendi in considerazione l'utilizzo di timestamp di scadenza per i dati (spesso noti come valori di time-to-live, o TTL). La probabilità di riutilizzo dei token di idempotenza diminuisce nel tempo.

Le comuni opzioni di archiviazione AWS genericamente utilizzate per memorizzare i token di idempotenza e il relativo stato includono:

- Amazon DynamoDB: DynamoDB è un servizio di database NoSQL che offre prestazioni a bassa latenza ed elevata disponibilità, il che lo rende adatto all'archiviazione di dati correlati all'idempotenza. Il modello di dati chiave-valore e documento di DynamoDB consente di memorizzare e recuperare in modo efficiente i token di idempotenza e le informazioni di stato associate. DynamoDB può anche far scadere automaticamente i token di idempotenza se l'applicazione imposta un valore TTL al momento dell'inserimento.
- Amazon ElastiCache: ElastiCache è in grado di archiviare token di idempotenza con un elevato throughput, bassa latenza e a costo ridotto. ElastiCache (Redis) e ElastiCache (Memcached) possono entrambi far scadere automaticamente i token di idempotenza se l'applicazione imposta un valore TTL al momento dell'inserimento.
- Amazon Relational Database Service (RDS): puoi utilizzare Amazon RDS per archiviare i token di idempotenza e le informazioni di stato correlate, soprattutto se l'applicazione utilizza già un database relazionale per altri scopi.
- Amazon Simple Storage Service (S3): Amazon S3 è un servizio di archiviazione di oggetti altamente scalabile e durevole che può essere utilizzato per archiviare i token di idempotenza e i metadati correlati. Le funzionalità di controllo delle versioni di S3 possono essere particolarmente utili per il mantenimento dello stato di operazioni idempotenti. La scelta del servizio di archiviazione dipende in genere da fattori quali il volume dei dati correlati all'idempotenza, le caratteristiche di prestazione richieste, l'esigenza di durabilità e disponibilità e il modo in cui il meccanismo di idempotenza si integra con l'architettura complessiva del carico di lavoro.

4. Implementa operazioni idempotenti

   Progetta l'API e i componenti del carico di lavoro in modo che siano idempotenti. Incorpora i controlli di idempotenza nei componenti del carico di lavoro. Prima di elaborare una richiesta o eseguire un'operazione, controlla se l'identificatore univoco è già stato elaborato. In caso affermativo, restituisci il risultato precedente invece di eseguire nuovamente l'operazione. Ad esempio, se un client invia una richiesta per creare un utente, verifica se esiste già un utente con lo stesso identificatore univoco. Se l'utente esiste, deve restituire le informazioni utente esistenti anziché crearne una nuovo. Allo stesso modo, se un consumatore di code riceve un messaggio con un token di idempotenza duplicato, deve ignorare il messaggio.

   Crea suite di test complete che convalidino l'idempotenza delle richieste. Devono coprire un'ampia gamma di scenari, come richieste andate a buon fine, richieste non riuscite e richieste duplicate.

   Se il carico di lavoro sfrutta le funzioni AWS Lambda, prendi in considerazione Powertools per AWS Lambda. Powertools per AWS Lambda è un kit di strumenti per sviluppatori che aiuta a implementare le best practice serverless e ad aumentare la velocità di sviluppo quando si lavora con le funzioni AWS Lambda Lambda. In particolare, fornisce un'utilità per convertire le funzioni Lambda in operazioni idempotenti, sicure per riprovare.
5. Comunica chiaramente l'idempotenza

   Documenta l'API e i componenti del carico di lavoro per comunicare chiaramente la natura idempotente delle operazioni. Questo aiuta i clienti a comprendere il comportamento previsto e a capire come interagire con il carico di lavoro in modo affidabile.
6. Monitora ed esegui audit

   Implementa meccanismi di monitoraggio e audit per rilevare eventuali problemi correlati all'idempotenza delle risposte, come variazioni impreviste delle risposte o gestione eccessiva di richieste duplicate. Questo può aiutare a rilevare e indagare su eventuali problemi o comportamenti imprevisti nel carico di lavoro.

Risorse

Best practice correlate:

- REL05-BP03 Controllo e limitazione delle chiamate di ripetizione
- REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro (generazione)
- REL06-BP03 Invio di notifiche (elaborazione e avvisi in tempo reale)

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

Documenti correlati:

- The Amazon Builders' Library: Making retries safe with idempotent APIs
- The Amazon Builders' Library: difficoltà dei sistemi distribuiti
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- Amazon Elastic Container Service: Ensuring idempotency
- Come faccio a rendere idempotente la mia funzione Lambda?
- Garantire l'idempotenza nelle richieste Amazon EC2 API

Video correlati:

- Building Distributed Applications with Event-driven Architecture - AWS Online Tech Talks
- AWS re:Invent 2023 - Building next-generation applications with event-driven architecture
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- 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)

Strumenti correlati:

- Idempotenza con AWS Lambda Powertools (Java)
- Idempotenza con AWS Lambda Powertools (Python)
- Pagina GitHub AWS Lambda Powertools

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 tollerare le sollecitazioni o i 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 consentono di prevenire gli errori e migliorare il tempo medio tra guasti (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 Controlla e limita le chiamate di nuovo tentativo
- REL05-BP04 Fallisci velocemente e limita le code
- REL05-BP05 Imposta i timeout dei client
- REL05-BP06 Rendere i sistemi senza stato ove possibile
- REL05-BP07 Implementare 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 in modo degradato. 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 non va a buon fine 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: elevato

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. Per 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 strategie di 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 rivolte a un sistema downstream non vanno costantemente a buon fine, 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: come quasi tutti i database relazionali, Amazon Relational Database Service può presentare 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 aumentare orizzontalmente e anche verticalmente. 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: Throttle API Requests for Better Throughput
- CircuitBreaker (riepilogo dell'interruttore dal libro "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 (DOP328)

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

Usa le richieste di limitazione (della larghezza di banda della rete) 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 limitazioni (della larghezza di banda della rete) 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 (della larghezza di banda della rete) dei tassi di richiesta 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 allocate 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 scalabilità orizzontale non hanno configurato le impostazioni di simultaneità massima.
- La limitazione della velocità per indirizzo IP non è stata implementata.

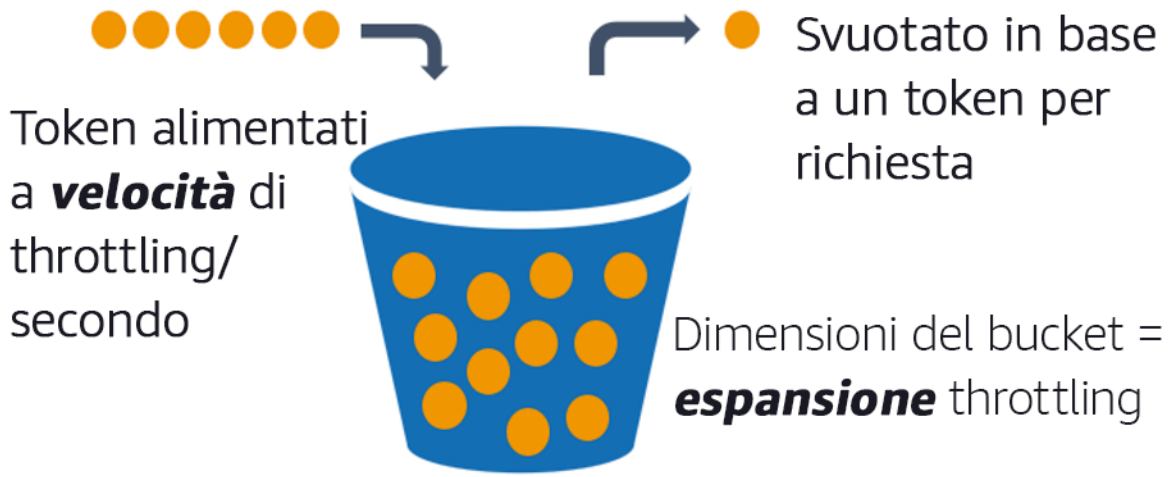
Vantaggi dell'adozione di questa best practice: i carichi di lavoro che stabiliscono limiti di accelerazione sono in grado di funzionare normalmente ed elaborare correttamente il caricamento delle richieste accettate in presenza di picchi di volume imprevisti. I picchi improvvisi o prolungati di richieste alle API e alle code vengono applicate limitazioni (della larghezza di banda della rete) e non esauriscono le risorse di elaborazione delle richieste. La limitazione (della larghezza di banda della rete) limita i singoli richiedenti in modo che gli alti volumi di traffico provenienti da un singolo indirizzo IP o da un consumatore di API non esauriscano le risorse che impattano sugli altri consumatori.

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

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 ha subito la limitazione (della larghezza di banda della rete). 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 limitazione (della larghezza di banda della rete) al secondo e svuotati in modo asincrono in base a un token per richiesta.



Algoritmo token bucket.

Gateway Amazon API 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 in buffer le richieste in modo da ridurre la frequenza delle richieste e consentire percentuali di limitazione (della larghezza di banda della rete) più elevate per le richieste che è possibile soddisfare. Infine, puoi implementare limitazioni della velocità con AWS WAF per la limitazione (della larghezza di banda della rete) di specifici utenti di API che generano un carico insolitamente elevato.

Passaggi dell'implementazione

Puoi configurare API Gateway con limiti relativi alla limitazione (della larghezza di banda della rete) per le tue API e restituire errori `429 Too Many Requests` in caso di superamento dei limiti. Puoi utilizzare AWS WAF con gli endpoint AWS AppSync e API Gateway 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 i tassi di limitazione (della larghezza di banda della rete).

Grazie all'elaborazione asincrona, dopo aver configurato Amazon SQS come origine di eventi per AWS Lambda, puoi configurare la simultaneità massima per evitare che percentuali elevate di eventi consumino la quota di esecuzione simultanea disponibile dell'account necessaria ad 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 limiti relativi alla limitazione (della larghezza di banda della rete) di API Gateway a livello di account per regione, API per fase e chiave API per livelli del piano di utilizzo.
- Applica regole di limitazione della velocità AWS WAF all'API Gateway e agli endpoint AWS AppSync per proteggerti dagli attacchi flood e bloccare gli IP dannosi. 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 ti occorre un controllo maggiore sulla limitazione (della larghezza di banda della rete) rispetto al controllo sulla limitazione della velocità per le API AWS AppSync. In tal caso, configura un'API Gateway prima dell'endpoint AWS AppSync.

- Se le code Amazon SQS sono configurate come trigger per gli utenti delle code Lambda, imposta la simultaneità massima su un valore che garantisca un'elaborazione sufficiente da soddisfare i tuoi obiettivi di livello di servizio, senza consumare 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 allocate siano equivalenti o superiori a quelle degli scenari di test prima di applicare l'aumento.

Risorse

Best practice correlate:

- REL04-BP03 Fai un lavoro costante
- REL05-BP03 Controlla e limita le chiamate di nuovo tentativo

Documenti correlati:

- Amazon API Gateway: Throttle API Requests for Better Throughput
- AWS WAF: Rate-based rule statement
- Introducing maximum concurrency of AWS Lambda when using Amazon SQS as an event source
- AWS Lambda: Maximum Concurrency

Esempi correlati:

- The three most important AWS WAF rate-based rules
- Java Bucket4j
- Algoritmo token bucket per Python
- Algoritmo token bucket a livello di nodo

- Limitazione della velocità di threading del sistema .NET

Video correlati:

- Implementing GraphQL API security best practices with AWS AppSync

Strumenti correlati:

- Gateway Amazon API
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF
- Sala d'attesa virtuale su AWS

REL05-BP03 Controlla e limita le chiamate di nuovo tentativo

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 la randomizzazione degli 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 ridurne 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 backend 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 presupponendo che i nuovi tentativi siano già integrati in scenari di ripetizione e senza test. SDK
- 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 vanno a buon fine, 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: elevato

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 la randomizzazione degli intervalli di ripetizione e limitare il numero massimo di tentativi.

Per impostazione predefinita, alcuni AWS SDKs implementano nuovi tentativi e il backoff esponenziale. Utilizza queste AWS implementazioni integrate laddove applicabile nel tuo carico di lavoro. Implementa una logica simile nel tuo carico di lavoro nelle chiamate di 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.
- Sii consapevole delle strategie esistenti SDKs che implementano strategie di ripetizione comprovate con backoff e jitter esponenziali per il linguaggio che preferisci, e preferiscile alla stesura di implementazioni personalizzate con nuovi tentativi.
- Verifica che i servizi siano caratterizzati dall'idempotenza prima dell'implementazione di nuovi tentativi. Una volta implementati i nuovi tentativi, assicurati che siano testati e che vengano regolarmente eseguiti in produzione.
- Quando chiami il AWS servizioAPIs, usa AWS SDKsand e AWS CLIcomprendi le opzioni di configurazione Retry. 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 idempotenti le operazioni di mutazione
- REL05-BP02 Richieste di limitazione (della larghezza di banda della rete)
- REL05-BP04 Fallisci velocemente e limita le code
- REL05-BP05 Imposta i timeout dei client
- REL11-BP01 Monitoraggio di tutti i componenti del carico di lavoro per la rilevazione dei guasti

Documenti correlati:

- Errore, tentativi e backoff esponenziale in AWS
- The Amazon Builders' Library: timeout, nuovi tentativi e backoff con jitter
- Exponential Backoff and Jitter
- Rendere sicuri i nuovi tentativi con idempotent APIs

Esempi correlati:

- Spring Retry
- Resilience4j Retry

Video correlati:

- Riprova, backoff e jitter: AWS re:Invent 2019: Presentazione di The Amazon Builders' Library () DOP328

Strumenti correlati:

- AWS SDKsStrumenti e strumenti: Riprova il comportamento
- AWS Command Line Interface: AWS CLI riprova

REL05-BP04 Fallisci velocemente e limita le code

Se un servizio non è in grado di rispondere correttamente a una richiesta, procede ad anticipare 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 restituisce un esito negativo 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 volte ad anticipare l'errore (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 backend. 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 configurazione delle code di lettere morte (DLQ) o degli allarmi sui DLQ volumi 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 last in first out (LIFO) risponderebbe meglio alle esigenze dei clienti, ad esempio quando non è richiesto un ordine rigoroso e l'elaborazione degli arretrati ritarda tutte le richieste nuove e urgenti, con conseguenti violazioni dei livelli di servizio per tutti i clienti.
- Esporre le code interne ai clienti anziché esporre i clienti a questi ultimi, gestire l'assunzione di lavoro e inserire le APIs richieste nelle 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 in grado di anticipare l'errore (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 in grado di anticipare l'errore (fail fast) 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: elevato

Guida all'implementazione

Le strategie volte ad anticipare l'errore (fail fast) possono essere codificate in soluzioni software e configurate nell'infrastruttura. Oltre ad anticipare l'errore (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 offre funzionalità di monitoraggio e allarme in caso di guasti. Una volta accertato il malfunzionamento di un sistema, è possibile richiamare strategie di mitigazione, ad esempio per evitare problemi dovuti a risorse danneggiate. Quando i sistemi implementano code con Amazon SQS e altre tecnologie di coda per semplificare il caricamento, devono considerare come gestire gli arretrati delle code e gli errori di consumo dei messaggi.

Passaggi dell'implementazione

- Implementa asserzioni programmatiche o metriche specifiche nel tuo software e usale per ricevere avvisi espliciti in caso di problemi di sistema. Amazon ti CloudWatch aiuta a creare metriche e allarmi basati sullo schema e SDK sulla strumentazione dei log delle applicazioni.
- Usa CloudWatch metriche e allarmi per evitare che risorse compromettano la velocità di elaborazione o che impediscono ripetutamente di elaborare le richieste.
- Utilizza l'elaborazione asincrona progettando APIs per accettare richieste e aggiungere richieste alle code interne utilizzando AmazonSQS, quindi rispondi al client che produce i messaggi con un messaggio di successo in modo che il client possa liberare risorse e passare ad altre attività mentre gli utenti della coda di backend elaborano le richieste.
- Misura e monitora la latenza di elaborazione delle code generando una CloudWatch metrica ogni volta che togli un messaggio dalla coda confrontando «now» 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'LIFOelaborazione e consente la normale elaborazione del sistema per tutti i nuovi lavori.

- 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 Implementare dipendenze liberamente accoppiate
- REL05-BP02 Richieste di limitazione (della larghezza di banda della rete)
- REL05-BP03 Controlla e limita le chiamate di nuovo tentativo
- REL06-BP02 Definizione e calcolo dei parametri (aggregazione)
- REL06-BP07 Monitora il end-to-end tracciamento delle richieste attraverso il sistema

Documenti correlati:

- Evitare insormontabili backlog di code
- Fail Fast
- Come posso evitare un crescente arretrato di messaggi nella mia SQS coda Amazon?
- Elastic Load Balancing: spostamento zonale
- Amazon Application Recovery Controller: controllo del routing per il failover del traffico

Esempi correlati:

- Modelli di integrazione aziendale: canale DLQ

Video correlati:

- AWS re:Invent 2022 - Utilizzo di applicazioni Multi-AZ ad alta disponibilità

Strumenti correlati:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 Imposta i 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: elevato

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 backend 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 limitazione (della larghezza di banda della rete) 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 gli schemi di progettazione consolidati, come i tentativi e gli interruttori automatici, per gestire i timeout in modo corretto e supportare approcci di tipo fail-fast. AWS SDKse AWS CLIconsentono la configurazione dei timeout di connessione e di richiesta e i nuovi tentativi con backoff e jitter esponenziali. AWS Lambdale funzioni supportano la configurazione dei timeout e, con AWS Step Functions, è possibile creare interruttori automatici a basso codice che sfruttano le integrazioni predefinite con i servizi e. AWS SDKs 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 tuo carico di lavoro effettua chiamate con un AWS SDK, consulta la documentazione per la configurazione del timeout specifica della lingua.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++
- Quando utilizzi AWS CLI i comandi AWS SDKs or nel tuo carico di lavoro, configura i valori di timeout predefiniti impostando i valori di AWS configurazione predefiniti per e. `connectTimeoutInMillis` `tlsNegotiationTimeoutInMillis`
- Applica le opzioni della riga di comando `cli-connect-timeout` e controlla comandi singoli `cli-read-timeout` AWS CLI ai 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 CloudWatch metriche e il rilevamento delle CloudWatch anomalie sui tassi di errore delle chiamate, sugli obiettivi dei livelli di servizio per la latenza e sui valori anomali di latenza per fornire informazioni sulla gestione di timeout eccessivamente aggressivi o permissivi.
- Configura i timeout sulle funzioni Lambda.
- APII client Gateway devono implementare i propri tentativi quando gestiscono i timeout. APIGateway supporta un timeout di integrazione da 50 millisecondi a 29 secondi per le integrazioni downstream e non riprova quando le richieste di integrazione scade.
- Implementa il modello dell'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, esamina le funzionalità di App Mesh Envoy per sfruttare timeout e interruttori integrati.
- Utilizzali AWS Step Functions per creare interruttori automatici a basso codice per chiamate di assistenza remota, in particolare quando richiami integrazioni Step Functions AWS native SDKs e supportate per semplificare il carico di lavoro.

Risorse

Best practice correlate:

- REL05-BP03 Controlla e limita le chiamate di nuovo tentativo
- REL05-BP04 Fallisci velocemente e limita le code
- REL06-BP07 Monitora il end-to-end tracciamento delle richieste attraverso il sistema

Documenti correlati:

- AWS SDK: Tentativi e timeout
- The Amazon Builders' Library: timeout, nuovi tentativi e backoff con jitter
- Quote e note importanti di Amazon API Gateway
- AWS Command Line Interface: opzioni della riga di comando
- AWS SDK for Java 2.x: Configura API i timeout
- AWS Botocore utilizzando l'oggetto config e Config Reference
- AWS SDK per .NET: nuovi tentativi e timeout
- AWS Lambda: configurazione delle opzioni della funzione Lambda

Esempi correlati:

- Utilizzo del pattern di interruttore automatico con AWS Step Functions e Amazon DynamoDB
- Martin Fowler: CircuitBreaker

Strumenti correlati:

- AWS SDKs
- AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

REL05-BP06 Rendere i sistemi senza stato ove 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 progettati per essere stateless, è possibile utilizzare servizi di elaborazione serverless, come o. AWS Lambda AWS Fargate

Oltre alla sostituzione dei server, un altro vantaggio delle applicazioni stateless è la possibilità di scalare orizzontalmente perché tutte le risorse di elaborazione disponibili (come EC2 istanze e AWS Lambda funzioni) possono soddisfare qualsiasi richiesta.

Vantaggi dell'adozione di questa best practice: i sistemi con progettazione stateless sono più adattabili alla scalabilità orizzontale, così da poter aggiungere o rimuovere capacità in base alle fluttuazioni di traffico e 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 la scalabilità 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 degli utenti dal codice dell'applicazione mediante funzionalità come pool di identità, pool di utenti e Amazon Cognito Sync.
  - Puoi separare dati degli utenti con AWS Secrets Manager, archiviando i segreti in un luogo sicuro e centralizzato. Il codice dell'applicazione pertanto non dovrà più memorizzare i segreti, rendendolo più sicuro.
  - Per archiviare dati non strutturati di grandi dimensioni, come immagini e documenti, prendi in considerazione l'utilizzo di Amazon S3. L'applicazione può recuperare questi dati quando richiesto, eliminando la necessità di archiviarli in memoria.
  - Utilizza Amazon DynamoDB per archiviare 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) e EFS Amazon MemoryDB sono esempi AWS di servizi che puoi utilizzare per scaricare i 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 Automatizza la guarigione su tutti i livelli

Documenti correlati:

- 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
- Best practice per Stateless Web Tier su AWS

REL05-BP07 Implementare 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, applicano la limitazione (della larghezza di banda della rete) 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: l'implementazione delle leve di emergenza consente di definire processi noti e validi per mantenere 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 maggiori dettagli sulla degradazione graduale, vedere REL05-BP01 Implementare graceful degradation per trasformare le dipendenze rigide applicabili in dipendenze morbide.

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 migliora la disponibilità dei componenti critici del carico di lavoro fornendo agli addetti alla risoluzione processi consolidati per rispondere a picchi imprevisti della domanda o a guasti delle dipendenze non critiche.

Livello di rischio associato se questa best practice non fosse adottata: 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 di Amazon, consulta Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second.
  - 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 -BP05 Utilizzare la stabilità statica per prevenire comportamenti bimodali. REL11
- 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) o l'implementazione della normale riduzione delle prestazioni.

Risorse

Best practice correlate:

- REL05-BP01 Implementa una degradazione graduale per trasformare le dipendenze rigide applicabili in dipendenze morbide
- REL05-BP02 Richieste Throttle
- REL11-BP05 Usa la stabilità statica per prevenire comportamenti bimodali

Documenti correlati:

- Automatizzazione di distribuzioni pratiche e sicure
- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second

Video correlati:

- AWS re:Invent 2020: affidabilità, coerenza e fiducia grazie all'immutabilità

Gestione delle modifiche

Le modifiche apportate al carico di lavoro o al relativo ambiente devono essere previste e gestite in modo da garantire l'affidabilità del carico di lavoro. Le modifiche includono quelle imposte al carico di lavoro, ad esempio i picchi di domanda, nonché quelle dall'interno quali le implementazioni delle funzionalità e delle patch di sicurezza.

Le sezioni seguenti illustrano le best practice per la gestione delle modifiche.

Argomenti

Monitoraggio delle risorse del carico di lavoro

I log e le metriche sono strumenti molto efficaci per ottenere informazioni sullo stato del carico di lavoro. Puoi configurare il carico di lavoro in modo da monitorare i log e le metriche e inviare notifiche in caso di superamento delle soglie o di eventi significativi. Il monitoraggio permette al carico di lavoro di riconoscere il superamento delle soglie di prestazioni basse o il verificarsi di errori, in modo da ripristinarlo in automatico di conseguenza.

Il monitoraggio è essenziale per accertarti di soddisfare i requisiti di disponibilità. Il monitoraggio deve rilevare in modo efficace 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 ciascuna richiesta, per tutte le dipendenze e per le operazioni chiave all'interno del processo. Registriamo anche metriche di operazioni di successo. In questo modo vediamo 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 sarà comunque insoddisfacente. 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 Analizza i registri
- REL06-BP06 Revisione periodica dell'ambito e delle metriche di monitoraggio
- REL06-BP07 Monitora il end-to-end tracciamento 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 log, se necessario, e imposta soglie per richiamare gli 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 log 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 machine learning 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 delle variazioni di stagionalità e di tendenza dei parametri.

AWS mette a disposizione una grande quantità di informazioni di monitoraggio e di log 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 machine learning 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 svolto attraverso transazioni sintetiche (talvolta definite canary dell'utente, ma da non confondere con le distribuzioni canary) che eseguono periodicamente alcune attività comuni che corrispondono a quelle effettuate 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 consente di creare canary sintetici per monitorare endpoint e API. Puoi anche combinare i nodi client sintetici canary con la console AWS X-Ray per individuare quali canary sintetici stanno riscontrando problemi con errori, guasti o tassi di limitazione (della larghezza di banda della rete) 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 dell'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: elevato

Guida all'implementazione

1. Attiva la creazione di log, laddove possibile. I dati di monitoraggio devono essere ottenuti da tutti i componenti dei carichi di lavoro. Attiva ulteriori log, come i log di accesso S3, e consenti al carico di lavoro di creare log per i dati specifici del carico di lavoro. Raccogli i parametri relativi a 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, e Amazon EMR. Consulta AWS Services That Publish CloudWatch Metrics per un elenco di 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 i tuoi parametri in CloudWatch tramite la AWS CLI o un'API.
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. Concentrarsi su parametri e soglie critiche consente di affinare il numero di avvisi, così da ridurre al minimo i falsi positivi.
4. Definisci gli avvisi e il processo di recupero del carico di lavoro dopo il richiamo 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 del tempo di ripristino (RTO). Puoi usare gli allarmi di Amazon CloudWatch per richiamare 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. Grazie alle transazioni sintetiche, puoi scoprire i problemi prima che vengano rilevati dai 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
- AWS Services That Publish CloudWatch Metrics
- Access Logs for Your Network Load Balancer
- Access logs for your application load balancer
- Accesso ad Amazon CloudWatch Logs per AWS Lambda
- Registrazione degli accessi al server Amazon S3
- Enable Access Logs for Your Classic Load Balancer
- Exporting log data to Amazon S3
- Install the CloudWatch agent on an Amazon EC2 instance
- Publishing Custom Metrics
- Using Amazon CloudWatch Dashboards
- Using Amazon CloudWatch Metrics
- Utilizzo di Canary (Amazon CloudWatch Synthetics)
- What are Amazon CloudWatch Logs?

Guide per l'utente:

- Creating a trail
- Monitoraggio dei parametri relativi a memoria e disco per le istanze Linux di Amazon EC2
- Utilizzo dei CloudWatch Logs con le istanze di container
- Log di flusso VPC
- What is Amazon DevOps Guru?
- Che cos'è AWS X-Ray?

Blog correlati:

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray

Esempi correlati:

- The Amazon Builders' Library: strumentazione di sistemi distribuiti per visibilità operativa
- Workshop sull'osservabilità

REL06-BP02 Definizione e calcolo dei parametri (aggregazione)

Raccogli metriche e log dai componenti del carico di lavoro e calcola le metriche aggregate pertinenti. Queste metriche forniscono un'ampia e profonda osservabilità del carico di lavoro e possono migliorare in modo significativo lo stato di resilienza.

L'osservabilità non si limita alla semplice raccolta di metriche dai componenti del carico di lavoro e alla possibilità di visualizzarle e di emettere avvisi. Si tratta di avere una comprensione olistica del comportamento del carico di lavoro. Queste informazioni comportamentali provengono da tutti i componenti dei carichi di lavoro, compresi i servizi cloud da cui dipendono, i log ben realizzati e le metriche. Questi dati consentono di controllare il comportamento del carico di lavoro nel suo complesso e di comprendere l'interazione di ogni componente con ogni unità di lavoro a un livello di dettaglio molto elevato.

Risultato desiderato:

- Raccogli i log dai componenti del carico di lavoro e dalle dipendenze dei servizi AWS e li pubblichi in una posizione centrale dove possono essere facilmente consultati ed elaborati.
- I log contengono timestamp accurati e ad alta fedeltà.
- I log contengono informazioni rilevanti sul contesto di elaborazione, come l'identificativo della traccia, l'identificativo dell'utente o dell'account e l'indirizzo IP remoto.
- Dai log crei metriche aggregate che rappresentano il comportamento del carico di lavoro da una prospettiva di alto livello.
- Puoi eseguire query sui log aggregati per ottenere informazioni approfondite e pertinenti sul carico di lavoro e identificare problemi effettivi e potenziali.

Anti-pattern comuni:

- Non raccogli log o metriche pertinenti dalle istanze di calcolo su cui vengono eseguiti i carichi di lavoro o dai servizi cloud che utilizzano.
- Trascuri la raccolta di log e metriche collegate agli indicatori chiave delle prestazioni (KPI) aziendali.

- Analizzi la telemetria correlata al carico di lavoro in modo isolato, senza aggregazione e correlazione.
- Consenti che le metriche e i log scadano troppo rapidamente, il che ostacola l'analisi delle tendenze e l'identificazione dei problemi ricorrenti.

Vantaggi dell'adozione delle best practice: puoi rilevare un maggior numero di anomalie e correlare eventi e metriche tra i diversi componenti del carico di lavoro. Puoi creare approfondimenti sui componenti del carico di lavoro in base alle informazioni contenute nei log che spesso non sono disponibili nelle sole metriche. Puoi determinare più rapidamente le cause degli errori eseguendo query sui log su larga scala.

Livello di rischio associato se queste best practice non fossero adottate: elevato

Guida all'implementazione

Identifica le origini di dati di telemetria rilevanti per i carichi di lavoro e i relativi componenti. Questi dati provengono non solo dai componenti che pubblicano metriche, come il sistema operativo (OS) e i runtime delle applicazioni come Java, ma anche dai log delle applicazioni e dei servizi cloud. Ad esempio, i server web in genere registrano ogni richiesta con informazioni dettagliate come il timestamp, la latenza di elaborazione, l'ID utente, l'indirizzo IP remoto, il percorso e la stringa di query. Il livello di dettaglio di questi log consente di eseguire query dettagliate e di generare metriche che altrimenti non sarebbero disponibili.

Raccogli le metriche e i log utilizzando strumenti e processi appropriati. I log generati dalle applicazioni in esecuzione su un'istanza Amazon EC2 possono essere raccolti da un agente come Agente Amazon CloudWatch e pubblicati su un servizio di archiviazione centrale come Amazon CloudWatch Logs. I servizi di elaborazione gestiti da AWS come AWS Lambda e Amazon Elastic Container Service pubblicano automaticamente i log su CloudWatch Logs. Abilita la raccolta di log per i servizi di archiviazione ed elaborazione AWS utilizzati dai carichi di lavoro come Amazon CloudFront, Amazon S3, Elastic Load Balancing e Gateway Amazon API.

Arricchisci i dati di telemetria con dimensioni che possono aiutarti a vedere più chiaramente i modelli di comportamento e a isolare i problemi relativi a gruppi di componenti correlati. Una volta aggiunte, è possibile osservare il comportamento dei componenti a un livello di dettaglio più fine, rilevare gli errori correlati e adottare le operazioni correttive appropriate. Esempi di dimensioni utili includono zona di disponibilità, ID istanza EC2 e attività del container o Pod ID.

Dopo aver raccolto le metriche e i log, puoi scrivere query e generare metriche aggregate che forniscono informazioni utili sul comportamento normale e anomalo. Ad esempio, puoi utilizzare

Approfondimenti di Amazon CloudWatch Logs per ricavare parametri personalizzati dai log delle applicazioni, approfondimenti sulle metriche Amazon CloudWatch per eseguire query sui parametri su larga scala, approfondimenti sui container Amazon CloudWatch per raccogliere, aggregare e riepilogare metriche e log dalle applicazioni e microservizi containerizzati o Lambda Insights di Amazon CloudWatch se utilizzi le funzioni AWS Lambda. Per creare una metrica aggregata del tasso di errore, è possibile incrementare un contatore ogni volta che si trova una risposta o un messaggio di errore nei log del componente o calcolare il valore aggregato di una metrica del tasso di errore esistente. Puoi utilizzare questi dati per generare istogrammi che mostrano il comportamento della coda, come le richieste o i processi con le prestazioni peggiori. Puoi anche eseguire la scansione di questi dati in tempo reale alla ricerca di modelli anomali utilizzando soluzioni come il rilevamento delle anomalie di CloudWatch Logs. Queste informazioni approfondite possono essere inserite in dashboard per essere organizzate in base alle esigenze e alle preferenze.

L'esecuzione di query sui log può aiutare a comprendere come sono state gestite richieste specifiche dai componenti del carico di lavoro e a rivelare modelli di richiesta o altri contesti che hanno un impatto sulla resilienza del carico di lavoro. Può essere utile ricercare e preparare le query in anticipo, in base alla conoscenza del comportamento delle applicazioni e degli altri componenti, in modo da poterle eseguire più facilmente quando necessario. Ad esempio, con Approfondimenti di Amazon CloudWatch Logs, puoi cercare e analizzare in modo interattivo i dati di log memorizzati in CloudWatch Logs. Puoi anche usare Amazon Athena per eseguire query sui log da più origini, inclusi molti servizi AWS, su una scala di petabyte.

Quando si definisce una policy di conservazione dei log, devi considerare il valore dei log storici. I log storici possono aiutare a identificare modelli di utilizzo e di comportamento a lungo termine, regressioni e miglioramenti delle prestazioni del carico di lavoro. I log eliminati definitivamente non possono essere analizzati in seguito. Tuttavia, il valore dei log storici tende a diminuire nel corso di lunghi periodi di tempo. Scegli una policy che sia in grado di bilanciare le esigenze e che sia conforme ai requisiti legali o contrattuali a cui potresti essere soggetto.

Passaggi dell'implementazione

1. Scegli i meccanismi di raccolta, archiviazione, analisi e visualizzazione dei dati di osservabilità.
2. Installa e configura i raccoglitori di metriche e di log sui componenti appropriati del carico di lavoro (ad esempio, sulle istanze Amazon EC2 e nei container sidecar). Configura questi raccoglitori in modo che si riavviino automaticamente nel caso in cui si arrestino in modo imprevisto. Abilita il buffering su disco o in memoria per i collettori, in modo che le interruzioni temporanee della pubblicazione non abbiano ripercussioni sulle applicazioni né comportino la perdita di dati.

3. Abilita la registrazione sui servizi AWS utilizzati come parte dei carichi di lavoro e inoltra i log al servizio di archiviazione selezionato, se necessario. Per istruzioni dettagliate, consulta le guide per l'utente o gli sviluppatori dei rispettivi servizi.
4. Definisci le metriche operative rilevanti per i carichi di lavoro, basate sui dati di telemetria. Questi possono essere basati su metriche dirette emesse dai componenti del carico di lavoro, che possono includere metriche correlate a KPI aziendali, o sui risultati di calcoli aggregati come somme, tassi, percentili o istogrammi. Calcola queste metriche utilizzando l'analizzatore log e inseriscile nelle dashboard come opportuno.
5. Prepara query di log appropriate per analizzare i componenti del carico di lavoro, le richieste o il comportamento delle transazioni, se necessario.
6. Definisci e abilita una policy di conservazione dei log per i log dei componenti. Elimina periodicamente i log quando diventano più vecchi di quanto consentito dalla policy.

Risorse

Best practice correlate:

- REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro (generazione)
- 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 dei log
- REL06-BP06 Revisione periodica dell'ambito e delle metriche di monitoraggio
- REL06-BP07 Monitoraggio del tracciamento end-to-end delle richieste attraverso il sistema

Documentazione correlata:

- How Amazon CloudWatch works
- Amazon Managed Prometheus
- Grafana gestito da Amazon
- Analyzing log data with CloudWatch Logs Insights
- Lambda Insights di Amazon CloudWatch
- Approfondimenti sui container Amazon CloudWatch
- Query your metrics with CloudWatch Metrics Insights

- AWS Distro per OpenTelemetry
- Query di esempio di Approfondimenti di Amazon CloudWatch Logs
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- Ricerca e filtraggio dei dati di log
- Invio di log direttamente ad Amazon S3
- The Amazon Builders' Library: strumentazione di sistemi distribuiti per visibilità operativa

Workshop correlati:

- One Observability Workshop

Strumenti correlati:

- AWS Distro for OpenTelemetry (GitHub)

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: elevato

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 consente di creare allarmi di parametri e compositi mediante gli allarmi CloudWatch basati su soglie statiche, rilevamento di anomalie e altri criteri. Per ulteriori informazioni sui tipi di allarmi configurabili mediante CloudWatch, consulta la sezione sugli allarmi della documentazione di CloudWatch.

Puoi creare per i tuoi team visualizzazioni personalizzate dei parametri e degli avvisi delle risorse AWS utilizzando i pannelli di controllo di 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 Amazon EC2 o un'azione Amazon EC2 Auto Scaling oppure creando un OpsItem o un incidente 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 ulteriori informazioni sulla configurazione delle notifiche di Amazon SNS, consulta Configuring Amazon SNS.

CloudWatch invia eventi EventBridge 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 di Systems Manager.

Con AWS Health si rimane sempre aggiornati. AWS Health è la fonte autorevole di informazioni sull'integrità delle risorse Cloud AWS. AWS Health consente di ricevere le notifiche in caso di eventi del servizio confermati, in modo da poter adottare rapidamente le misure necessarie per mitigare qualsiasi impatto. Si creano notifiche di eventi AWS Health personalizzati per i canali e-mail e

chat con Notifiche all'utente AWS e si usano gli strumenti di monitoraggio e avviso con Amazon EventBridge per l'integrazione a livello di codice. Se si utilizza AWS Organizations, è possibile aggregare gli eventi AWS Health tra gli account.

Quando utilizzare EventBridge o 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 per creare applicazioni che reagiscano agli eventi delle applicazioni, delle applicazioni SaaS e dei servizi AWS. EventBridge è l'unico servizio basato su eventi integrato direttamente con partner SaaS di terze parti. EventBridge inoltre acquisisce automaticamente eventi da oltre 200 servizi AWS senza che gli sviluppatori debbano 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 supporta al momento oltre 20 servizi AWS come destinazione, tra cui AWS Lambda, Amazon SQS, Amazon SNS, flusso di dati Amazon Kinesis e 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 assumere qualsiasi formato. Amazon SNS consente di inoltrare 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 mediante gli allarmi di 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, nell'esecuzione di un'azione Amazon EC2 o un'azione Amazon EC2 Auto Scaling oppure nella creazione di un OpsItem o di un incidente 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 di Amazon SNS quando cambiano stato e possono creare oggetti OpsItem di Systems Manager o incidenti quando passano allo stato di allarme, ma non possono eseguire azioni EC2 o azioni Auto Scaling.

2. Configura le notifiche di 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 corrispondenti 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, il che attiva una funzione Lambda tramite una regola EventBridge per ripulire lo spazio. Per ulteriori informazioni sulle destinazioni EventBridge, consulta EventBridge targets.

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
- Using Amazon CloudWatch alarms
- Using Amazon CloudWatch dashboards
- Uso di parametri di Amazon CloudWatch
- Impostazione delle notifiche Amazon SNS
- CloudWatch anomaly detection
- Protezione dei dati di CloudWatch Logs
- Amazon EventBridge

- Amazon Simple Notification Service

Video correlati:

- Video sull'osservabilità 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: identificazione degli allarmi in tempo reale e impostazione dell'elaborazione automatica degli allarmi per richiamare le azioni appropriate intraprese per rispettare 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: migliore resilienza del sistema grazie all'automazione dell'elaborazione degli allarmi. 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 se questa best practice non fosse adottata: 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, utilizza AWS CLI mediante il comando `describe-alarms` di Amazon CloudWatch. In base al numero di allarmi impostati, potrebbe essere necessario utilizzare la paginazione per recuperare un sottoinsieme di allarmi per ciascuna chiamata o, in alternativa, è possibile utilizzare l'SDK AWS per recuperare gli allarmi utilizzando una chiamata API.
2. Documenta tutte le azioni associate all'allarme: aggiorna un runbook con tutti gli allarmi e le relative azioni, a prescindere che siano manuali o automatizzati. AWS Systems Manager offre runbook predefiniti. Per informazioni sull'uso dei runbook, consulta Working with runbooks. Per informazioni sulla visualizzazione dei contenuti dei runbook, consulta Visualizza il contenuto del runbook.

3. Configura e gestisci le azioni associate all'allarme: per tutti gli allarmi che richiedono un'azione, specifica l'azione automatizzata mediante l'SDK CloudWatch. 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 utilizzare Amazon EventBridge per rispondere automaticamente agli eventi di sistema, come i problemi relativi alla disponibilità delle applicazioni o le modifiche delle risorse. Puoi creare regole che indichino a quali eventi sei interessato e quali operazioni automatizzate eseguire quando un evento corrisponde a una regola. Le azioni avviabili in automatico includono il richiamare una funzione AWS Lambda, il richiamare `Run Command` di Amazon EC2, l'inoltro dell'evento al flusso di dati Amazon Kinesis e la visualizzazione del comando Automate di Amazon EC2 mediante EventBridge.

4. Procedure operative standard (SOP): in base ai componenti dell'applicazione, AWS Resilience Hub suggerisce 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. È altresì possibile creare una SOP in base alle raccomandazioni di Resilience Hub, laddove sia necessaria un'applicazione Resilience Hub con una policy di resilienza associata, nonché valutare a livello cronologico la resilienza rispetto a tale applicazione. Le raccomandazioni per la SOP sono prodotte dalla valutazione della resilienza.

   Resilience Hub lavora in sinergia con Systems Manager per automatizzare le fasi delle SOP, fornendo una serie di documenti SSM utilizzabili 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 usare Amazon DevOps Guru per monitorare automaticamente le risorse dell'applicazione al fine di rilevare comportamenti anomali e fornire raccomandazioni mirate per accelerare i tempi di identificazione e riparazione dei problemi. DevOps Guru consente di monitorare flussi di dati operativi quasi in tempo reale da più origini, tra cui i parametri di Amazon CloudWatch, AWS Config, AWS CloudFormation e AWS X-Ray. Inoltre, puoi usare DevOps Guru per creare in automatico OpsItems in OpsCenter e inviare eventi a EventBridge per un ulteriore livello di automazione.

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
- Creazione di una regola EventBridge che attivi un evento da una risorsa AWS
- One Observability Workshop
- The Amazon Builders' Library: strumentazione di sistemi distribuiti per visibilità operativa
- What is Amazon DevOps Guru?
- Utilizzo dei documenti di automazione (playbook)

Video correlati:

- AWS re:Invent 2022 - Observability best practices at 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 su Amazon CloudWatch e Systems Manager

REL06-BP05 Analizza i registri

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 abbonamenti che consentono ai dati di fluire senza problemi verso Amazon S3, dove puoi utilizzare o Amazon Athena per interrogare i dati. Supporta, inoltre, le query su un'ampia gamma di formati. Per ulteriori

informazioni, consulta la sezione Formati supportati SerDes e di dati nella Guida per l'utente di Amazon Athena. Per l'analisi di enormi set di file di log, puoi eseguire un EMR cluster Amazon per eseguire analisi su scala petabyte.

Esistono numerosi strumenti forniti dai AWS partner e da terze parti che consentono l'aggregazione, l'elaborazione, l'archiviazione e l'analisi. Questi strumenti includono New Relic, Splunk, Loggly, Logstash e Nagios. CloudHealth 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 ad Amazon S3 Glacier per lo storage a lungo termine e in seguito la 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: medio

Guida all'implementazione

- CloudWatch Logs Insights ti consente di cercare e analizzare in modo interattivo i dati di log in Amazon CloudWatch Logs.
  - Analisi dei dati di log con Logs Insights CloudWatch
  - Domande di esempio su Amazon CloudWatch Logs Insights
- Usa Amazon CloudWatch Logs per inviare i log ad Amazon S3 dove puoi usare o Amazon Athena per interrogare i dati.
  - Come posso usare Amazon Athena per analizzare i log di accesso al server Amazon S3?
    - 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 analizzati da Athena per ogni query.
      - Come posso creare una policy del ciclo di vita per un bucket S3?

Risorse

Documenti correlati:

- Domande di esempio su Amazon CloudWatch Logs Insights
- Analisi dei dati di log con Logs Insights CloudWatch
- Esecuzione del debug con Amazon Synthetics e CloudWatch AWS X-Ray
- Come posso creare una policy del ciclo di vita per un bucket S3?
- Come posso usare Amazon Athena per analizzare i log di accesso al server Amazon S3?
- One Observability Workshop
- The Amazon Builders' Library: strumentazione di sistemi distribuiti per visibilità operativa

REL06-BP06 Revisione periodica dell'ambito e delle metriche di monitoraggio

Esegui revisioni frequenti della modalità di implementazione del monitoraggio del carico di lavoro e aggiornarla in base all'evoluzione del carico di lavoro e della relativa architettura. Gli audit regolari del monitoraggio aiutano a ridurre il rischio di indicatori di problemi mancati o trascurati e aiutano ulteriormente il carico di lavoro a raggiungere gli obiettivi di disponibilità.

Un monitoraggio efficace si basa su metriche aziendali chiave, che si evolvono in base al cambiamento delle priorità aziendali. Il processo di revisione del monitoraggio deve porre l'accento sugli indicatori del livello di servizio (SLI) e incorporare le informazioni approfondite provenienti dall'infrastruttura, dalle applicazioni, dai client e dagli utenti.

Risultato desiderato: disponi di una strategia di monitoraggio efficace che viene regolarmente rivista e aggiornata periodicamente, oltre che dopo qualsiasi evento o cambiamento significativo. Verifichi che gli indicatori chiave di integrità dell'applicazione siano ancora pertinenti con l'evoluzione del carico di lavoro e dei requisiti aziendali.

Anti-pattern comuni:

- Raccogli solo metriche predefinite.
- Hai impostato una strategia di monitoraggio, ma non esegui mai alcuna revisione.
- Non discuti il monitoraggio quando vengono distribuite modifiche importanti.
- Fai affidamento a metriche obsolete per determinare l'integrità del carico di lavoro.
- I team operativi sono sommersi da avvisi falsi positivi dovuti a metriche e soglie obsolete.
- Manca l'osservabilità dei componenti dell'applicazione che non vengono monitorati.

- Ti concentri solo su metriche tecniche di basso livello ed escludi le metriche aziendali dal monitoraggio.

Vantaggi dell'adozione di questa best practice: una revisione regolare del monitoraggio consente di anticipare i potenziali problemi e di verificare la capacità di rilevarli. Inoltre, consente di scoprire i punti ciechi che potrebbero essere sfuggiti durante le revisioni precedenti, migliorando ulteriormente la capacità di individuare i problemi.

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

Guida all'implementazione

Rivedi le metriche di monitoraggio e l'ambito durante il processo di revisione della prontezza operativa (ORR). Esegui periodicamente revisioni della prontezza operativa per valutare se ci sono lacune tra il carico di lavoro attuale e il monitoraggio configurato. Stabilisci una cadenza regolare per le revisioni delle prestazioni operative e la condivisione delle conoscenze per migliorare la capacità di ottenere prestazioni più elevate dai team operativi. Convalida se le soglie di allarme esistenti sono ancora adeguate e verifica le situazioni in cui i team operativi ricevono avvisi falsi positivi o non monitorano aspetti dell'applicazione che invece devono esserlo.

Il framework di analisi della resilienza fornisce indicazioni utili che possono aiutare a esplorare il processo. L'obiettivo del framework è identificare le potenziali modalità di errore e i controlli preventivi e correttivi da utilizzare per mitigare l'impatto. Queste conoscenze possono aiutare a identificare le metriche e gli eventi giusti da monitorare e segnalare.

Passaggi dell'implementazione

1. Pianifica ed effettua revisioni periodiche dei pannelli di controllo del carico di lavoro. La frequenza può essere diversa a seconda di quanto l'ispezione sia approfondita.
2. 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. Esempi di questo tipo sono l'aumento della latenza, la riduzione della funzione aziendale primaria e l'aumento delle risposte agli errori.
3. Esamina i valori anomali e le anomalie nelle metriche, che possono essere mascherate dalle medie o dalle mediane. Osserva i valori più alti e più bassi durante l'arco temporale e indaga sulle cause di osservazioni che sono molto al di fuori dei limiti normali. Continuando a eliminare queste cause, puoi restringere i limiti delle metriche previste in risposta al miglioramento della coerenza delle prestazioni del carico di lavoro.

4. Ricerca di bruschi cambiamenti nel comportamento. Un cambiamento immediato nella quantità o nella direzione di una metrica può indicare che si è verificato un cambiamento nell'applicazione o nei fattori esterni che potrebbe richiedere l'aggiunta di ulteriori metriche da monitorare.
5. Verifica se l'attuale strategia di monitoraggio rimane pertinente per l'applicazione. Sulla base di un'analisi degli incidenti precedenti (o del framework di analisi della resilienza), valuta se ci sono ulteriori aspetti dell'applicazione che dovrebbero essere incorporati nell'ambito del monitoraggio.
6. Esamina le metriche di monitoraggio dell'utente reale (RUM, Real User Monitoring) per determinare se ci sono lacune nella copertura delle funzionalità dell'applicazione.
7. Rivedi il processo di gestione del cambiamento. Se necessario, aggiorna le procedure per includere una fase di analisi di monitoraggio da eseguire prima di approvare una modifica.
8. Implementa la revisione del monitoraggio come parte della revisione della prontezza operativa e dei processi di correzione degli errori.

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-BP07 Monitoraggio del tracciamento end-to-end delle richieste attraverso il sistema
- REL12-BP02 Esecuzione di analisi post-incidente
- REL12-BP06 Esecuzione regolare di GameDay

Documenti correlati:

- Why you should develop a correction of error (COE)
- Using Amazon CloudWatch Dashboards
- Building dashboards for operational visibility
- Advanced Multi-AZ Resilience Patterns - Gray failures
- Query di esempio di Approfondimenti di Amazon CloudWatch Logs
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- One Observability Workshop
- The Amazon Builders' Library: strumentazione di sistemi distribuiti per visibilità operativa

- Using Amazon CloudWatch Dashboards
- AWS Observability Best Practices
- Resilience Analysis Framework
- Resilience Analysis Framework - Observability
- Operational Readiness Review - ORR

REL06-BP07 Monitora il end-to-end tracciamento 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 tracciamento completo su tutti i componenti sono facili da eseguire il debug, migliorando il tempo medio di risoluzione (MTTR) degli errori e la latenza semplificando l'individuazione della causa principale. End-to-endil tracciamento riduce il tempo necessario per scoprire i componenti interessati e approfondire in dettaglio le cause profonde degli errori o della latenza.

Anti-pattern comuni:

- Il tracciamento viene utilizzato per alcuni componenti ma non per tutti. Ad esempio, senza tracciamento, i team potrebbero non comprendere chiaramente la AWS Lambda latenza causata dagli avviamenti a freddo in un carico di lavoro con picchi di lavoro.
- I canarini sintetici o il monitoraggio degli utenti reali () RUM non sono configurati con il tracciamento. Senza canaries orRUM, la telemetria relativa all'interazione con il cliente viene omessa dall'analisi delle tracce, producendo un profilo prestazionale incompleto.
- I carichi di lavoro ibridi includono strumenti di tracciamento cloud-native (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, è necessario utilizzare il tracciamento nativo del cloud per strumentare componenti che non sono nativi del cloud oppure gli strumenti di terze parti SDKs devono essere configurati per assimilare la telemetria di tracciamento nativa 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 creazione di log, 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 richiamare 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 (software intermediario) e router di eventi, componenti di elaborazione e archiviazione, tra cui gli archivi e i database dei valori chiave. Includi canarini sintetici e il monitoraggio di utenti reali nella configurazione di end-to-end tracciamento per misurare le interazioni e la latenza dei clienti remoti in modo da poter valutare con precisione le prestazioni dei sistemi rispetto agli accordi e agli obiettivi sui livelli di servizio.

Puoi utilizzare AWS X-Rayi servizi di strumentazione Amazon CloudWatch Application Monitoring per fornire una visione completa delle richieste mentre viaggiano attraverso la tua applicazione. X-Ray raccoglie la telemetria delle applicazioni e consente di visualizzarla e filtrarla tra payload, funzioni, tracceAPIs, servizi e può essere attivata per componenti di sistema senza codice o low-code. CloudWatch il monitoraggio delle applicazioni include l'integrazione delle tracce con ServiceLens metriche, registri e allarmi. CloudWatch il monitoraggio delle applicazioni include anche strumenti sintetici per monitorare gli endpoint eAPIs, oltre al monitoraggio degli utenti reali, per strumentare i client delle applicazioni Web.

Passaggi dell'implementazione

- AWS X-Ray Utilizzalo su tutti i servizi nativi supportati come Amazon S3 e Amazon AWS Lambda API Gateway. Questi AWS servizi abilitano X-Ray con toggle di configurazione utilizzando l'infrastruttura come codice, AWS SDKs oppure. AWS Management Console
- Dota di strumenti le applicazioni AWS Distro for Open Telemetry e X-Ray o gli agenti di raccolta di terze parti.

- Consulta la Guida per gli sviluppatori AWS X-Ray per l'implementazione specifica del linguaggio di programmazione. Queste sezioni della documentazione descrivono in dettaglio come strumentare HTTP richieste, SQL interrogazioni e altri processi specifici del linguaggio di programmazione delle applicazioni.
- Usa il tracciamento X-Ray per Amazon CloudWatch Synthetic Canaries e CloudWatch RUMAmazon per analizzare il percorso della richiesta dal tuo client utente finale all'infrastruttura downstream. AWS
- Configura CloudWatch metriche e allarmi in base allo stato delle risorse e alla telemetria di Canary in modo che i team vengano avvisati rapidamente dei problemi e possano quindi approfondire 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 Relic o Dynatrace in caso di utilizzo di 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
- Esecuzione del debug con Amazon Synthetics e CloudWatch AWS X-Ray
- The Amazon Builders' Library: strumentazione di sistemi distribuiti per visibilità operativa
- Integrazione AWS X-RayAWS con altri servizi
- AWS Distro per e OpenTelemetry AWS X-Ray
- Amazon CloudWatch: utilizzo del monitoraggio sintetico
- Amazon CloudWatch: utilizzo CloudWatch RUM
- Configura Amazon CloudWatch synthetics canary e Amazon Alarm CloudWatch
- Disponibilità e oltre: comprensione e miglioramento della resilienza dei sistemi distribuiti su AWS

Esempi correlati:

- One Observability Workshop

Video correlati:

- AWS re:Invent 2022 - 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 garantisce l'elasticità per aggiungere o rimuovere risorse in automatico, in modo che sussista 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 Ottenere risorse dopo aver rilevato una compromissione del carico di lavoro
- REL07-BP03 Ottenimento di risorse dopo aver rilevato che sono necessarie più risorse per un carico di lavoro
- REL07-BP04 Load Testa il tuo carico di lavoro

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

Una pietra miliare dell'affidabilità nel cloud è la definizione, il provisioning e la gestione programmatica dell'infrastruttura e delle risorse. L'automazione aiuta a semplificare il provisioning delle risorse, facilitare implementazioni coerenti e sicure e scalare le risorse sull'intera infrastruttura.

Risultato desiderato: gestisci infrastructure as code (IaC). Il codice dell'infrastruttura viene definito e gestito nei sistemi di controllo delle versioni (VCS). Deleghi il provisioning delle risorse AWS a meccanismi automatici e sfrutti i servizi gestiti come Application Load Balancer (ALB), Network Load Balancer (NLB) e i gruppi Auto Scaling. Il provisioning delle risorse si avvale di pipeline di integrazione continua/distribuzione continua (CI/CD) in modo che le modifiche al codice avviino automaticamente gli aggiornamenti delle risorse, inclusi gli aggiornamenti delle configurazioni di dimensionamento automatico.

Anti-pattern comuni:

- Distribuisci le risorse manualmente utilizzando la riga di comando o la AWS Management Console (conosciuto anche come ClickOps).
- Accoppi strettamente i componenti o le risorse dell'applicazione, creando di conseguenza architetture poco flessibili.
- Implementi policy di dimensionamento rigide che non si adattano ai requisiti aziendali in evoluzione, ai modelli di traffico o ai nuovi tipi di risorse.
- Esegui la stima manuale della capacità di soddisfare la domanda prevista.

Vantaggi della definizione di questa best practice: l'infrastructure as code (IaC) consente di definire l'infrastruttura a livello di programmazione. Questo aiuta a gestire le modifiche all'infrastruttura attraverso lo stesso ciclo di vita dello sviluppo software delle modifiche all'applicazione, favorendo la coerenza e la ripetibilità e riducendo il rischio di attività manuali soggette a errori. Il processo di provisioning e aggiornamento delle risorse può essere semplificato ulteriormente implementando IaC con pipeline di distribuzione automatiche. È possibile implementare gli aggiornamenti dell'infrastruttura in modo affidabile ed efficiente, senza bisogno di interventi manuali. Questa agilità è particolarmente importante quando si tratta di scalare le risorse per soddisfare richieste fluttuanti.

È possibile ottenere una scalabilità dinamica e automatizzata delle risorse in combinazione con IaC e pipeline di distribuzione. Monitorando le metriche chiave e applicando policy di dimensionamento predefinite, il dimensionamento automatico è in grado di effettuare automaticamente il provisioning o il deprovisioning delle risorse secondo necessità, migliorando le prestazioni e l'efficienza in termini di costi. In questo modo si riduce il potenziale di errori manuali o di ritardi in risposta alle modifiche dei requisiti delle applicazioni o del carico di lavoro.

La combinazione di IaC, pipeline di distribuzione automatizzate e dimensionamento automatico consente alle organizzazioni di effettuare il provisioning, l'aggiornamento e il dimensionamento dei propri ambienti in tutta tranquillità. Questa automazione è essenziale per mantenere un'infrastruttura cloud reattiva, resiliente e gestita in modo efficiente.

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

Guida all'implementazione

Per configurare l'automazione con le pipeline CI/CD e infrastructure as code (IaC) per l'architettura AWS, scegli un sistema di controllo delle versioni come Git per archiviare i modelli e la configurazione IaC. Questi modelli possono essere scritti utilizzando strumenti quali AWS CloudFormation. Per iniziare, definisci i componenti dell'infrastruttura (ad esempio, AWS VPC, gruppi Amazon EC2 Auto Scaling e database Amazon RDS) all'interno di questi modelli.

Successivamente, integra questi modelli IaC con una pipeline CI/CD per automatizzare il processo di implementazione. AWS CodePipeline fornisce una soluzione AWS nativa senza soluzione di continuità oppure puoi utilizzare altre soluzioni CI/CD di terze parti. Crea una pipeline che si attivi quando vengono apportate modifiche al repository di controllo delle versioni. Configura la pipeline in modo da includere le fasi di lint e validazione dei modelli IaC, distribuisci l'infrastruttura in un ambiente di gestione temporanea, esegui i test automatici e infine distribuisci in produzione. Incorpora le fasi di approvazione ove necessario per mantenere il controllo sulle modifiche. Questa pipeline automatica non solo accelera l'implementazione, ma facilita anche la coerenza e l'affidabilità tra gli ambienti.

Configura il dimensionamento automatico di risorse come istanze Amazon EC2, attività Amazon ECS e repliche di database nell'ambiente IaC per garantire aumento e riduzione orizzontale secondo necessità. Questo approccio migliora la disponibilità e le prestazioni delle applicazioni e ottimizza i costi regolando dinamicamente le risorse in base alla domanda. Per un elenco delle risorse supportate, consulta Amazon EC2 Auto Scaling e AWS Auto Scaling.

Passaggi dell'implementazione

1. Crea e utilizza un repository del codice sorgente per archiviare il codice che controlla la configurazione dell'infrastruttura. Esegui il commit delle modifiche a questo repository per riflettere eventuali modifiche in corso da apportare.
2. Seleziona una soluzione infrastructure as code (IaC) come AWS CloudFormation per mantenere l'infrastruttura aggiornata e rilevare le incoerenze (deviazione) rispetto allo stato previsto.
3. Integra la piattaforma IaC con la pipeline CI/CD per automatizzare le implementazioni.
4. Determina e raccogli le metriche appropriate per il dimensionamento automatico delle risorse.
5. Configura il dimensionamento automatico delle risorse utilizzando policy appropriate per aumentare orizzontalmente e ridurre orizzontalmente le risorse per i componenti del carico di lavoro. Considera l'utilizzo di un dimensionamento pianificato per modelli di utilizzo prevedibili.

6. Monitora le implementazioni per rilevare guasti e regressioni. Implementa meccanismi di ripristino dello stato precedente all'interno della piattaforma CI/CD per annullare le modifiche, se necessario.

Risorse

Documenti correlati:

- AWS Auto Scaling: How Scaling Plans Work
- Marketplace AWS: prodotti che possono essere utilizzati con Auto Scaling
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- Uso di un bilanciatore del carico con un gruppo Auto Scaling
- What Is AWS Global Accelerator?
- What Is Amazon EC2 Auto Scaling?
- Che cos'è AWS Auto Scaling?
- What is Amazon CloudFront?
- What is Amazon Route 53?
- Cos'è l'Elastic Load Balancing?
- Cos'è un Network Load Balancer?
- Cos'è un Application Load Balancer?
- Integrating Jenkins with AWS CodeBuild and AWS CodeDeploy
- Creating a four stage pipeline with AWS CodePipeline

Video correlati:

- Back to Basics: Deploy Your Code to Amazon EC2
- AWS Supports You | Starting Your Infrastructure as Code Solution Using AWS CloudFormation Templates
- Streamline Your Software Release Process Using AWS CodePipeline
- Monitor AWS Resources Using Amazon CloudWatch Dashboards
- Create Cross Account & Cross Region CloudWatch Dashboards | Amazon Web Services

REL07-BP02 Ottenere risorse dopo aver rilevato una compromissione del carico di lavoro

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

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

La scalabilità può essere regolata manualmente in base al carico di lavoro (ad esempio, modificando il numero di EC2 istanze in un gruppo Auto Scaling o modificando il throughput di una tabella DynamoDB tramite o). AWS Management Console AWS CLI Tuttavia, è opportuno ricorrere all'automazione ogni volta che è possibile (consulta Utilizzo dell'automazione per l'acquisizione o il dimensionamento delle risorse).

Risultato desiderato: avvio di operazioni di dimensionamento (in automatico o manualmente) per il ripristino della disponibilità in caso di rilevamento di un guasto o di un peggioramento dell'esperienza del cliente.

Livello di rischio associato se questa best practice non fosse adottata: 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 automatizzate, che ridimensionano le risorse richieste. o Per ulteriori informazioni, consulta REL11-BP01 Monitora tutti i componenti del carico di lavoro per rilevare eventuali guasti.

Passaggi dell'implementazione

- Definisci le procedure (manuali o automatiche) per scalare 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 utilizzati AWS Auto Scaling possono utilizzare piani di scalabilità per configurare una serie di istruzioni per scalare le risorse. Se utilizzi AWS CloudFormation o aggiungi tag alle AWS risorse, puoi impostare piani di ridimensionamento per diversi set di risorse

per applicazione. Auto Scaling fornisce raccomandazioni per strategie di dimensionamento personalizzate per ogni risorsa. Dopo aver creato il piano di dimensionamento, Auto Scaling combina i metodi di dimensionamento dinamico e predittivo per supportare la tua strategia di dimensionamento. Per ulteriori informazioni, consulta How scaling plans work.

- Amazon EC2 Auto Scaling verifica che tu abbia il numero corretto di EC2 istanze Amazon disponibili per gestire il carico della tua applicazione. Si creano raccolte di EC2 istanze, chiamate gruppi di Auto Scaling. Puoi specificare il numero minimo e massimo di istanze in ogni gruppo di Auto Scaling e Amazon Auto EC2 Scaling garantisce che il tuo gruppo non superi o superi mai questi limiti. Per ulteriori dettagli, consulta Cos'è Amazon EC2 Auto Scaling?
- La scalabilità automatica di Amazon DynamoDB utilizza il servizio Application Auto Scaling per regolare in modo dinamico la capacità effettiva di trasmissione allocata per conto tuo in risposta ai modelli di traffico effettivi. In tal modo una tabella o un indice secondario globale può aumentare la capacità di lettura e scrittura allocata per gestire improvvisi aumenti di traffico, senza limitazione (della larghezza di banda della rete). Per ulteriori dettagli, consulta Managing throughput capacity automatically with DynamoDB auto scaling.

Risorse

Best practice correlate:

- REL07-BP01 Usa l'automazione per ottenere o scalare le risorse
- REL11-BP01 Monitora tutti i componenti del carico di lavoro per rilevare i guasti

Documenti correlati:

- AWS Auto Scaling: How Scaling Plans Work
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- Che cos'è Amazon EC2 Auto Scaling?

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

Una delle caratteristiche più preziose del cloud computing è la capacità di fornire risorse in modo dinamico.

Negli ambienti di calcolo tradizionali on-premises, è necessario identificare e fornire in anticipo la capacità sufficiente per soddisfare i picchi di domanda. Questo è un problema perché è costoso e perché comporta rischi per la disponibilità se si sottovalutano le esigenze di capacità di picco del carico di lavoro.

Nel cloud non è necessario farlo. Invece, è possibile allocare capacità di calcolo, database e altre risorse in base alle esigenze per soddisfare la domanda attuale e prevista. Soluzioni automatizzate come Amazon EC2 Auto Scaling e Application Auto Scaling possono portare automaticamente le risorse online in base a metriche specificate dall'utente. Ciò può rendere il processo di dimensionamento più semplice e prevedibile e può rendere il carico di lavoro significativamente più affidabile garantendo che siano sempre disponibili risorse sufficienti.

Risultato desiderato: configuri il dimensionamento automatico delle risorse di calcolo e di altro tipo per soddisfare la domanda. Nelle policy di dimensionamento fornisci un margine sufficiente per consentire la gestione di picchi di traffico mentre vengono messe online altre risorse.

Anti-pattern comuni:

- Fornisci un numero fisso di risorse scalabili.
- Scegli una metrica di dimensionamento che non è correlata alla domanda effettiva.
- I piani di dimensionamento non prevedono un margine sufficiente per supportare picchi di domanda.
- Le policy di dimensionamento aggiungono capacità troppo tardi, con conseguente esaurimento della capacità e degrado del servizio mentre altre risorse vengono messe online.
- Non riesci a configurare correttamente il conteggio delle risorse minime e massime, con conseguenti errori di dimensionamento.

Vantaggi dell'adozione di questa best practice: disporre di risorse sufficienti per soddisfare la domanda attuale è fondamentale per fornire un'elevata disponibilità del carico di lavoro e rispettare gli obiettivi di livello di servizio (SLO) definiti. Il dimensionamento automatico consente di fornire la giusta quantità di calcolo, database e altre risorse necessarie al carico di lavoro per soddisfare la domanda attuale e prevista. Non è necessario determinare le esigenze di capacità di picco e allocare staticamente le risorse per soddisfarle. Invece, man mano che la domanda cresce, è possibile allocare più risorse per soddisfarla e, quando la domanda diminuisce, è possibile disattivare le risorse per ridurre i costi.

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

Guida all'implementazione

In primo luogo, è necessario determinare se il componente del carico di lavoro è adatto al dimensionamento automatico. Questi componenti sono chiamati a scalabilità orizzontale perché forniscono le stesse risorse e si comportano in modo identico. Esempi di componenti a scalabilità orizzontale includono le istanze EC2 configurate allo stesso modo, attività Amazon Elastic Container Service (ECS) e pod in esecuzione su Amazon Elastic Kubernetes Service (EKS). Queste risorse di calcolo sono in genere collocate dietro un bilanciatore del carico e vengono chiamate repliche.

Altre risorse replicate possono includere repliche di lettura del database, tabelle Amazon DynamoDB e cluster Amazon ElastiCache (Redis OSS). Per un elenco completo delle risorse supportate, consulta la pagina relativa ai servizi AWS che puoi utilizzare con Application Auto Scaling.

Per le architetture basate su container, potrebbe essere necessario scalare in due modi diversi. In primo luogo, potrebbe essere necessario scalare i container che forniscono servizi scalabili orizzontalmente. In secondo luogo, potrebbe essere necessario scalare le risorse di calcolo per creare spazio per nuovi container. Per ogni livello esistono diversi meccanismi di dimensionamento automatico. Per scalare le attività ECS, puoi utilizzare Application Auto Scaling. Per scalare i pod Kubernetes, puoi utilizzare Horizontal Pod Autoscaler (HPA) o Kubernetes Event-driven Autoscaling (KEDA). Per scalare le risorse di calcolo, puoi utilizzare Provider di capacità per ECS o puoi utilizzare Karpenter o Cluster Autoscaler per Kubernetes.

Quindi, seleziona la modalità di esecuzione del dimensionamento automatico. Esistono tre opzioni principali: dimensionamento basato su metriche, dimensionamento pianificato e dimensionamento predittivo.

Dimensionamento basato su metriche

Il dimensionamento basato su metriche fornisce le risorse in base al valore di una o più metriche di dimensionamento. Una metrica di dimensionamento è quella che corrisponde alla domanda del carico di lavoro. Un buon modo per determinare le metriche di dimensionamento appropriate è quello di eseguire test di carico in un ambiente non di produzione. Durante i test di carico, mantieni fisso il numero di risorse scalabili e aumenta lentamente la domanda (ad esempio, il throughput, la simultaneità o gli utenti simulati). Cerca quindi le metriche che aumentano (o diminuiscono) con l'aumento della domanda e, viceversa, che diminuiscono (o aumentano) con il calo della domanda. Le metriche di dimensionamento tipiche includono l'utilizzo della CPU, la profondità della coda di lavoro (ad esempio una coda Amazon SQS), il numero di utenti attivi e il throughput di rete.

> **Note**
>
> AWS ha osservato che con la maggior parte delle applicazioni, l'utilizzo della memoria aumenta durante la preparazione dell'applicazione per poi raggiunge un valore costante. Quando la domanda diminuisce, l'utilizzo della memoria rimane in genere elevato anziché diminuire in parallelo. Poiché l'utilizzo della memoria non corrisponde alla domanda in entrambe le direzioni, ovvero cresce e diminuisce con la domanda, valuta attentamente questa metrica prima di selezionarla per il dimensionamento automatico.

Il dimensionamento basato sulle metriche è un'operazione latente. Le metriche di utilizzo possono impiegare diversi minuti per propagarsi ai meccanismi di dimensionamento automatico, che in genere attendono un chiaro segnale di aumento della domanda prima di reagire. Man mano che l'autoscaler crea nuove risorse, può essere necessario del tempo aggiuntivo per raggiungere il pieno servizio. Per questo motivo, è importante non impostare i target delle metriche di dimensionamento troppo vicini all'utilizzo completo (ad esempio, 90% di utilizzo della CPU). Così facendo, si rischia di esaurire la capacità delle risorse esistenti prima che una capacità aggiuntiva possa essere messa online. Gli obiettivi tipici di utilizzo delle risorse possono variare tra il 50 e il 70% per una disponibilità ottimale, a seconda dei modelli di domanda e del tempo necessario per effettuare il provisioning di risorse aggiuntive.

Dimensionamento pianificato

Il dimensionamento pianificato fornisce o rimuove le risorse in base al calendario o all'ora del giorno. È spesso utilizzato per i carichi di lavoro che hanno una domanda prevedibile, come i picchi di utilizzo durante le ore di lavoro nei giorni feriali o gli eventi di vendita. Sia Amazon EC2 Auto Scaling che Application Auto Scaling supportano il dimensionamento pianificato. Il cron scaler di KEDA supporta il dimensionamento pianificato dei pod Kubernetes.

Dimensionamento predittivo

Il dimensionamento predittivo utilizza il machine learning per scalare automaticamente le risorse in base alla domanda prevista. Il dimensionamento predittivo analizza il valore storico di una metrica di utilizzo fornita dall'utente e ne prevede continuamente il valore futuro. Il valore previsto viene quindi utilizzato per scalare la risorsa verso l'alto o verso il basso. Amazon EC2 Auto Scaling può eseguire il dimensionamento predittivo.

Passaggi dell'implementazione

1. Determina se il componente del carico di lavoro è adatto al dimensionamento automatico.
2. Determina il tipo di meccanismo di dimensionamento più appropriato per il carico di lavoro: dimensionamento basato sulle metriche, dimensionamento pianificato o dimensionamento predittivo.
3. Seleziona il meccanismo di dimensionamento automatico appropriato per il componente. Per le istanze Amazon EC2, utilizza Amazon EC2 Auto Scaling. Per altri servizi AWS, utilizza Application Auto Scaling. Per i pod Kubernetes (come quelli in esecuzione in un cluster Amazon EKS), prendi in considerazione Horizontal Pod Autoscaler (HPA) o Kubernetes Event-driven Autoscaling (KEDA). Per i nodi Kubernetes o EKS, prendi in considerazione Karpenter e Cluster Auto Scaler (CAS).
4. Per il dimensionamento basato sulle metriche o pianificato, esegui test di carico per determinare le metriche di dimensionamento e i valori di destinazione appropriati per il carico di lavoro. Per il dimensionamento pianificato, determina il numero di risorse necessarie alle date e agli orari selezionati. Determina il numero massimo di risorse necessarie per servire i picchi di traffico previsti.
5. Configura l'autoscaler in base alle informazioni raccolte in precedenza. Per maggiori dettagli, consulta la documentazione del servizio di dimensionamento automatico. Verifica che i limiti di dimensionamento massimo e minimo siano configurati correttamente.
6. Verifica che la configurazione di dimensionamento funzioni come previsto. Esegui i test di carico in un ambiente non di produzione e osserva come reagisce il sistema, regolandolo se necessario. Quando abiliti il dimensionamento automatico in produzione, configura gli allarmi appropriati per segnalare qualsiasi comportamento imprevisto.

Risorse

Documenti correlati:

- What Is Amazon EC2 Auto Scaling?
- AWS Prescriptive Guidance: Load testing applications
- Marketplace AWS: prodotti che possono essere utilizzati con Auto Scaling
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- Scheduled Scaling for Amazon EC2 Auto Scaling

- Telling Stories About Little's Law

REL07-BP04 Load Testa il tuo 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 dovrebbero scoprire il punto di rottura e testare le prestazioni del carico di lavoro. AWS semplifica la configurazione di ambienti di test temporanei che modellano 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 rispetti ai test on-premises.

I test di carico in produzione dovrebbero anche essere considerati come parte delle giornate di gioco 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 implementazioni 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.

- 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:

- Test di carico distribuito su AWS: simula migliaia di utenti connessi
- Load testing applications

Video correlati:

- AWS Summit ANZ 2023: accelera con sicurezza grazie ai test di carico AWS distribuiti

Implementazione delle modifiche

Le modifiche controllate sono necessarie per implementare nuove funzionalità e 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.

Modelli di implementazione aggiuntivi per ridurre al minimo i rischi

I flag delle funzionalità (noti anche come interruttori funzionalità) sono opzioni di configurazione in un'applicazione. È possibile implementare il software con una funzionalità disattivata, in modo che i clienti non la visualizzino. Puoi quindi attivare la funzionalità, come faresti per una distribuzione

canary, oppure impostare il ritmo di modifica su 100% per vedere l'effetto. Se l'implementazione presenta problemi, è possibile semplicemente disattivare la funzionalità senza eseguire il rollback.

Implementazione zonale isolata con errori: una delle regole più importanti che AWS ha stabilito per le proprie implementazioni è evitare di toccare allo stesso momento più zone di disponibilità all'interno di una regione. Ciò è fondamentale per garantire che le zone di disponibilità siano indipendenti ai fini dei nostri calcoli di disponibilità. Si consiglia di utilizzare considerazioni simili nelle implementazioni.

Revisioni sulla prontezza operativa (ORR)

AWS ritiene utile eseguire revisioni di prontezza operativa che valutino la completezza dei test, la capacità di monitorare e, soprattutto, la capacità di controllare le prestazioni dell'applicazioni rispetto ai suoi SLA e fornire dati in caso di interruzione o di altre anomalie operative. Una ORR formale viene condotta prima dell'implementazione iniziale in produzione. AWS ripeterà periodicamente le ORR (una volta all'anno o prima dei periodi critici di prestazione) per garantire che non ci sia stata "deriva" dalle aspettative operative. Per ulteriori informazioni sulla prontezza operativa, consulta il pilastro dell'eccellenza operativa del Framework AWS Well-Architected.

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 Integra i test di resilienza come parte della tua 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, mettere in atto processi per garantire la sicurezza del rollback durante le implementazioni. Garantire la possibilità di eseguire il rollback di un'implementazione 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 richiamalo per l'esecuzione automatica, se necessario.

Anche per carichi di lavoro sofisticati e altamente automatizzati, i runbook sono ancora utili per l'esecuzione di giornate di gioco o per 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: la pianificazione efficace aumenta la capacità di eseguire correttamente le modifiche, perché sei a conoscenza di tutti i sistemi interessati. Convalidare le modifiche negli ambienti di test aumenta la tua sicurezza.

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

Guida all'implementazione

- Fornisci risposte coerenti e tempestive a eventi noti documentando le procedure nei runbook.
  - Framework AWS Well-Architected, concetti: runbook
- Usa il principio di Infrastructure as code 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
      - Controlling access with AWS Identity and Access Management
    - Utilizza un sistema di gestione del codice sorgente ospitato basato su una tecnologia popolare come Git per archiviare il codice sorgente e la configurazione infrastructure as code (IaC).

Risorse

Documenti correlati:

- Partner APN: partner per la creazione di soluzioni di implementazione automatizzate
- Marketplace AWS: prodotti per l'automazione delle implementazioni
- Framework AWS Well-Architected, concetti: runbook
- Che cos'è AWS CloudFormation?

Esempi correlati:

- Automazione delle operazioni con playbook e runbook

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

Utilizza tecniche come i test di unità e i test di integrazione per convalidare le funzionalità richieste.

Il test di unità è un processo in cui si testa la più piccola unità funzionale di codice per convalidarne il comportamento. I test di integrazione cercano di convalidare che ogni funzionalità dell'applicazione operi secondo i requisiti del software. Mentre i test di unità si concentrano sulla verifica di una parte dell'applicazione in modo isolato, i test di integrazione considerano gli effetti collaterali (ad esempio, l'effetto della modifica dei dati attraverso un'operazione di mutazione). In entrambi i casi, i test devono essere integrati in una pipeline di implementazione e, se i criteri di esito positivo non sono soddisfatti, la pipeline viene interrotta o ripristinata. Questi test vengono eseguiti in un ambiente di pre-produzione, gestito per fasi prima della produzione nella pipeline.

Puoi ottenere i migliori risultati quando questi test vengono eseguiti automaticamente come parte delle operazioni di sviluppo e implementazione. Ad esempio, con AWS CodePipeline, gli sviluppatori affidano le modifiche a un repository di origine in cui CodePipeline rileva automaticamente le modifiche. L'applicazione viene creata e i test di unità vengono eseguiti. Dopo che i test di unità sono stati superati, il codice creato viene distribuito sui server di gestione temporanea per il test. Dal server temporaneo, CodePipeline esegue più test, ad esempio test di integrazione o caricamento. Una volta completati con successo i test, CodePipeline distribuisce il codice testato e approvato alle istanze di produzione.

Risultato desiderato: utilizzi l'automazione per eseguire test di unità e di integrazione per verificare che il codice si comporti come previsto. Questi test sono integrati nel processo di implementazione e un errore del test interrompe l'implementazione.

Anti-pattern comuni:

- Ignori o aggiri gli errori di test e i piani durante il processo di implementazione per accelerare la tempistica di implementazione.
- I test vengono eseguiti manualmente al di fuori della pipeline di implementazione.
- Non esegui le fasi di test nell'automazione tramite i flussi di lavoro manuali di emergenza.
- Esegui i test automatici in un ambiente che non assomiglia molto all'ambiente di produzione.
- Crei una suite di test non sufficientemente flessibile e difficile da mantenere, aggiornare o scalare con l'evoluzione dell'applicazione.

Vantaggi dell'adozione di questa best practice: i test automatici durante il processo di implementazione individuano tempestivamente i problemi, riducendo il rischio di un rilascio in produzione con bug o comportamenti imprevisti. I test di unità verificano che il codice si comporti come desiderato e che i contratti API siano rispettati. I test di integrazione convalidano il funzionamento del sistema in base ai requisiti specificati. Questi tipi di test verificano il funzionamento previsto di componenti quali interfacce utente, API, database e codice sorgente.

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

Guida all'implementazione

Adotta un approccio alla scrittura del software basato sullo sviluppo guidato dai test (TDD, Test-Driven Development), in cui sviluppi casi di test per specificare e convalidare il codice. Per iniziare, crea casi di test per ogni funzione. Se il test non va a buon fine, scrivi un nuovo codice per superare il test. Questo approccio consente di convalidare il risultato atteso di ciascuna funzione. Esegui i test di unità e verifica che vengano superati prima di eseguire il commit del codice in un repository del codice sorgente.

Implementa test di unità e di integrazione come parte delle fasi di compilazione, test e implementazione della pipeline CI/CD. Automatizza i test e avvia automaticamente i test ogni volta che una nuova versione dell'applicazione è pronta per essere implementata. Se non si soddisfano i criteri di esito positivo, la pipeline si arresta o viene sottoposta a rollback.

Se l'applicazione è un'app web o per dispositivi mobili, esegui test di integrazione automatizzati su più browser desktop o dispositivi reali. Questo approccio è particolarmente utile per convalidare la compatibilità e la funzionalità delle app per dispositivi mobili su una vasta gamma di dispositivi.

Passaggi dell'implementazione

1. Scrivi test di unità prima di scrivere codice funzionale (sviluppo basato su test o TDD). Stabilisci linee guida per il codice in modo che la scrittura e l'esecuzione di test di unità siano un requisito di codifica non funzionale.
2. Crea una suite di test di integrazione automatizzati che coprano le funzionalità testabili identificate. Questi test devono simulare le interazioni degli utenti e convalidare i risultati attesi.
3. Crea l'ambiente di test necessario per eseguire i test di integrazione. Questo può includere ambienti di gestione temporanea o di pre-produzione che simulano fedelmente l'ambiente di produzione.
4. Configura le fasi di origine, compilazione, test e distribuzione utilizzando la console AWS CodePipeline o AWS Command Line Interface (CLI).
5. Distribuisci l'applicazione una volta che il codice è stato compilato e testato. AWS CodeDeploy può distribuirla negli ambienti di gestione temporanea (test) e di produzione. Questi ambienti possono includere istanze Amazon EC2, funzioni AWS Lambda o server on-premises. Per distribuire l'applicazione in tutti gli ambienti si deve utilizzare lo stesso meccanismo di implementazione.
6. Monitora l'andamento della pipeline e lo stato di ogni fase. Utilizza i controlli di qualità per bloccare la pipeline in base allo stato dei test. Puoi inoltre ricevere notifiche per qualsiasi errore che si verifica durante l'esecuzione o il completamento della pipeline.
7. Monitora costantemente i risultati dei test e cerca modelli, regressioni o aree che richiedono maggiore attenzione. Utilizza queste informazioni per migliorare la suite di test, identificare le aree dell'applicazione che richiedono test più approfonditi e ottimizza il processo di implementazione.

Risorse

Best practice correlate:

- REL07-BP04 Esecuzione di un test di carico sul carico di lavoro
- REL08-BP03 Esecuzione di test di resilienza come parte integrante dell'implementazione
- REL12-BP04 Test della resilienza tramite l'utilizzo dell'ingegneria del caos

Documenti correlati:

- AWS Prescriptive Guidance: Test automation
- Continuous Delivery and Continuous Integration
- Indicators for functional testing
- Monitoring pipelines
- Use AWS CodePipeline with AWS CodeBuild to test code and run builds
- AWS Device Farm

REL08-BP03 Integra i test di resilienza come parte della tua 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: maggiore fiducia nella capacità del sistema di resistere al degrado nella produzione grazie ai test di resilienza. 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 delle best practice: i test di resilienza integrati nelle implementazioni consentono di identificare problemi sconosciuti nel sistema che altrimenti passerebbero inosservati, con conseguenti tempi di inattività nella 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 aggiornamenti ai piani di disaster recovery e alle procedure operative standard (SOPs) in 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 al fine di valutare in modo continuo e automatico la resilienza nell'ambito di ogni implementazione.
- Definisci le tue applicazioni in AWS Resilience Hub. Le valutazioni della resilienza generano frammenti di codice che ti aiutano a creare procedure di ripristino come documenti di AWS Systems Manager per le tue applicazioni e forniscono un elenco di monitor e allarmi Amazon CloudWatch consigliati.
- Una volta aggiornati i piani di disaster recovery e SOPs i tuoi piani di disaster recovery, completa i test di disaster recovery 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 disaster recovery, ti consigliamo di definire il Recovery Time Objective (RTO) e il Recovery Point Objective (RPO) per semplificare la scelta della strategia da simulare. AWS offre strumenti di disaster recovery AWS Elastic Disaster Recoveryper aiutarvi a iniziare con 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. Analogamente alle altre strategie, esegui simulazioni controllate di guasti in ambienti non di produzione, con 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
- Valutazione continua della resilienza delle applicazioni con e AWS Resilience HubAWS CodePipeline
- Architettura di disaster recovery (DR) attiva AWS, parte 1: Strategie per il ripristino nel cloud
- Verify the resilience of your workloads using Chaos Engineering
- Principles of Chaos Engineering
- Workshop su Chaos Engineering

Video correlati:

- AWS re:Invent 2020: Testing Resilience using Chaos Engineering
- Migliora la resilienza delle applicazioni con il servizio AWS Fault Injection
- Prepara e proteggi le tue applicazioni dalle interruzioni con 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 gli ambienti: l'assenza di differenze nelle risorse dell'infrastruttura tra ambienti garantisce l'aumento della coerenza e la semplificazione dei test.
- Riduzione delle deviazioni di configurazione: sostituendo le risorse dell'infrastruttura con una configurazione nota e controllata dalla versione, l'infrastruttura viene impostata a uno stato noto, testato e affidabile, evitando deviazioni di configurazione.
- Implementazioni atomiche affidabili: il completamento delle implementazioni avviene con successo o non cambia nulla, così da aumentare coerenza e affidabilità del processo di implementazione.
- Implementazioni semplificate: le implementazioni sono semplificate poiché non devono supportare gli aggiornamenti. Gli aggiornamenti sono solo nuove implementazioni.
- Implementazioni più sicure con processi di rollback e ripristino rapidi: le implementazioni sono più sicure poiché la versione funzionante precedente non viene modificata. Puoi eseguire il rollback se vengono rilevati errori.
- Potenziamento del profilo di sicurezza: non consentire modifiche all'infrastruttura si traduce nella possibilità di disabilitare i meccanismi di accesso remoto (come SSH). Questo riduce il vettore di attacco, migliorando il profilo di sicurezza dell'organizzazione.

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

Guida all'implementazione

Automazione

Nel definire una strategia di implementazione dell'infrastruttura immutabile, si consiglia di utilizzare il più possibile l'automazione per aumentare la riproducibilità e ridurre al minimo i potenziali errori umani. Per maggiori dettagli, consulta REL08-BP05 Implementazione delle modifiche tramite automazione e Automatizzazione di distribuzioni 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 conservate 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.

Modelli 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:

Distribuzione canary: è la 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 a implementare alla velocità desiderata, monitorando le modifiche alla ricerca di errori, fino a quando l'implementazione non sarà completata. AWS CodeDeploy può essere configurato con una configurazione di implementazione che consente una distribuzione canary.

Implementazione blu/verde: simile alla distribuzione canary, tranne per il fatto che un intero parco dell'applicazione è implementato in parallelo. Puoi alternare le implementazioni tra i due stack (blu e verde). Ancora una volta, puoi inviare il traffico alla nuova versione e tornare alla versione precedente in caso di problemi con l'implementazione. Generalmente, tutto il traffico viene trasferito contemporaneamente, tuttavia puoi anche utilizzare frazioni del traffico verso ciascuna versione per accelerare l'adozione della nuova versione mediante le funzionalità di instradamento DNS ponderato di Amazon Route 53. AWS CodeDeploy e AWS Elastic Beanstalk possono essere impostati con una configurazione di implementazione che consente un'implementazione blu/verde.

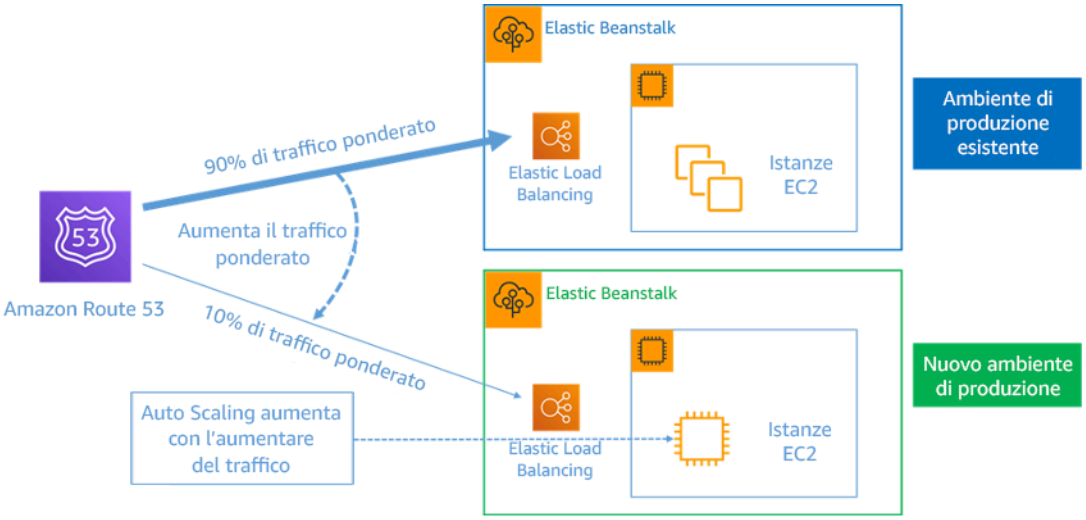


Figura 8: Implementazione blu/verde con AWS Elastic Beanstalk e Amazon Route 53

Rilevamento delle deviazioni

Per deviazione si intende qualsiasi modifica che causa uno stato o una configurazione di una risorsa dell'infrastruttura diversi da quelli previsti. 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 a livello centrale le autorizzazioni in modo granulare 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 illustrato nel whitepaper Introduzione a DevOps in AWS, l'automazione è fondamentale per i servizi AWS ed è supportata a livello interno in tutti i servizi, le funzionalità e le offerte.
  - La preparazione preliminare di Amazon Machine Image (AMI) può velocizzare i tempi di avvio. EC2 Image Builder è un servizio AWS completamente gestito che consente di automatizzare creazione, manutenzione, convalida, condivisione e 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 scalare rapidamente applicazioni e servizi Web sviluppati con Java, .NET, PHP, Node.js, Python, Ruby, Go e Docker su server comuni come Apache, NGINX, Passenger e IIS.
    - AWS Proton consente ai team della piattaforma di connettere e coordinare tutti i vari strumenti necessari ai team di sviluppo per il provisioning dell'infrastruttura, l'implementazione del codice, il monitoraggio e gli aggiornamenti. AWS Proton abilita il provisioning e l'implementazione basati sul modello Infrastructure as code 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 consente agli sviluppatori di 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. Crei le risorse in formato JSON o YAML con qualsiasi editor di codice e le inserisci in un sistema di controllo delle versioni. A questo punto, AWS CloudFormation crea i servizi specificati in modo sicuro e ripetibile.
    - AWS Serverless Application Model (AWS SAM) è un framework open source utilizzabile per la creazione di applicazioni serverless in AWS. AWS SAM si integra con altri servizi AWS, oltre a essere un'estensione di AWS CloudFormation.
    - AWS Cloud Development Kit (AWS CDK) è un framework di sviluppo software open source per modellare ed eseguire il provisioning delle risorse delle applicazioni cloud utilizzando linguaggi di programmazione familiari. 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 e List (CRUDL) per consentire agli sviluppatori di gestire la propria infrastruttura cloud in modo semplice e coerente. Le API comuni (API Cloud Control) 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.
    - Distribuzione canary:
        - Set up an API Gateway canary release deployment
        - Create a pipeline with canary deployments for Amazon ECS using AWS App Mesh

  - Implementazioni blu/verde: il whitepaper Blue/Green Deployments on AWS riporta tecniche esemplificative per implementare strategie di implementazione blu/verde.
- Rileva le deviazioni a livello di configurazione o stato. Per ulteriori informazioni, consulta Detecting unmanaged configuration changes to stacks and resources.

Risorse

Best practice correlate:

- REL08-BP05 Implementazione delle modifiche tramite automazione

Documenti correlati:

- Automatizzazione di distribuzioni pratiche e sicure
- Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank
- Infrastructure as code
- Implementing an alarm to automatically detect drift in AWS CloudFormation stacks

Video correlati:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

REL08-BP05 Implementazione delle modifiche tramite automazione

Le implementazioni 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: integrazione della sicurezza dell'implementazione automatizzata nel processo di rilascio con test di pre-produzione completi, 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 il tempo di incorporamento.

Vantaggi dell'adozione di questa best practice: l'utilizzo dell'automazione per implementare tutte le modifiche elimina la possibilità di introdurre errori umani, oltre a offrire la possibilità di eseguire test prima di apportare modifiche alla 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 implementazione 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 di ciò è l'adozione della cultura basata sull'integrazione continua e sulla consegna/implementazione continua (CI/CD), dove un commit o una modifica del codice passa lungo varie fasi automatizzate, dalle fasi di creazione e test, fino 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 conservare il codice in modo sicuro: utilizza un sistema di gestione del codice sorgente ospitato basato su una tecnologia popolare come Git per memorizzare il codice sorgente e la configurazione infrastructure as code (IaC).
- Configura un servizio di integrazione continua per compilare il codice sorgente, eseguire test e creare artefatti di implementazione: per configurare un progetto di compilazione a tale scopo, consulta Getting started with AWS CodeBuild using the console.

- Configura un servizio di implementazione in grado di automatizzare le implementazioni delle applicazioni e gestire la complessità degli aggiornamenti delle stesse senza fare affidamento su implementazioni manuali soggette a errori: AWS CodeDeploy automatizza le implementazioni software in svariati servizi di calcolo, come Amazon EC2, AWS Fargate, AWS Lambda e i tuoi server on-premises. Per la configurazione di questi passaggi, consulta Getting started with CodeDeploy.
- Imposta un servizio di distribuzione continua in grado di automatizzare le pipeline di rilascio per aggiornamenti più rapidi e affidabili delle applicazioni e dell'infrastruttura: prendi in considerazione l'utilizzo di AWS CodePipeline per automatizzare le tue 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
- Partner APN: partner per la creazione di soluzioni di implementazione automatizzate
- Marketplace AWS: prodotti per l'automazione delle implementazioni
- Automatizza i messaggi delle chat con webhook
- Amazon Builders' Library: garantire la sicurezza del rollback durante le distribuzioni
- Amazon Builders' Library: più velocità con una consegna continua
- Che cos'è 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 dei guasti

> 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. Questi sono dati scontati, indipendentemente dal fatto che si stia utilizzando hardware di alta qualità o componenti a basso costo, Werner Vogels, Direttore Tecnico (CTO) - Amazon.com

I guasti dei componenti hardware di basso livello vengono risolti ogni giorno in un data center on-premises. Nel cloud, tuttavia, devi essere protetto dalla maggior parte di questi tipi di guasti. Ad esempio, i volumi Amazon EBS vengono collocati in una zona di disponibilità specifica in cui avviene la loro replica in automatico per proteggerti dai guasti di un singolo componente. Tutti i volumi EBS sono progettati per garantire una disponibilità del 99,999%. Gli oggetti di Amazon S3 vengono archiviati in almeno tre zone di disponibilità, garantendo una durabilità degli oggetti pari al 99,999999999% per un determinato anno. Indipendentemente dal provider di servizi cloud, è possibile che si verifichino guasti che influiscono sul tuo carico di lavoro. Pertanto, occorre 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 di affidabilità in modo da conseguirli. Queste persone devono essere informate e addestrate per questi requisiti di affidabilità.

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

Argomenti

Esecuzione del 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 utilizza le funzionalità di backup dei servizi e delle risorse di dati usati dal carico di lavoro. La maggior parte dei servizi offre funzionalità per eseguire il backup dei dati del carico di lavoro.

Risultato desiderato: le origini dati sono state identificate e classificate 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 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 dati per il carico di lavoro e della loro criticità.
- Non si eseguono backup delle origini dati critiche.
- Esecuzione di backup solo di alcune origini 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 i backup e l'implementazione di un meccanismo per la creazione di backup, o la possibilità di riprodurre i dati da una fonte esterna, migliorano la capacità di ripristinare e recuperare i dati durante un'interruzione.

Livello di rischio associato se questa best practice non fosse adottata: 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. Diversi servizi AWS offrono la possibilità di copiare i backup su un'altra Regione AWS. AWS Backup è uno strumento che permette di centralizzare e automatizzare la protezione dei dati tra i servizi AWS. AWS Elastic Disaster Recovery consente di copiare carichi di lavoro server completi e mantenere una protezione continua dei dati on-premises, tra diverse 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 dati gestite dal cliente e da AWS. I servizi AWS come Amazon EBS, Amazon RDS, e Amazon DynamoDB presentano funzionalità integrate per la creazione di backup. È anche possibile utilizzare software di backup di terze parti.

È possibile eseguire il backup dei dati on-premises nel Cloud AWS utilizzando AWS Storage Gateway o AWS DataSync. È possibile usare i bucket Amazon S3 per archiviare questi dati in AWS. Amazon S3 offre più livelli di archiviazione come Amazon S3 Glacier o S3 Glacier Deep Archive per ridurre i costi dell'archiviazione di dati.

Potresti essere in grado di soddisfare le esigenze di recupero dei dati riproducendo i dati da altre origini. Ad esempio, i nodi di replica di Amazon ElastiCache o le repliche di lettura di Amazon RDS consentono di riprodurre i dati in caso di perdita del nodo primario. In caso di possibile utilizzo di queste origini per soddisfare l'Obiettivo del punto di ripristino (RPO) e l'Obiettivo del tempo di ripristino (RTO), potrebbe non essere necessario un backup. Un altro esempio: con Amazon EMR, potrebbe non essere necessario eseguire il backup del data store HDFS, finché è 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. L'archiviazione dei dati può avvenire su varie risorse come database, volumi, file system, sistemi di log e storage a oggetti. Consulta la sezione Risorse per trovare i documenti correlati in merito ai vari servizi AWS di archiviazione dei dati e alle funzionalità di backup fornite da questi.
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. Utilizza i servizi AWS o di terze parti per creare backup dei dati. AWS Backup è un servizio gestito che consente la creazione di backup di varie origini dati su AWS. AWS Elastic Disaster Recovery gestisce la replica automatizzata dei dati in meno di un secondo in una Regione AWS. La maggior parte dei servizi AWS include anche funzionalità native per la creazione di backup. Marketplace AWS offre molte soluzioni che offrono anche queste funzionalità. Consulta la sezione Risorse più avanti per informazioni su come creare backup dei dati da vari 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 a un data warehouse (ad esempio Amazon Redshift) o a un cluster MapReduce (ad esempio Amazon EMR) per eseguire analisi su tali dati, questo può essere un esempio di dati che possono essere riprodotti da altre 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?
- What is AWS DataSync?
- What is Volume Gateway?
- Partner APN: partner per il backup
- Marketplace AWS: prodotti che possono essere utilizzati per il backup
- Snapshot Amazon EBS
- Backup Amazon EFS
- Backup di Amazon FSx per Windows File Server
- Backup e ripristino per ElastiCache for Redis
- Creating a DB Cluster Snapshot in Neptune
- Creating a DB Snapshot
- Creating an EventBridge Rule That Triggers on a Schedule
- Replica tra regioni con Amazon S3
- AWS Backup da EFS a EFS
- Exporting Log Data to Amazon S3
- Gestione del ciclo di vita degli oggetti
- Backup e ripristino on demand per DynamoDB
- Ripristino point-in-time per DynamoDB
- Uso di snapshot di indici del servizio OpenSearch di Amazon
- What is AWS Elastic Disaster Recovery?

Video correlati:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account and Cross-Region Backup

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

REL09-BP02 Protezione e crittografia dei backup

Controlla e rileva l'accesso ai backup utilizzando l'autenticazione e l'autorizzazione. Previeni 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 in caso di esposizione accidentale.

Livello di rischio associato se questa best practice non fosse adottata: 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). Previeni e rileva se l'integrità dei dati dei backup è compromessa utilizzando la crittografia.

Amazon S3 supporta diversi metodi di crittografia dei dati a riposo. 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 ad 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 vengono 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.
   - Utilizza 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.
   - Utilizza la crittografia sui volumi Amazon EBS. Puoi configurare la crittografia predefinita o specificare una chiave univoca al momento della creazione del volume.
   - Utilizza la crittografia di Amazon DynamoDB necessaria. DynamoDB codifica 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.
   - Codifica i 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 tenendo presente che le chiavi sono specifiche per regione. Puoi specificare le chiavi di destinazione quando configuri la replica.
   - Scegli se utilizzare la crittografia Amazon EBS per Elastic Disaster Recovery predefinita o personalizzata. 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 privilegio minimo per accedere ai backup. Segui le best practice per limitare l'accesso a backup, snapshot e repliche in conformità con le best practice di sicurezza.

Risorse

Documenti correlati:

- Marketplace AWS: prodotti che possono essere utilizzati per il backup
- Amazon EBS Encryption
- Amazon S3: protezione dei dati tramite la crittografia
- Configurazione aggiuntiva CRR: replica di oggetti creati con crittografia lato server (SSE) utilizzando le chiavi di crittografia archiviate in AWS KMS
- DynamoDB Encryption at Rest
- Encrypting Amazon RDS Resources
- Crittografia dei dati e dei metadati in Amazon EFS

- Encryption for Backups in AWS
- Gestione di tabelle crittografate
- Pilastro della sicurezza: Framework AWS Well-Architected
- What is AWS Elastic Disaster Recovery?

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 automatizzato che crea backup delle origini dati con una cadenza stabilita.

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 vengano eseguiti regolarmente in base all'RPO e avvisa se non vengono eseguiti.

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

Guida all'implementazione

AWS Backup consente di creare backup automatici di dati di varie origini 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 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 inoltre funzionalità di backup native. I servizi AWS con backup automatizzato e ripristino point-in-time includono Amazon DynamoDB, Amazon RDS e Amazon Keyspaces (per Apache Cassandra), il cui ripristino è possibile

in un momento specifico all'interno della cronologia di 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. Il controllo delle versioni di Amazon S3, una volta abilitato, è automatico. È possibile utilizzare 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 Image (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-premises 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-premises utilizzando AWS Storage Gateway.

Oltre al controllo delle versioni, Amazon S3 offre la funzionalità 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 al momento sottoposte a backup manuale. Per ulteriori dettagli, 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 dettagli, consulta REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati.
3. Utilizza una soluzione di backup automatico o un servizio gestito. AWS Backup è un servizio totalmente gestito che semplifica la centralizzazione e l'automatizzazione della protezione dei dati in tutti i servizi AWS, nel cloud e on-premises. 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 una guida pratica su come creare backup automatici con AWS Backup, consulta Testing Backup and Restore of Data. 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 automatico o da un servizio gestito, come le origini dati on-premises o le code di messaggi, è consigliabile utilizzare 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 utilizzare Amazon EventBridge per richiamare la stessa 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 utilizzati per il backup
- Creating an EventBridge Rule That Triggers on a Schedule
- Che cos'è AWS Backup?
- Che cos'è AWS Step Functions?
- What is AWS Elastic Disaster Recovery?

Video correlati:

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

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: i dati dei backup vengono ripristinati periodicamente utilizzando meccanismi ben definiti per verificare 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: il test del ripristino dei backup verifica che i dati possano essere ripristinati quando necessario senza preoccuparsi che possano essere mancanti o danneggiati, che il ripristino e il recupero siano possibili entro l'RTO per il carico di lavoro e che qualsiasi perdita di dati rientri nell'RPO per il carico di lavoro.

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

Se tutti i dati sono disponibili, non sono danneggiati, sono accessibili e qualsiasi perdita di dati rientra 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. Con la replica dei server di origine, gli stati point-in-time vengono registrati nel corso del tempo in base alla policy configurata. Elastic Disaster Recovery verifica 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 stai eseguendo il backup e dove sono 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 criteri per la convalida dei dati per ciascuna 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 dati al momento della creazione del backup.
3. Definisci l'RTO e l'RPO per il ripristino dei dati in base alla relativa criticità. 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. AWS Resilience Hub ti consente di valutare il tuo 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 utilizzando i processi attualmente in uso in produzione per il ripristino dei dati. Questi processi dipendono dal modo in cui è stato eseguito il backup dell'origine dati iniziale, dal formato e dalla posizione di archiviazione del backup stesso o dalla riproduzione dei dati da altre fonti. Ad esempio, in caso di utilizzo di un servizio gestito, come AWS Backup, potrebbe essere semplice ripristinare il backup in una nuova risorsa. In caso di utilizzo di AWS Elastic Disaster Recovery, è possibile avviare un'esercitazione di ripristino.
6. Convalida il ripristino dei dati dalla risorsa ripristinata in base ai criteri stabiliti 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 recupero e ripristino, quindi confrontalo con l'RTO stabilito. 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 hanno una datazione temporale 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 utilizzi un processo automatizzato, puoi sfruttare servizi come Amazon DynamoDB 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. All'interno di AWS Backup, le azioni di backup e di ripristino sono denominate processi. Questi contengono informazioni sulla data e l'ora come parte dei metadati che possono essere utilizzati per misurare il tempo necessario per il ripristino e il recupero.

8. Comunica alle parti interessate se la convalida dei dati non riesce o se il tempo necessario per il ripristino e il recupero supera l'RTO stabilito per il carico di lavoro. Nell'implementare l'automazione a tale scopo, come in questo lab, è possibile utilizzare servizi come Amazon Simple Notification Service (Amazon SNS) per inviare notifiche push come e-mail o SMS alle parti interessate. I messaggi in questione possono essere pubblicati anche su applicazioni di messaggistica come Amazon Chime, Slack o Microsoft Teams o utilizzati per creare attività come OpsItems mediante AWS Systems Manager OpsCenter.
9. Automatizza questo processo per eseguirlo 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 richiamare periodicamente questo flusso di lavoro di automazione, come mostrato nel diagramma di architettura sottostante. Per ulteriori informazioni, consulta Automate data recovery validation with AWS Backup. Inoltre, questo Well-Architected lab fornisce un'esperienza pratica su come realizzare l'automazione di alcuni dei passaggi qui descritti.

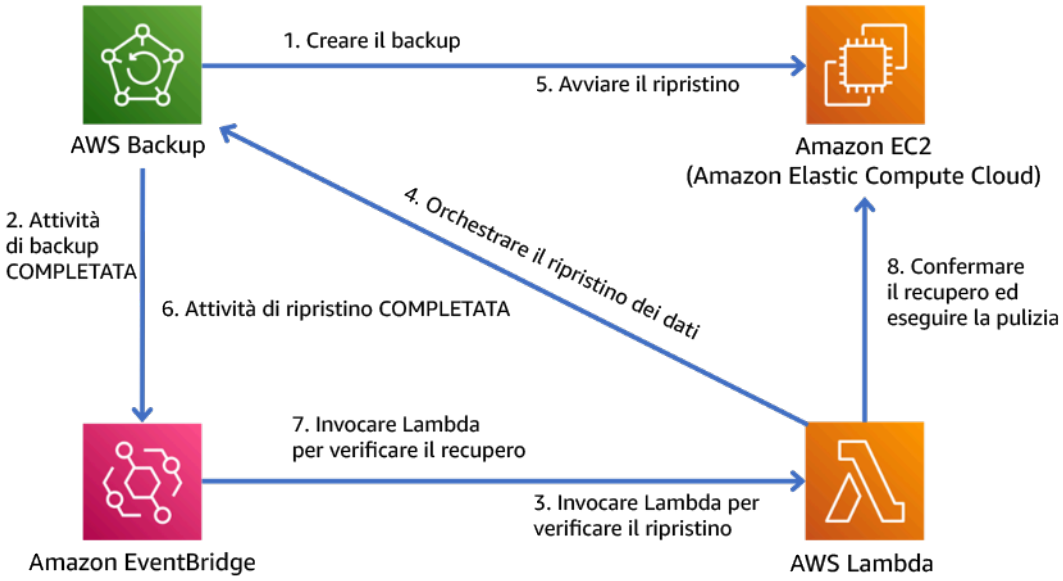


Figura 9. Processo di backup e ripristino automatico

Livello di impegno per il piano di implementazione: da moderato a elevato, in base alla complessità dei criteri di convalida.

Risorse

Documenti correlati:

- Automate data recovery validation with AWS Backup.
- Partner APN: partner per il backup
- Marketplace AWS: prodotti che possono essere utilizzati per il backup
- Creating an EventBridge Rule That Triggers on a Schedule
- Backup e ripristino on demand per DynamoDB
- Che cos'è AWS Backup?
- Che cos'è AWS Step Functions?
- Cos'è AWS Elastic Disaster Recovery?
- AWS Elastic Disaster Recovery

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

L'isolamento dei guasti limita l'impatto di un guasto di un componente o di un sistema entro una determinata barriera. Con un isolamento adeguato, i componenti al di fuori della barriera non subiscono gli effetti del guasto. Utilizzando più barriere per l'isolamento dei guasti, è possibile rendere un carico di lavoro più resiliente ai guasti.

Best practice

- REL10-BP01 Implementazione del carico di lavoro in diversi luoghi
- REL10-BP02 Ripristino automatico dei componenti vincolati a una singola posizione
- REL10-BP03 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.

Un principio fondamentale per la progettazione dei servizi in AWS è quello di evitare singoli punti di errore, inclusa l'infrastruttura fisica sottostante. AWS fornisce risorse e servizi di cloud computing a livello globale in più posizioni geografiche chiamate Regioni. Ogni Regione è fisicamente e logicamente indipendente ed è costituita da tre o più zone di disponibilità (AZ). Le zone di disponibilità

sono geograficamente vicine ma fisicamente separate e isolate. Distribuendo i carichi di lavoro tra le zone di disponibilità e le Regioni, si riducono i rischi legati a minacce quali incendi, inondazioni, disastri meteorologici, terremoti ed errori umani.

Crea una strategia di localizzazione per fornire un'alta disponibilità adeguata ai carichi di lavoro.

Risultato desiderato: i carichi di lavoro di produzione sono distribuiti tra più zone di disponibilità (AZ) o Regioni per ottenere tolleranza ai guasti e alta disponibilità.

Anti-pattern comuni:

- Il carico di lavoro di produzione esiste solo in una singola zona di disponibilità.
- Viene implementata un'architettura multiregionale quando invece un'architettura multi-AZ è in grado di soddisfare i requisiti aziendali.
- Le implementazioni o i dati vengono desincronizzati, con conseguenti deviazioni di configurazione o dati sottoreplicati.
- Non tieni conto delle dipendenze tra i componenti dell'applicazione se i requisiti di resilienza e multi-posizione differiscono tra tali componenti.

Vantaggi dell'adozione di questa best practice:

- Il carico di lavoro è più resiliente in caso di incidenti, come interruzioni di corrente, problemi con i controlli ambientali, disastri naturali, errori dei servizi upstream o problemi di rete che hanno un impatto su un'AZ o su un'intera Regione.
- È possibile accedere a un inventario più ampio di istanze Amazon EC2 e ridurre le probabilità che si verifichino eccezioni InsufficientCapacityException (ICE) quando si avviano tipi specifici di istanze EC2.

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

Guida all'implementazione

Implementa e gestisci tutti i carichi di lavoro di produzione in almeno due zone di disponibilità (AZ) in una Regione.

Utilizzo di più zone di disponibilità

Le zone di disponibilità sono posizioni di hosting delle risorse fisicamente separate l'una dall'altra per evitare guasti correlati dovuti a rischi quali incendi, inondazioni e trombe d'aria. Ogni zona di

disponibilità ha un'infrastruttura fisica indipendente, che include le connessioni alla rete elettrica, le fonti di alimentazione di backup, i servizi meccanici e la connettività di rete. Questa disposizione limita i guasti di uno qualsiasi di questi componenti alla sola zona di disponibilità interessata. Ad esempio, se un incidente a livello di AZ rende non disponibili le istanze EC2 nella zona di disponibilità interessata, è comunque possibile utilizzare le istanze in altre zone di disponibilità.

Nonostante siano fisicamente separate, le zone di disponibilità nella stessa Regione AWS sono sufficientemente vicine da garantire una rete a elevato throughput e bassa latenza (inferiore ai 10 millisecondi). Puoi replicare i dati in modo sincrono tra le zone di disponibilità per la maggior parte dei carichi di lavoro senza influire in modo significativo sull'esperienza dell'utente. Ciò significa che puoi utilizzare le zone di disponibilità in una Regione in una configurazione attiva/attiva o attiva/in standby.

Tutta l'elaborazione associata al carico di lavoro deve essere distribuita tra più zone di disponibilità. Sono incluse le istanze Amazon EC2, le attività AWS Fargate e le funzioni AWS Lambda collegate al VPC. I servizi di elaborazione AWS, compresi EC2 Auto Scaling, Amazon Elastic Container Service (ECS) e Amazon Elastic Kubernetes Service (EKS), offrono la possibilità di avviare e gestire l'elaborazione nelle zone di disponibilità. Configurarli per sostituire automaticamente l'elaborazione, secondo necessità, in una zona di disponibilità diversa per mantenere la disponibilità. Per indirizzare il traffico verso zone di disponibilità integre, posiziona un bilanciatore del carico davanti al computer, ad esempio un Application Load Balancer o un Network Load Balancer. I bilanciatori del carico AWS possono reindirizzare il traffico verso le istanze disponibili in caso di compromissione della zona di disponibilità.

È inoltre necessario replicare i dati per il carico di lavoro e renderli disponibili in più zone di disponibilità. Alcuni servizi di dati gestiti da AWS, come Amazon S3, Amazon Elastic File Service (EFS), Amazon Aurora, Amazon DynamoDB, Amazon Simple Queue Service (SQS) e Flusso di dati Amazon Kinesis replicano i dati in più zone di disponibilità per impostazione predefinita e sono robusti rispetto alla compromissione della zona di disponibilità. Con altri servizi dati gestiti da AWS, come Amazon Relational Database Service (RDS), Amazon Redshift e Amazon ElastiCache, è necessario abilitare la replica multi-AZ. Una volta abilitati, questi servizi rilevano automaticamente la compromissione di una zona di disponibilità, reindirizzano le richieste verso una zona di disponibilità integra e replicano i dati in base alle esigenze dopo il ripristino senza l'intervento del cliente. Per comprendere le funzionalità, i comportamenti e le operazioni multi-AZ di ciascun servizio dati gestito da AWS utilizzato, consulta la guida per l'utente.

Se utilizzi un'archiviazione autogestita, come i volumi Amazon Elastic Block Store (EBS) o l'archiviazione di istanze Amazon EC2, devi gestire autonomamente la replica multi-AZ.

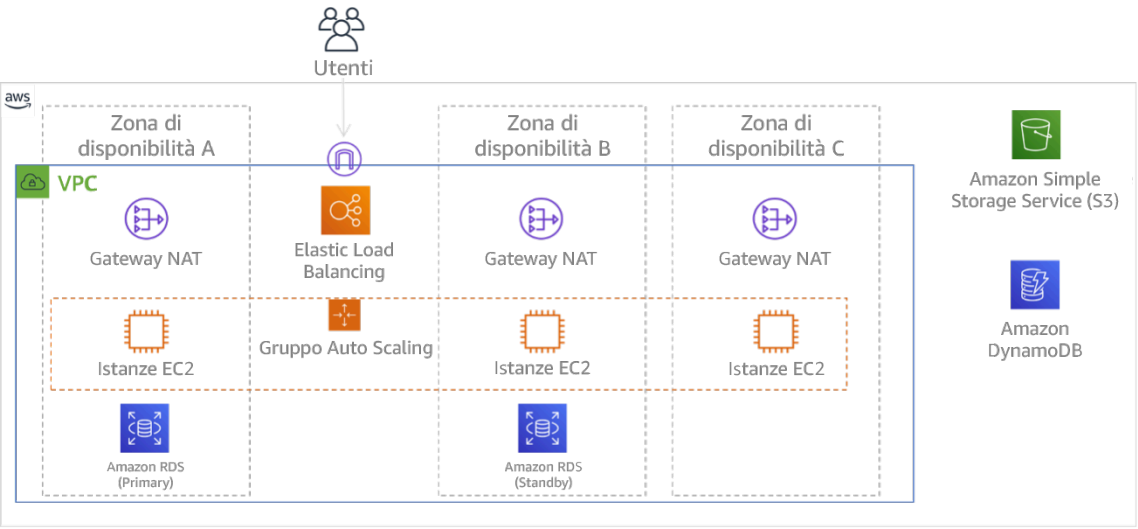


Figura 9: architettura multi-livello distribuita su tre zone di disponibilità. Nota: Amazon S3 e Amazon DynamoDB sono sempre ad AZ multiple automaticamente. L'ELB viene inoltre distribuito in tutte e tre le zone.

Utilizzo di più Regioni AWS

In presenza di carichi di lavoro che richiedono una resilienza estrema (come infrastrutture critiche, applicazioni sanitarie o servizi con requisiti di disponibilità stringenti da parte dei clienti o imposti), potrebbe essere richiesta disponibilità aggiuntiva rispetto a quella che può fornire una singola Regione AWS. In questo caso, è necessario implementare e gestire il carico di lavoro su almeno due Regioni AWS (supponendo che ciò sia consentito dai requisiti di residenza dei dati).

Le Regioni AWS sono situate in diverse aree geografiche del mondo e in più continenti. Le Regioni AWS hanno una separazione fisica e un isolamento ancora maggiori rispetto alle sole zone di disponibilità. I servizi AWS, con poche eccezioni, sfruttano questa struttura per operare in modo completamente indipendente tra le diverse Regioni (noti anche come servizi regionali). Un guasto di un servizio in una Regione AWS, non vi è alcun impatto sul servizio in un'altra Regione.

Quando il carico di lavoro viene gestito in più Regioni, è necessario considerare ulteriori requisiti. Poiché le risorse in Regioni diverse sono separate e indipendenti l'una dall'altra, è necessario duplicare i componenti del carico di lavoro in ciascuna Regione. Questo include l'infrastruttura di base, come i VPC, oltre ai servizi dati e di elaborazione.

NOTA: se prendi in considerazione una progettazione multiregionale, verifica che sia possibile eseguire il carico di lavoro in una singola Regione. Se crei dipendenze tra le Regioni, in cui un componente di una Regione si affida a servizi o componenti di una Regione diversa, il rischio di errore potrebbe aumentare, indebolendo in maniera significativa la propria postura di affidabilità.

Per facilitare le implementazioni multiregionali e mantenere la coerenza, AWS CloudFormation StackSets può replicare l'intera infrastruttura AWS in più Regioni. AWS CloudFormation può anche rilevare deviazioni di configurazione e informare l'utente quando le risorse AWS in una Regione non sono sincronizzate. Molti servizi AWS offrono la replica in più Regioni per le risorse importanti del carico di lavoro. Ad esempio, EC2 Image Builder può pubblicare Amazon Machine Image (AMI) EC2 dopo ogni compilazione su ogni Regione utilizzata. Amazon Elastic Container Registry (ECR) può replicare le immagini del container nelle Regioni selezionate.

È inoltre necessario replicare i dati in ciascuna delle Regioni scelte. Molti servizi dati gestiti da AWS offrono funzionalità di replica multiregionale, tra cui Amazon S3, Amazon DynamoDB, Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon Elasticache e Amazon EFS. Le tabelle globali di Amazon DynamoDB accettano scritture in qualsiasi Regione supportata e replicano i dati tra tutte le altre Regioni configurate. Con altri servizi, è necessario designare una Regione primaria per le scritture, mentre le altre Regioni contengono repliche di sola lettura. Per ogni servizio dati gestito da AWS utilizzato dal carico di lavoro, consulta la relativa guida per l'utente e la guida per gli sviluppatori per comprenderne le funzionalità e i limiti multiregionali. È opportuno prestare particolare attenzione a dove devono essere indirizzate le scritture, alle capacità e alle limitazioni transazionali, a come viene eseguita la replica e a come monitorare la sincronizzazione tra le Regioni.

AWS offre anche la possibilità di instradare il traffico delle richieste verso le implementazioni regionali con grande flessibilità. Ad esempio, puoi configurare i record DNS utilizzando Amazon Route 53 per indirizzare il traffico verso la Regione disponibile più vicina all'utente. In alternativa, puoi configurare i record DNS in una configurazione attiva/in standby, in cui una Regione viene designata come primaria e si ricorre a una replica regionale solo se la Regione primaria perde la propria integrità. Puoi configurare i controlli dell'integrità di Route 53 per rilevare gli endpoint non integri ed eseguire il failover automatico, nonché utilizzare Amazon Application Recovery Controller (ARC) per fornire un controllo di instradamento ad alta disponibilità per reinstradare manualmente il traffico, se necessario.

Anche se scegli di non operare in più Regioni per l'alta disponibilità, considera più Regioni come parte della propria strategia di disaster recovery (DR). Se possibile, replica i componenti e i dati dell'infrastruttura del carico di lavoro in una configurazione warm standby o fiamma pilota in una Regione secondaria. In questa progettazione, si replica l'infrastruttura di base dalla Regione primaria come VPC, gruppi Auto Scaling, orchestratori di container e altri componenti, ma si configurano i componenti di dimensioni variabili nella Regione di standby (come il numero di istanze EC2 e le repliche di database) in modo che siano di dimensioni minimamente utilizzabili. Puoi anche organizzare la replica continua dei dati dalla Regione primaria alla Regione di standby. Se si verifica un incidente, puoi aumentare orizzontalmente, o incrementare, le risorse nella Regione di standby e quindi promuoverla a Regione primaria.

Passaggi dell'implementazione

1. Collabora con le parti interessate aziendali e gli esperti in materia di residenza dei dati per determinare quali Regioni AWS possono essere utilizzate per ospitare le risorse e i dati.
2. Collabora con le parti interessate aziendali e tecniche per 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 multiregionale (se sono consentite più Regioni). L'uso di più Regioni può garantire maggiore disponibilità, ma può comportare complessità e costi aggiuntivi. Nella valutazione, considera i seguenti fattori:
   a. Obiettivi aziendali e requisiti dei clienti: quanto tempo di inattività è consentito nel caso in cui si verifichi un incidente che impatta sul carico di lavoro in una zona di disponibilità o in una Regione? Valuta gli obiettivi dei punti di ripristino come descritto in REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati.
   b. Requisiti per il disaster recovery (DR): contro quale tipo di potenziale disastro desideri assicurarti? Considera la possibilità di perdita di dati o di indisponibilità a lungo termine a livello di diversi ambiti di impatto, da una singola zona di disponibilità a un'intera Regione. Se si replicano i dati e le risorse tra le zone di disponibilità e in una singola zona di disponibilità si verifica un guasto prolungato, il servizio può essere ripristinato in un'altra zona di disponibilità. Se si replicano i dati e le risorse tra Regioni, puoi ripristinare il servizio in un'altra Regione.
3. Distribuisci le risorse di elaborazione in più zone di disponibilità.
   a. Nel VPC, crea più sottoreti in diverse zone di disponibilità. Configura ciascuna di esse in modo che siano sufficientemente grandi da ospitare le risorse necessarie per servire il carico di lavoro, anche durante un incidente. Per ulteriori informazioni consulta REL02-BP03 Verifica che l'allocazione delle sottoreti IP consenta l'espansione e la disponibilità.
   b. Se utilizzi istanze Amazon EC2, utilizza EC2 Auto Scaling per gestire le istanze. Specifica le sottoreti scelte nel passaggio precedente durante la creazione di gruppi Auto Scaling.
   c. Se utilizzi l'elaborazione AWS Fargate per Amazon ECS o Amazon EKS, seleziona le sottoreti che hai scelto nel primo passaggio durante l'operazione di creazione di un servizio ECS, l'avvio di un'attività ECS o la creazione di un profilo Fargate per EKS.
   d. Se utilizzi funzioni AWS Lambda che devono essere eseguite nel VPC, seleziona le sottoreti che hai scelto nel primo passaggio dell'operazione di creazione della funzione Lambda. Per tutte le funzioni che non dispongono di una configurazione VPC, AWS Lambda gestisce automaticamente la disponibilità.
   e. Colloca i direttori del traffico, come i bilanciatori del carico, davanti alle risorse di elaborazione. Se il bilanciamento del carico tra zone è abilitato, AWS Application Load Balancer e Network

Load Balancer rilevano quando destinazioni come istanze e container EC2 non sono raggiungibili a causa della compromissione della zona di disponibilità e reinstradano il traffico verso destinazioni in zone di disponibilità integre. Se il bilanciamento del carico tra zone è disabilitato, utilizza Amazon Application Recovery Controller (ARC) per fornire funzionalità di spostamento zonale. Se utilizzi un bilanciatore del carico di terze parti o hai implementato bilanciatori del carico personalizzati, configurali con più front-end in diverse zone di disponibilità.

4. Replica i dati del carico di lavoro in più zone di disponibilità.
   a. Se utilizzi un servizio dati gestito da AWS come Amazon RDS, Amazon ElastiCache o Amazon FSx, consulta la relativa guida per l'utente per comprendere le relative funzionalità di replica dei dati e di resilienza. Se necessario, abilita la replica e il failover tra AZ.
   b. Se utilizzi servizi di archiviazione gestiti da AWS come Amazon S3, Amazon EFS e Amazon FSx, evita di utilizzare configurazioni Single-AZ o One Zone per i dati che richiedono un'elevata durabilità. Utilizza una configurazione multi-AZ per questi servizi. Consulta la guida per l'utente del rispettivo servizio per determinare se la replica multi-AZ è abilitata per impostazione predefinita o se è necessario abilitarla.
   c. Se esegui un database, una coda o un altro servizio di archiviazione autogestito, organizza la replica multi-AZ in base alle istruzioni o alle best practice dell'applicazione. Informati sulle procedure di failover della tua applicazione.
5. Configura il servizio DNS per rilevare compromissione dell'AZ e reinstrada il traffico verso una zona di disponibilità integra. Se utilizzato in combinazione con Elastic Load Balancer, Amazon Route 53 può eseguire questa operazione automaticamente. Route 53 può anche essere configurato con record di failover che utilizzano i controlli dell'integrità per rispondere alle query con soli indirizzi IP integri. Per tutti i record DNS utilizzati per il failover, specifica un valore TTL (time to live) breve (ad esempio, 60 secondi o meno) per evitare che la memorizzazione nella cache dei record impedisca il ripristino (i record alias di Route 53 forniscono i TTL appropriati).

Passaggi aggiuntivi quando si utilizzano più Regioni AWS

1. Replica tutto il sistema operativo (OS) e il codice dell'applicazione utilizzati dal carico di lavoro nelle Regioni selezionate. Se necessario, replica le Amazon Machine Image (AMI) utilizzate dalle istanze EC2 utilizzando soluzioni come Amazon EC2 Image Builder. Replica le immagini di container archiviate nei registri utilizzando soluzioni come la replica tra Regioni di Amazon ECR. Abilita la replica regionale per tutti i bucket Amazon S3 utilizzati per archiviare le risorse dell'applicazione.

2. Distribuisci le risorse di elaborazione e i metadati di configurazione (come i parametri archiviati in AWS Systems Manager Parameter Store) in più Regioni. Utilizza le stesse procedure descritte nei passaggi precedenti, ma replica la configurazione per ogni Regione utilizzata per il carico di lavoro. Utilizza soluzioni infrastructure as code, ad esempio AWS CloudFormation, per riprodurre in modo uniforme le configurazioni tra le Regioni. Se utilizzi una Regione secondaria in una configurazione fiamma pilota per il disaster recovery, puoi ridurre il numero di risorse di elaborazione a un valore minimo per risparmiare sui costi, con un corrispondente aumento del tempo di ripristino.
3. Replica i dati dalla Regione primaria alle Regioni secondarie.
    a. Le tabelle globali di Amazon DynamoDB forniscono repliche globali dei dati in cui è possibile scrivere da qualsiasi Regione supportata. Con altri servizi dati gestiti da AWS, come Amazon RDS, Amazon Aurora e Amazon Elasticache, si designano una Regione primaria (lettura/scrittura) e Regioni di replica (sola lettura). Per informazioni dettagliate sulla replica regionale, consulta le guide per l'utente e per gli sviluppatori dei rispettivi servizi.
    b. Se esegui un database autogestito, organizza la replica in più Regioni in base alle istruzioni o alle best practice dell'applicazione. Informati sulle procedure di failover della tua applicazione.
    c. Se il carico di lavoro utilizza AWS EventBridge, potrebbe essere necessario inoltrare eventi selezionati dalla Regione primaria alle Regioni secondarie. A tal fine, specifica i bus di eventi nelle Regioni secondarie come destinazioni per gli eventi corrispondenti nella Regione primaria.
4. Considera se e in che misura usare chiavi di crittografia identiche tra le varie Regioni. Un approccio tipico, che consente di bilanciare sicurezza e facilità d'uso, consiste nell'utilizzare chiavi con ambito di Regione per i dati e l'autenticazione a livello di Regione e utilizzare chiavi con ambito globale per la crittografia dei dati replicati tra le diverse Regioni. AWS Key Management Service (KMS) supporta chiavi multiregionali per distribuire e proteggere in modo sicuro le chiavi condivise tra le Regioni.
5. Prendi in considerazione AWS Global Accelerator per migliorare la disponibilità dell'applicazione indirizzando il traffico verso Regioni che contengono endpoint integri.

Risorse

Best practice correlate:

- REL02-BP03 Verifica che l'allocazione delle sottoreti IP consenta l'espansione e la disponibilità
- REL11-BP05 Utilizzo della stabilità statica per evitare un comportamento bimodale
- REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati

Documenti correlati:

- Infrastruttura globale di AWS
- White paper: AWS Fault Isolation Boundaries
- Resilience in Amazon EC2 Auto Scaling
- Amazon EC2 Auto Scaling: Example: Distribute instances across Availability Zones
- How EC2 Image Builder works
- How Amazon ECS places tasks on container instances (includes Fargate)
- Resilienza in AWS Lambda
- Amazon S3: Replicating objects overview
- Private image replication in Amazon ECR
- Global Tables: Multi-Region Replication with DynamoDB
- Amazon Elasticache for Redis OSS: Replication across Regioni AWS using global datastores
- Resilience in Amazon RDS
- Using Amazon Aurora global databases
- AWS Global Accelerator Developer Guide
- Multi-Region keys in AWS KMS
- Amazon Route 53: Configuring DNS failover
- Amazon Application Recovery Controller (ARC) Developer Guide
- Sending and receiving Amazon EventBridge events between Regioni AWS
- Creating a Multi-Region Application with AWS Services blog series
- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud
- Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby

Video correlati:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure

REL10-BP02 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-premises, devi rendere possibile la ricostruzione completa del carico di lavoro in base agli obiettivi di ripristino definiti.

Livello di rischio associato se questa best practice non fosse adottata: 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 perché l'esecuzione di 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. Utilizzando il file system EMR (EMRFS), i dati in EMR possono essere memorizzati in Amazon S3, che a sua volta può essere replicato su 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. Viene effettuato il provisioning di tutti i nodi del cluster nella stessa zona.

Per carichi di lavoro basati su server stateful implementati in un data center on-premises, puoi usare AWS Elastic Disaster Recovery per proteggerli in AWS. Se il carico di lavoro è già ospitato in AWS, Elastic Disaster Recovery ti consente di proteggerlo in una zona di disponibilità o regione alternativa. Elastic Disaster Recovery sfrutta la replica a livello di blocco continua in un'area di gestione temporanea leggera per fornire il ripristino rapido e affidabile di applicazioni on-premises e basate sul cloud.

Passaggi dell'implementazione

1. Implementa l'autoriparazione. Implementa istanze o container utilizzando, quando possibile, il dimensionamento automatico. Se non è possibile utilizzare il dimensionamento 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.

- Utilizza gruppi Amazon EC2 Auto Scaling per carichi di lavoro di container e istanze che non richiedono un indirizzo IP di una singola istanza, un indirizzo IP privato, un indirizzo IP elastico o metadati di istanza.
  - 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.
- Utilizza il ripristino delle istanze Amazon EC2 per carichi di lavoro che richiedono un indirizzo ID di una singola istanza, un indirizzo IP privato, un indirizzo IP elastico e metadati di istanza.
  - Il ripristino automatico invierà avvisi sullo stato del ripristino a un argomento SNS quando viene rilevato l'errore dell'istanza.
- Utilizza gli eventi del ciclo di vita di istanze Amazon EC2 o gli eventi Amazon ECS per automatizzare l'autoriparazione dove non è possibile utilizzare il dimensionamento automatico o il ripristino EC2.
  - Utilizza gli eventi per richiamare l'automazione che riparerà il tuo componente secondo la logica di processo richiesta.
- Utilizza AWS Elastic Disaster Recovery per proteggere i carichi di lavoro stateful limitati a una singola posizione.

Risorse

Documenti correlati:

- Amazon ECS events
- Amazon EC2 Auto Scaling lifecycle hooks
- Recupero di un'istanza
- Ridimensionamento automatico del servizio
- What Is Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP03 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 utilizza più istanze isolate di un carico di lavoro, ciascuna delle quali 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: limitazione alla cella stessa di molti tipi comuni di errori, a garanzia di un ulteriore isolamento dei guasti, grazie alle architetture basate su celle. Questi limiti relativi agli errori possono garantire resilienza in caso di determinati tipi di errori, altrimenti difficili da contenere, come implementazioni di codice non riuscite o richieste danneggiate o che richiamano una modalità di errore specifica (nota anche come richieste poison pill).

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

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.

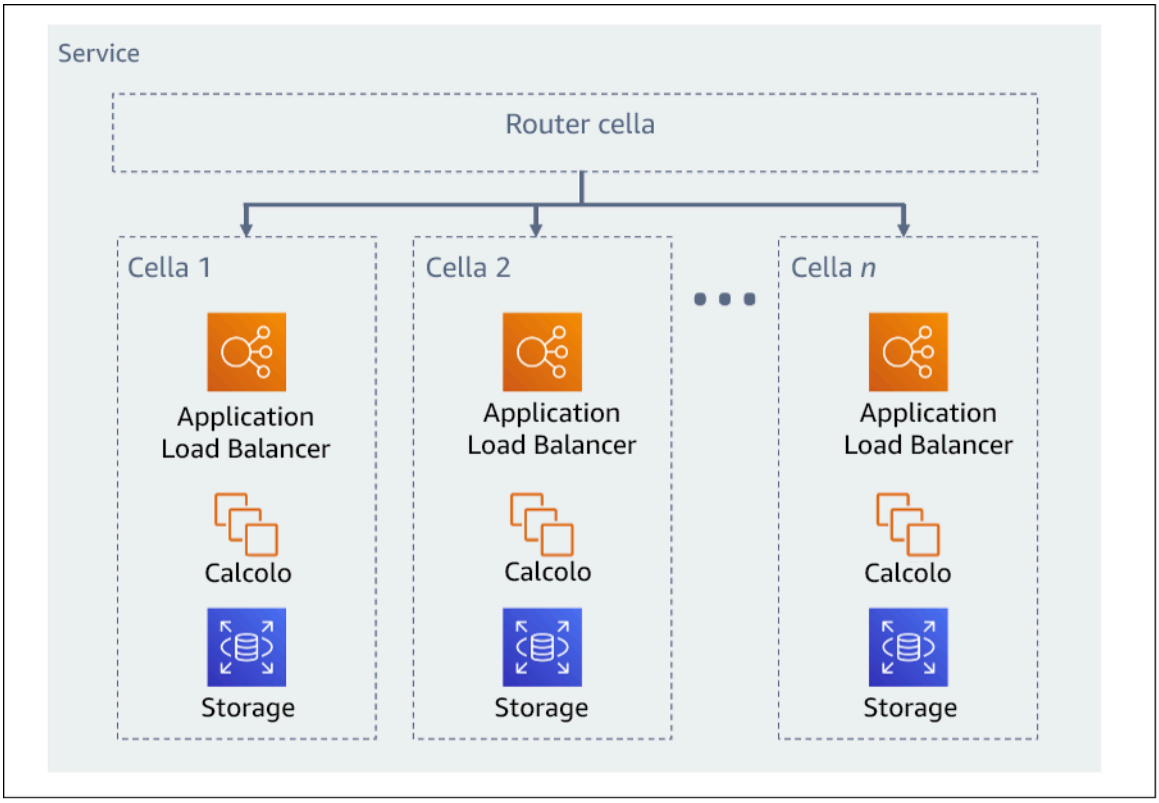


Figura 11: architettura basata su celle

Passaggi dell'implementazione

Nel progettare un'architettura basata su celle, devi tenere conto di diversi aspetti della progettazione:

1. Chiave di partizione: presta particolare attenzione alla scelta della chiave di partizione.
    - Questa 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 sono `customer ID` o `resource ID`.

- 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 una singola 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: il livello di instradamento è un componente condiviso tra celle, pertanto non può basarsi sulla stessa strategia di compartimentazione delle 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 illustrato da Colm MacCárthaigh in Reliability, constant work, and a good cup of coffee, progettazioni semplici e schemi di lavoro costanti si traducono in sistemi affidabili e nella riduzione dell'antifragilità.
4. Dimensione delle celle: le celle devono avere una dimensione massima che non deve essere superata
   - La dimensione massima va 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 Load Testa il tuo 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 o multi-regione: si consiglia di utilizzare più livelli di resilienza per proteggersi da diversi domini di errore.

- 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 dettagli, consulta REL10-BP01 Implementazione del carico di lavoro in diversi luoghi.

6. Implementazione del codice: è preferibile una strategia di implementazione del codice scaglionata rispetto all'implementazione simultanea di modifiche al codice in tutte le celle.
    - 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 distribuzioni pratiche e sicure.

Risorse

Best practice correlate:

- REL07-BP04 Load Testa il tuo carico di lavoro
- REL10-BP01 Implementazione del carico di lavoro in diversi luoghi

Documenti correlati:

- Reliability, constant work, and a good cup of coffee
- AWS and Compartmentalization
- Isolamento del carico di lavoro utilizzando lo sharding casuale
- Automatizzazione di distribuzioni pratiche e sicure

Video correlati:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience

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 Affidati al piano dati e non al piano di controllo durante il ripristino
- REL11-BP05 Usa la stabilità statica per prevenire il 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 gli accordi sul livello di servizio (SLA) 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 del tempo di ripristino (RTO)
- 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: elevato

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 dei guasti nell'area grigia. Per ulteriori informazioni sui guasti nell'area grigia, consulta Gray failures nel whitepaper 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 del tempo di ripristino (RTO)

- Configura il monitoraggio dettagliato per componenti e servizi gestiti.
  - Determina se è necessario un monitoraggio dettagliato per le istanze EC2 e Auto Scaling. Il monitoraggio dettagliato fornisce metriche a intervalli di un minuto, mentre il monitoraggio predefinito fornisce metriche a intervalli di cinque minuti.
  - Determina se è necessario un monitoraggio avanzato per RDS. 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 bilanciatori del carico.
  - Determina i requisiti di monitoraggio dei componenti di archiviazione per Amazon S3, Amazon FSx, Amazon EFS e Amazon EBS.
- Crea parametri personalizzati per misurare indicatori chiave di prestazione (KPI) aziendali. I carichi di lavoro implementano funzioni aziendali fondamentali, che devono essere utilizzate come KPI che aiutano a identificare quando si verifica un problema indiretto.
- Monitora la presenza di errori nell'esperienza utente tramite le canary degli utenti Il test sintetico delle transazioni (noto anche come "test canary", ma da non confondere con le distribuzioni canary) in grado di eseguire e simulare il comportamento dei clienti è uno dei processi di test più importanti. Esegui questi test costantemente sugli endpoint del carico di lavoro da diverse posizioni remote.
- Crea parametri personalizzati che monitorino l'esperienza dell'utente. Dotare l'esperienza del cliente di strumenti consente di determinare quando essa peggiora.
- Imposta gli allarmi per rilevare quando una qualsiasi parte del carico di lavoro non funziona correttamente e per indicare quando effettuare il dimensionamento automatico delle risorse. È possibile mostrare visivamente gli avvisi sui pannelli di controllo, inviarli tramite Amazon SNS o e-mail e utilizzarli con Auto Scaling per aumentare o ridurre le risorse del carico di lavoro.
- Crea pannelli di controllo per visualizzare i parametri. Utilizza i pannelli di controllo 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.
- Crea sistemi di monitoraggio (utilizzando CloudWatch o X-Ray), pannelli di controllo e raccolta dati in una regione e in un account separati.
- Con AWS Health si ricevono le informazioni sul degrado delle prestazioni del servizio. Si creano notifiche di eventi AWS Health personalizzati per i canali e-mail e chat con Notifiche all'utente AWS

e si usano gli strumenti di monitoraggio e avviso con Amazon EventBridge per l'integrazione a livello di codice.

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 canary dell'utente
- Abilitare o disabilitare il monitoraggio dettagliato della propria istanza
- Monitoraggio avanzato
- Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch
- Publishing Custom Metrics
- Using Amazon CloudWatch Alarms
- Using CloudWatch Dashboards
- Using Cross Region Cross Account CloudWatch Dashboards
- Uso del tracciamento X-Ray tra più regioni e account
- Understanding availability

Video correlati:

- Mitigating gray failures

Esempi correlati:

- 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. In questo modo, 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 garantiscano 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: elevato

Guida all'implementazione

I servizi AWS, come Elastic Load Balancing e Amazon EC2 Auto Scaling, consentono di distribuire il carico tra risorse e zone di disponibilità. In questo modo, 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, Sistema di controllo Amazon per il rispristino di applicazioni (ARC), Amazon CloudFront e AWS Global Accelerator consentono di instradare il traffico nelle Regioni AWS.

I 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 sono implementati in automatico 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, una delle opzioni di configurazione è Multi-AZ. AWS può indirizzare il traffico verso l'istanza integra in caso di

avvio del failover. Questa azione di failover può essere intrapresa direttamente da AWS o su richiesta del cliente.

Per le istanze Amazon EC2, Amazon Redshift, le attività Amazon ECS o i pod Amazon EKS, sei tu a scegliere le zone di disponibilità in cui effettuare l'implementazione. Per alcuni progetti, Elastic Load Balancing fornisce la soluzione per rilevare le istanze in zone non integre e instradare il traffico verso quelle integre. Elastic Load Balancing può inoltre instradare il traffico verso componenti nel tuo data center on-premises.

Per il failover del traffico multi-regione, il reindirizzamento può sfruttare Amazon Route 53, Sistema di controllo Amazon per il ripristino di applicazioni, AWS Global Accelerator, Route 53, DNS privato per VPC o CloudFront per fornire una modalità di definizione dei domini Internet e assegnare policy di instradamento, 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 propria 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 il modello infrastructure as code (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:

- Impostazione di obiettivi RTO e RPO
- Failover utilizzando il routing ponderato Route 53
- Disaster Recovery with Amazon Application Recovery Controller
- EC2 with autoscaling
- EC2 Deployments - Multi-AZ
- ECS Deployments - Multi-AZ
- Switch traffic using Amazon Application Recovery Controller
- 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
- Guidance for Cross Region Failover and Graceful Failback on AWS
- Failover using multi-region global accelerator

- Failover with DRS

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 jitter e un backoff 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.

- Implementazione 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: elevato

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 (control-plane) Kubernetes.

I modelli di progettazione a cui si accede tramite un bilanciatore 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. Ecco alcuni esempi di servizi: Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK e il Servizio OpenSearch di Amazon 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 modifiche 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 tuo carico di lavoro. Amazon EC2 Auto Scaling può essere configurato per verificare lo stato dell'istanza 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 avvia un'istanza 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 i gruppi Auto Scaling per implementare 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 menzionate in precedenza, utilizza il bilanciamento del carico e scegli il tipo di bilanciatore del carico adeguato.
- Prendi in considerazione la riparazione per Amazon RDS Utilizzando le istanze di standby, puoi configurare il failover automatico sulle stesse. 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 sulle istanze EC2 che includono applicazioni distribuite non implementabili in più posizioni e possono tollerare il riavvio in caso di guasti. 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. Vengono conservati i metadati dell'istanza e gli indirizzi IP associati, nonché i volumi EBS e i punti di montaggio su Amazon Elastic File System o file system per Lustre e Windows. Grazie a 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 oppure quando il ripristino automatico non riesce. 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 utilizzato per monitorare e filtrare eventi come allarmi CloudWatch o modifiche 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:

- Funzionamento di AWS Auto Scaling
- Ripristino automatico di 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: utilizzo della riparazione automatica per sostituire le istanze con errore
- Che cos'è AWS Step Functions?
- Che cos'è AWS Lambda?
- What Is Amazon EventBridge?
- Using Amazon CloudWatch Alarms
- Amazon RDS Failover
- SSM - Systems Manager Automation
- Best practice per architetture resilienti

Video correlati:

- Automatically Provision and Scale OpenSearch Service
- Amazon RDS Failover Automatically

Esempi correlati:

- Workshop su Amazon RDS Failover

Strumenti correlati:

- CloudWatch
- CloudWatch X-Ray

REL11-BP04 Affidati al piano dati e non al piano di controllo durante il ripristino

I piani di controllo forniscono le risorse amministrative APIs utilizzate per creare, leggere e descrivere, aggiornare, eliminare ed elencare (CRUDL) le risorse, mentre i piani dati gestiscono il traffico di day-to-day 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 (control-plane) 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 (control-plane): 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 (control-plane) 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 (control-plane) 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 DNS dei record per reindirizzare il traffico.
- Dipendenza dalle operazioni di dimensionamento del piano di controllo (control-plane) per sostituire i componenti danneggiati a causa di un provisioning delle risorse insufficiente.
- Affidarsi ad azioni estese, multiservizio e API multicontrollo sul piano 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 degrado del servizio, i piani di controllo (control-plane) sono interessati. La dipendenza dall'uso estensivo del piano di controllo per la riparazione può aumentare il tempo di ripristino () e il tempo medio di ripristino (RTO). MTTR

Guida all'implementazione

Per limitare le azioni del piano dati, esegui una valutazione del servizio determinare le azioni necessarie per ripristinarlo.

Sfrutta Amazon Application Recovery Controller per spostare il DNS traffico. Queste funzionalità monitorano continuamente la capacità dell'applicazione di ripristinarsi in caso di guasti e consentono di controllare il ripristino delle applicazioni su più Regioni AWS zone di disponibilità e in locale.

Le policy di instradamento di Route 53 utilizzano il piano di controllo (control-plane), quindi non fare affidamento su di esso per il ripristino. I piani dati Route 53 rispondono alle DNS domande ed eseguono e valutano i controlli di integrità. Sono distribuiti a livello globale e progettati per un accordo sul livello di servizio con disponibilità del 100% (SLA).

La gestione APIs e le console di Route 53 in cui è possibile creare, aggiornare ed eliminare le risorse Route 53 funzionano su piani di controllo progettati per dare priorità alla forte coerenza e alla durabilità necessarie durante la gestione. DNS A tal fine, i piani di controllo (control-plane) sono situati in un'unica regione: Stati Uniti orientali (Virginia settentrionale). Sebbene entrambi i sistemi siano progettati per essere molto affidabili, i piani di controllo non sono inclusi nel. SLA Possono verificarsi eventi rari in cui la progettazione resiliente del piano dati consente di mantenere la disponibilità mentre i piani di controllo (control-plane) non lo fanno. Per i meccanismi di ripristino di emergenza e failover, utilizzare le funzioni del piano dati per garantire la migliore affidabilità possibile.

Progetta la tua infrastruttura di elaborazione in modo che sia staticamente stabile per evitare di utilizzare il piano di controllo durante un incidente. Ad esempio, se utilizzi EC2 istanze Amazon, evita di effettuare il provisioning di nuove istanze manualmente o di chiedere agli Auto Scaling Groups di aggiungere istanze in risposta. Per ottenere i massimi livelli di resilienza, è necessario fornire una capacità sufficiente nel cluster utilizzato per il failover. Se questa soglia di capacità deve essere limitata, imposta dei limiti sull'intero end-to-end sistema per limitare in modo sicuro il traffico totale che raggiunge il set limitato di risorse.

Per servizi come Amazon DynamoDB, API Amazon Gateway, load balancer e serverless, l'utilizzo di AWS Lambda tali servizi sfrutta il piano dati. Tuttavia, la creazione di nuove funzioni, load balancer, API gateway o tabelle DynamoDB è un'azione del piano di controllo e deve essere completata prima del degrado come preparazione a un evento e alla ripetizione delle azioni di failover. Per AmazonRDS, le azioni del piano dati consentono l'accesso ai dati.

Per ulteriori informazioni sui piani dati, sui piani di controllo e su come AWS crea servizi per soddisfare gli obiettivi di alta disponibilità, consulta Stabilità statica tramite zone di disponibilità.

Capire quali operazioni sono sul piano dati e quali sul piano di controllo (control-plane).

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 (control-plane).

Prendi in considerazione la possibilità di modificare l'azione di controllo in un'azione del piano dati:

- Auto Scaling (piano di controllo) su risorse EC2 Amazon prescalate (piano dati)
- Scalabilità delle EC2 istanze Amazon (piano di controllo) alla AWS Lambda scalabilità (piano dati)
- Valuta qualsiasi progetto utilizzando Kubernetes e considerando la natura delle azioni del piano di controllo (control-plane). 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 (control-plane).

Prendi in considerazione approcci alternativi che consentano alle azioni del piano dati di incidere sulla stessa correzione.

- Route 53 Record change (piano di controllo) o Amazon Application Recovery Controller (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 (control-plane) e del piano dati in una regione non interessata dal problema.

- Amazon EC2 Auto Scaling o Amazon EKS in una regione principale rispetto ad Amazon Auto EC2 Scaling o EKS Amazon 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 (control-plane))

Risorse

Best practice correlate:

- Definizione di disponibilità
- REL11-BP01 Monitora tutti i componenti del carico di lavoro per rilevare i guasti

Documenti correlati:

- APNPartner: partner che possono contribuire all'automazione della tolleranza ai guasti
- Marketplace AWS: prodotti utilizzabili per la tolleranza ai guasti
- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
- Amazon API DynamoDB (piano di controllo e piano dati)
- AWS Lambda Esecuzioni (suddivise in piano di controllo e piano dati)
- AWS Elemental MediaStore Piano dati
- Creazione di applicazioni altamente resilienti utilizzando Amazon Application Recovery Controller, parte 1: stack per regione singola
- Creazione di applicazioni altamente resilienti utilizzando Amazon Application Recovery Controller, parte 2: stack multiregionale
- Creating Disaster Recovery Mechanisms Using Amazon Route 53
- Cos'è Amazon Application Recovery Controller
- Piano di controllo (control-plane) e piano dati di Kubernetes

Video correlati:

- Back to Basics - Using Static Stability
- Creazione di carichi di lavoro resilienti su più siti utilizzando servizi globali AWS

Esempi correlati:

- Presentazione di Amazon Application Recovery Controller
- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
- Creazione di applicazioni altamente resilienti utilizzando Amazon Application Recovery Controller, parte 1: stack per regione singola

- Creazione di applicazioni altamente resilienti utilizzando Amazon Application Recovery Controller, parte 2: stack multiregionale
- Stabilità statica con le zone di disponibilità

Strumenti correlati:

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 Usa la stabilità statica per prevenire il 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 allocate 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:

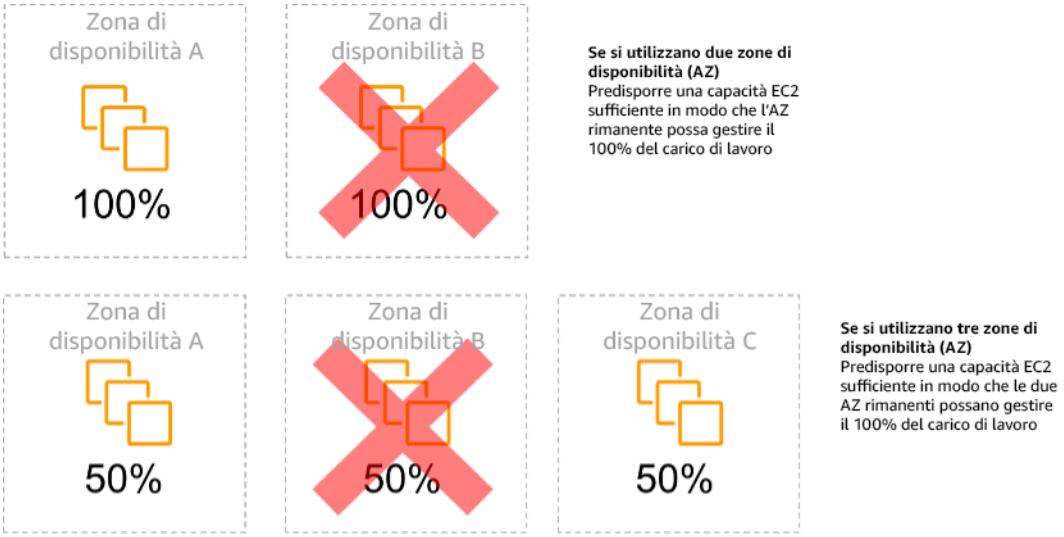
- Supporre che le risorse possano sempre essere allocate indipendentemente dall'ambito del guasto.
- Tentare di acquisire risorse in modo dinamico durante un guasto.
- Non rendere disponibili risorse adeguate tra zone o regioni diverse fino a quando non si verifica un guasto.
- Considerare i progetti 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 si verifica quando EC2 progetti Amazon stabili forniscono un numero sufficiente di istanze in ciascuna zona di disponibilità per gestire il carico di lavoro in caso di rimozione di una zona di disponibilità. Elastic Load Balancing o Amazon Route 53 controllano lo stato in modo da trasferire il carico dalle istanze danneggiate. Dopo lo spostamento del traffico, utilizzalo per sostituire in modo asincrono AWS Auto Scaling le istanze dalla zona in cui si è verificato l'errore e avviarle nelle zone funzionanti. La stabilità statica per l'implementazione dell'elaborazione (come EC2 istanze o contenitori) garantisce la massima affidabilità.



Stabilità statica delle EC2 istanze nelle 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. Le architetture di stabilità statica si applicano a una varietà di architetture, tra cui istanze di calcolo

distribuite tra zone di disponibilità, progetti di repliche di lettura del database, progetti di cluster Kubernetes (AmazonEKS) 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 non va a buon fine, 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 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:

- Elaborazione: AmazonEC2, EKS -EC2, ECS -EC2, EMR - EC2
- Database: Amazon Redshift, AmazonRDS, Amazon Aurora
- Archiviazione: Amazon S3 (zona singola), Amazon EFS (supporti), Amazon FSx (supporti)
- Bilanciatori del carico: in base a determinati progetti

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:

  - Routing tra regioni DNS
  - MRAPRouting Amazon S3 MultiRegion
  - AWS Global Accelerator
  - Controller di ripristino delle applicazioni Amazon
- 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. In caso di utilizzo della 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 bilanciatori del carico sono a volte configurati in modo errato o sono progettati per servire una zona di disponibilità specifica. In questo caso, il design staticamente stabile potrebbe consistere nel distribuire un carico di lavoro su più livelli AZs in un progetto più complesso. Il progetto 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 Monitora tutti i componenti del carico di lavoro per rilevare i guasti
- REL11-BP04 Fai affidamento sul piano dati e non sul 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
- Limiti di isolamento dei guasti
- Stabilità statica con le zone di disponibilità
- Multi-zona RDS
- Minimizing Dependencies in a Disaster Recovery Plan
- Routing interregionale DNS

- MRAPRouting Amazon S3 MultiRegion
- AWS Global Accelerator
- Controller di ripristino delle applicazioni Amazon
- Amazon S3 a zona singola
- Bilanciamento del carico su più zone

Video correlati:

- Stabilità statica in AWS: AWS re:Invent 2019: Presentazione di The Amazon Builders' Library () DOP328

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 allarmi.
- Invio di allarmi non utilizzabili.
- Impostazione di soglie di allarme troppo alte (troppo sensibili) o troppo basse (troppo poco sensibili).

- Mancato invio di allarmi per dipendenze esterne.
- Mancata presa in considerazione dei 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 ALARM` o `OK`) e i dettagli di ciò che lo ha attivato. In molti casi, è necessario rilevare un evento di allarme e inviare una notifica tramite e-mail. Questo è un esempio di operazione 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 eventi imprevisti in sistemi di monitoraggio degli eventi imprevisti 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.
- Gli allarmi infrastrutturali indicano quando scalare le risorse. È possibile mostrare visivamente gli allarmi sui pannelli di controllo, inviarli tramite Amazon SNS o e-mail e utilizzarli con Auto Scaling per ridurre orizzontalmente o aumentare orizzontalmente i carichi di lavoro.
- È possibile creare semplici allarmi statistici 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 un'API Amazon SNS per inviare notifiche e collegare qualsiasi automazione ai fini della riparazione o della comunicazione.
- Con AWS Health si ricevono le informazioni sul degrado delle prestazioni del servizio. Si creano notifiche di eventi AWS Health personalizzati per i canali e-mail e chat con Notifiche all'utente AWS e si usano gli strumenti di monitoraggio e avviso con Amazon EventBridge per l'integrazione a livello di codice.

Risorse

Best practice Well-Architected correlate:

- Definizione di disponibilità

Documenti correlati:

- Creating a CloudWatch Alarm Based on a Static Threshold
- What Is Amazon EventBridge?
- What is Amazon Simple Notification Service?
- Publishing Custom Metrics
- Using Amazon CloudWatch Alarms
- Configurazione degli allarmi compositi di CloudWatch
- What's new 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 gli accordi sul livello di servizio (SLA) per i tempi di attività

Progetta il tuo prodotto per soddisfare gli accordi sul livello di servizio (SLA) sul tempo di attività e sugli obiettivi di disponibilità. 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: ogni applicazione presenta un obiettivo definito in termini di disponibilità e uno SLA per le metriche delle prestazioni, monitorabili e gestibili per soddisfare i risultati aziendali.

Anti-pattern comuni:

- Progettazione e implementazione di carichi di lavoro senza predisporre alcun SLA.
- Impostazione di metriche elevate per lo SLA senza fondamento logico o requisiti aziendali.
- Impostazione di SLA senza tenere conto delle dipendenze e dei relativi SLA sottostanti.
- Progettazione delle applicazioni senza tenere conto del Modello di responsabilità condivisa per la resilienza.

Vantaggi dell'adozione di questa best practice: soddisfare gli obiettivi aziendali e le aspettative dei clienti grazie alla progettazione di applicazioni in base a obiettivi chiave in termini di resilienza. 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: medio

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 SLA. In generale, seleziona dipendenze con obiettivi di disponibilità uguali o maggiori rispetto agli obiettivi del carico di lavoro.
- Prendi in considerazione progettazioni con accoppiamento debole 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.
- L'osservabilità 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 i 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 disponibilità e la coerenza 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 alle parti interessate le metriche per lo SLA in base all'architettura del carico di lavoro. Tieni conto degli SLA di tutte le dipendenze usate dal carico di lavoro.
- Una volta definiti gli obiettivi dello SLA, ottimizza l'architettura in modo da soddisfarlo.
- Una volta impostata una progettazione che soddisfa lo SLA, implementa modifiche operative, automazione dei processi e runbook anch'essi incentrati sulla riduzione dell'MTTD e dell'MTTR.
- Dopo aver implementato lo SLA, devi monitorarlo e documentarlo.

Risorse

Best practice correlate:

- REL03-BP01 Scegli come segmentare il tuo 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-BP04 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:

- Availability with redundancy
- Pilastro dell'affidabilità: disponibilità
- Measuring availability
- Limiti di isolamento dei guasti di AWS
- Modello di responsabilità condivisa per la resilienza
- Stabilità statica con le zone di disponibilità
- Contratti sul livello di servizio (SLA) AWS
- Guidance for Cell-based Architecture on AWS
- AWS infrastructure

- Whitepaper Advanced Multi-AZ Resiliance Patterns

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 comprometterne 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 di scalabilità e prestazioni
- REL12-BP04 Test della resilienza tramite l'utilizzo dell'ingegneria del caos
- REL12-BP05 Esecuzione regolare di GameDay

REL12-BP01 Utilizzo dei playbook per analizzare gli errori

Consenti 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. Se 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 dei 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: elevato

Guida all'implementazione

- Utilizza playbook per identificare i problemi. I playbook sono processi documentati per eseguire indagini sui problemi. Promuovi 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 richiamate o eseguite nell'ambito delle attività dei playbook oppure possono richiedere l'esecuzione di un playbook in risposta agli eventi identificati.
    - Automatizza i playbook operativi con AWS Systems Manager

- AWS Systems Manager Run Command
- AWS Systems Manager Automation
- Che cos'è AWS Lambda?
- What Is Amazon EventBridge?
- Using Amazon CloudWatch Alarms

Risorse

Documenti correlati:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- Automatizza i playbook operativi con AWS Systems Manager
- Using Amazon CloudWatch Alarms
- Utilizzo di Canary (Amazon CloudWatch Synthetics)
- What Is Amazon EventBridge?
- Che cos'è AWS Lambda?

Esempi correlati:

- 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 incidenti. 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 dispongono di un approccio coerente e concordato per la 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 gli esiti 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 permette loro di implementare la mitigazione o il ripristino automatico prima che si verifichi un incidente.

Livello di rischio associato se questa best practice non fosse adottata: 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. Nomina 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. Utilizza tecniche come i Cinque Perché per analizzare in modo approfondito 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 gli esiti 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:
  - Che cos'è successo?
  - Quale impatto ha avuto su clienti e attività?
  - Qual è stata la causa principale?
  - Di quali dati disponi a supporto di ciò?
    - 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. Le decisioni aziendali possono stabilire le priorità di progettazione. 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?
    - Elementi d'azione

    - Voci correlate
- 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 gli esiti chiave in un sistema di gestione delle conoscenze e valuta eventuali modelli da aggiungere alle linee guide per gli sviluppatori o alle liste di controllo usate nella fase di pre-implementazione.

Risorse

Documenti correlati:

- Why you should develop a correction of error (COE)

Video correlati:

- Amazon's approach to failing successfully
- AWS re:Invent 2021 - Amazon Builders' Library: Operational Excellence at Amazon

REL12-BP03 Test dei requisiti di scalabilità 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 per il carico di lavoro su richiesta. Invece di affidarti a un ambiente di test con risorse ridotte verticalmente, che potrebbe portare a previsioni imprecise dei comportamenti in produzione, puoi utilizzare il cloud per fornire un ambiente di test che rispecchi fedelmente l'ambiente di produzione previsto. Questo ambiente consente di eseguire i test in una simulazione più precisa delle condizioni reali in cui si trova l'applicazione.

Oltre ai test sulle prestazioni, è essenziale verificare che le risorse di base, le impostazioni di dimensionamento, Service Quotas e la progettazione di resilienza funzionino come previsto sotto carico. Questo approccio olistico verifica che l'applicazione sia in grado di scalare in modo affidabile e funzionare come richiesto, anche nelle condizioni più difficili.

Risultato desiderato: il carico di lavoro mantiene il comportamento previsto anche quando è soggetto a picchi di carico. Affronti in modo proattivo tutti i problemi di prestazioni che possono verificarsi con la crescita e l'evoluzione dell'applicazione.

Anti-pattern comuni:

- Utilizzi ambienti di test che non corrispondono strettamente all'ambiente di produzione.
- Tratti il test di carico come un'attività separata e una tantum, anziché come una parte integrata della pipeline di integrazione continua (CI) dell'implementazione.
- Non definisci requisiti di prestazione chiari e misurabili, come tempi di risposta, throughput e obiettivi di scalabilità.
- Esegui test con scenari di carico non realistici o insufficienti e non esegui test per picchi di carico, picchi improvvisi e carico elevato sostenuto.
- Non solleciti il carico di lavoro superando i limiti di carico previsti.
- Utilizzi strumenti di test di carico e di profiling delle prestazioni inadeguati o inappropriati.
- Non disponi di sistemi di monitoraggio e di avviso completi per monitorare le metriche delle prestazioni e rilevare anomalie.

Vantaggi dell'adozione di questa best practice:

- I test di carico aiutano a identificare i potenziali colli di bottiglia delle prestazioni del sistema prima del passaggio in produzione. Quando simuli il traffico e i carichi di lavoro a livello di produzione, puoi identificare le aree in cui il sistema può avere difficoltà a gestire il carico, come tempi di risposta lenti, vincoli delle risorse o errori di sistema.
- Testando il sistema in varie condizioni di carico, puoi comprendere meglio i requisiti delle risorse necessari per supportare il carico di lavoro. Queste informazioni possono aiutarti a prendere decisioni informate sull'allocazione delle risorse e a prevenire l'eccesso o il difetto di provisioning di risorse.
- Per identificare i potenziali punti di errore, puoi osservare come si comporta il carico di lavoro in condizioni di carico elevato. Queste informazioni aiutano a migliorare l'affidabilità e la resilienza del

carico di lavoro implementando meccanismi di tolleranza agli errori, strategie di failover e misure di ridondanza, a seconda dei casi.

- Identifichi e risolvi tempestivamente i problemi di prestazioni, evitando così le costose conseguenze di interruzioni del sistema, tempi di risposta lenti e utenti insoddisfatti.
- I dati dettagliati sulle prestazioni e le informazioni di profiling raccolte durante i test possono aiutarti a risolvere i problemi correlati alle prestazioni che potrebbero sorgere in produzione. Questo può portare a una risposta e a una risoluzione più rapida degli incidenti, riducendo l'impatto sugli utenti e sulle operazioni dell'organizzazione.
- In alcuni settori, il test proattivo delle prestazioni può aiutare il carico di lavoro a soddisfare gli standard di conformità, riducendo il rischio di sanzioni o problemi legali.

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

Guida all'implementazione

Il primo passo consiste nel definire una strategia di test completa che copra tutti gli aspetti dei requisiti di dimensionamento e prestazioni. Per iniziare, definisci chiaramente gli obiettivi di livello di servizio (SLO) del carico di lavoro in base alle esigenze aziendali, come il throughput, l'istogramma della latenza e il tasso di errore. Quindi, progetta una suite di test in grado di simulare vari scenari di carico che vanno dall'utilizzo medio a picchi improvvisi e a carichi di picco sostenuti, e verifica che il comportamento del carico di lavoro soddisfi gli SLO. Questi test devono essere automatizzati e integrati nella pipeline di integrazione e implementazione continua per individuare le regressioni delle prestazioni nelle prime fasi del processo di sviluppo.

Per testare efficacemente il dimensionamento e le prestazioni, investi negli strumenti e nell'infrastruttura corretti. Ciò include strumenti di test di carico in grado di generare un traffico utente realistico, strumenti di profiling delle prestazioni per identificare i colli di bottiglia e soluzioni di monitoraggio per tracciare le metriche chiave. È importante verificare che gli ambienti di test corrispondano fedelmente all'ambiente di produzione in termini di infrastrutture e condizioni ambientali, in modo da ottenere risultati il più possibile accurati. Per rendere più facile replicare e scalare in modo affidabile le configurazioni simili a quelle di produzione, utilizza infrastructure as code e le applicazioni basate su container.

I test di dimensionamento e sulle prestazioni sono un processo continuo, non un'attività una tantum. Implementa il monitoraggio completo e gli avvisi per monitorare le prestazioni dell'applicazione in produzione e utilizza questi dati per perfezionare continuamente le strategie di test e gli sforzi di ottimizzazione. Analizza regolarmente i dati sulle prestazioni per identificare i problemi emergenti,

testare nuove strategie di dimensionamento e implementare ottimizzazioni per migliorare l'efficienza e l'affidabilità dell'applicazione. Quando adotti un approccio iterativo e impari costantemente dai dati di produzione, puoi verificare che l'applicazione è in grado di adattarsi alle richieste variabili degli utenti e mantenere la resilienza e le prestazioni ottimali nel tempo.

Passaggi dell'implementazione

1. Stabilisci requisiti di prestazioni chiari e misurabili, come tempi di risposta, throughput e obiettivi di scalabilità. Questi requisiti devono essere basati sui modelli di utilizzo del carico di lavoro, sulle aspettative degli utenti e sulle esigenze aziendali.
2. Seleziona e configura uno strumento di test del carico in grado di simulare accuratamente i modelli di carico e il comportamento degli utenti nell'ambiente di produzione.
3. Per migliorare la precisione dei risultati dei test, configura un ambiente di test che corrisponde strettamente all'ambiente di produzione, comprese le condizioni dell'infrastruttura e dell'ambiente.
4. Crea una suite di test che copre un'ampia gamma di scenari, dai modelli di utilizzo medio ai picchi di carico, ai picchi rapidi e ai carichi elevati sostenuti. Integra i test nelle pipeline di implementazione e distribuzione continua per individuare regressioni delle prestazioni fin dalle prime fasi del processo di sviluppo.
5. Esegui test di carico per simulare il traffico reale degli utenti e capire come si comporta l'applicazione in diverse condizioni di carico. Per sollecitare l'applicazione, supera il carico previsto e osserva il suo comportamento, come il degrado dei tempi di risposta, l'esaurimento delle risorse o gli errori di sistema. Ciò aiuta a identificare il punto di rottura dell'applicazione e a informare le strategie di dimensionamento. Valuta la scalabilità del carico di lavoro aumentandolo progressivamente e misura l'impatto sulle prestazioni per identificare i limiti di dimensionamento e pianificare le esigenze di capacità future.
6. Implementa monitoraggio completo e avvisi per tracciare le metriche delle prestazioni, rilevare le anomalie e avviare azioni di dimensionamento o notifiche quando vengono superate le soglie.
7. Monitora e analizza costantemente i dati sulle prestazioni per identificare le aree di miglioramento. Itera le strategie di test e gli sforzi di ottimizzazione.

Risorse

Best practice correlate:

- REL01-BP04 Monitoraggio e gestione delle quote
- REL06-BP01 Monitoraggio di tutti i componenti per il carico di lavoro (generazione)

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

Documenti correlati:

- Load testing applications
- Test del carico distribuito su AWS
- Monitoraggio delle prestazioni delle applicazioni
- Amazon EC2 Testing Policy

Esempi correlati:

- Distributed Load Testing on AWS (GitHub)

Strumenti correlati:

- Profilatore Amazon CodeGuru
- Amazon CloudWatch RUM
- Apache JMeter
- K6
- Vegeta
- Hey
- ab
- wrk
- Test del carico distribuito su AWS

REL12-BP04 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 guasti 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. – Principles of Chaos Engineering

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.

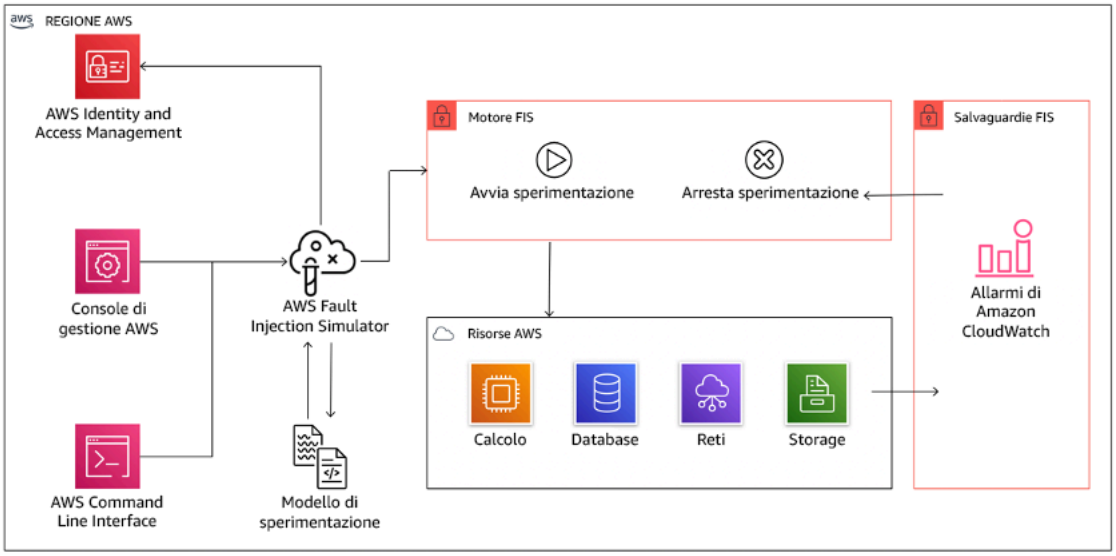
Per gli errori a livello di applicazione, ad esempio gli arresti anomali, puoi iniziare utilizzando fattori di stress, ad esempio l'esaurimento della memoria o della CPU.

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 guasti 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 su diversi tipi di risorse, tra cui Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS) e Amazon RDS. Questi errori includono la terminazione delle risorse, la forzatura dei failover, l'applicazione di fattori di stress a CPU o memoria, la limitazione (della larghezza 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 guasti per i carichi di lavoro.

Esistono anche diverse opzioni di terze parti per gli esperimenti di iniezione di guasti. Queste opzioni comprendono strumenti open source come Chaos Toolkit, Chaos Mesh e Litmus Chaos, nonché altre opzioni commerciali come Gremlin. Per ampliare l'ambito dei guasti che è possibile iniettare in AWS, AWS FIS si integra con Chaos Mesh e Litmus Chaos, così da coordinare i flussi di lavoro di iniezione di guasti 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

1. Determinazione dei guasti da utilizzare per gli esperimenti.

   Valutazione della progettazione del carico di lavoro a livello di resilienza. Tali progettazioni (create seguendo le best practice del Framework Well-Architected) tengono conto dei rischi basati su dipendenze critiche, eventi passati, problemi noti e 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 sulla creazione di tali elenchi, consulta il whitepaper sulla prontezza operativa, che illustra come creare un processo finalizzato alla prevenzione del ripetersi di incidenti precedenti. 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 illustrato in maggiore dettaglio da Adrian Cockcroft in Failure Modes and Continuous Resilience.

2. Assegna una priorità a ogni errore.

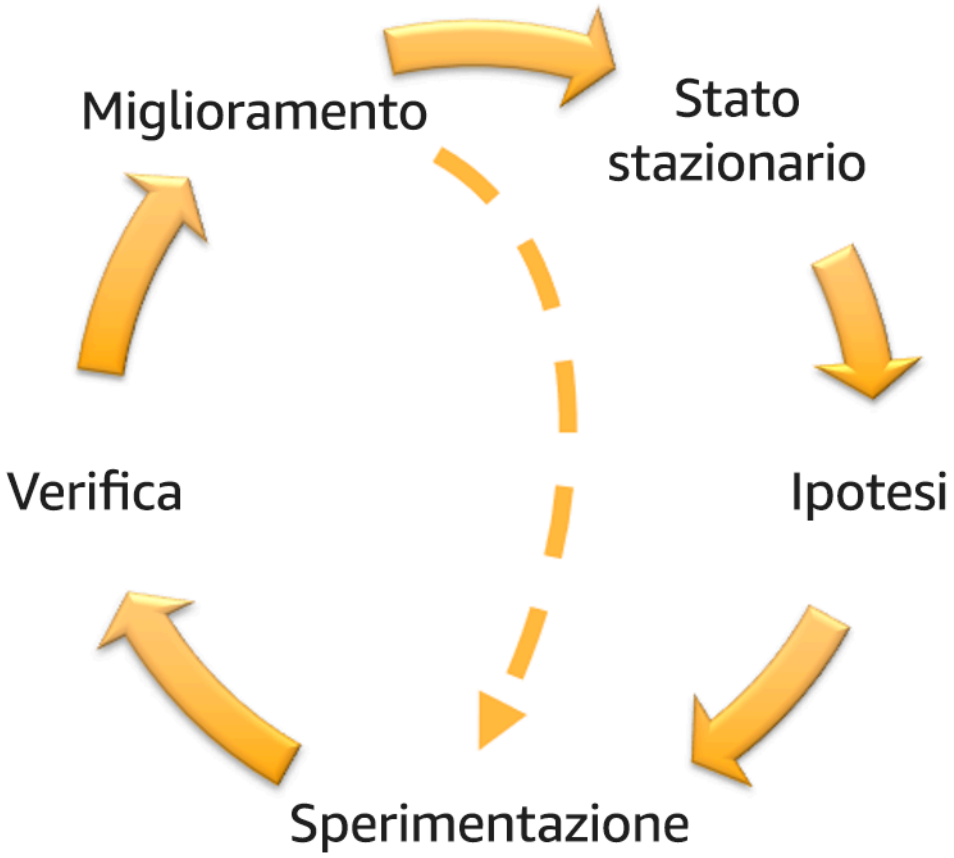
   Comincia con una categorizzazione approssimativa, ad esempio alta, media o bassa. Per valutare la priorità, considera la frequenza dell'errore e l'impatto dell'errore sul carico di lavoro nel suo complesso.

   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 limitazione (della larghezza di banda della rete) 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 guasti.
3. Per ogni esperimento eseguito, attieniti ai principi del volano dell'ingegneria del caos e della resilienza continua nella figura seguente.



Volano dell'ingegneria del caos e della resilienza continua, che utilizza il metodo scientifico di Adrian Hornsby.

a. 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.

b. 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 poiché 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 *`guasto specifico`*, il carico di lavoro *`nome del carico di lavoro illustrerà la mitigazione dei controlli`* per controbilanciare l'*`impatto sulle metriche aziendali o tecniche`*.

Per 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 di Elastic Load Balancing del sistema degli ordini farà sì che Elastic Load Balancing si limiti a inviare richieste alle rimanenti istanze integre, mentre 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 connetterà all'istanza database in standby Amazon RDS per mantenere meno di un minuto di errori di lettura o scrittura del database (stato stazionario).

c. 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. Per scenari noti sconosciuti si intendono gli scenari di cui sei consapevole ma che non comprendi appieno, mentre scenari completamente sconosciuti si riferiscono a quegli scenari a te non noti e che non comprendi appieno. 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 su AWS, AWS FIS offre diverse simulazioni di guasto predefinite denominate operazioni. Puoi anche definire operazioni personalizzate eseguibili in AWS FIS utilizzando i documenti di 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 su una varietà più ampia di esperimenti mediante AWS FIS, consulta anche il lab Resilient and Well-Architected Apps with Chaos Engineering. 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 andrebbero eseguiti in produzione con un carico reale mediante distribuzioni canary che attivano l'implementazione di 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 monitoraggio 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 guasti, 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 con difetti 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.

d. Verifica l'ipotesi.

   Principles of Chaos Engineering fornisce 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. Il throughput 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 monitoraggio sintetico, definito canary utente, in qualsiasi API o URI a cui il client del carico di lavoro ha accesso diretto.

e. 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 carico di lavoro per azzerare l'impatto dell'errore applicando le best practice illustrate nel pilastro dell'affidabilità di AWS Well-Architected. Puoi trovare ulteriori informazioni nella AWS Builder's Library, che offre, tra gli altri, articoli su come migliorare i controlli dell'integrità o 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.

4. 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 su come effettuare tale operazione, consulta questo blog su come eseguire esperimenti AWS FIS mediante AWS CodePipeline. Poi fare pratica con questo lab sugli esperimenti AWS FIS ricorrenti in una pipeline CI/CD.

   Gli esperimenti di iniezione di guasti fanno inoltre parte delle giornate di gioco (consulta REL12-BP05 Esecuzione regolare di GameDay). 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.

5. Acquisisci e archivia i risultati degli esperimenti.

   I risultati degli esperimenti di iniezione di guasti 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. Puoi inserire la creazione di log degli esperimenti mediante AWS FIS nel processo di acquisizione dei dati.

Risorse

Best practice correlate:

- REL08-BP03 Integra i test di resilienza come parte della tua implementazione
- REL13-BP03 Esecuzione di test sull'implementazione del disaster recovery per convalidare l'implementazione

Documenti correlati:

- Che cos'è AWS Fault Injection Service?
- Che cos'è AWS Resilience Hub?
- Principles of Chaos Engineering
- Chaos Engineering: Planning your first experiment
- Resilience Engineering: Learning to Embrace Failure
- Chaos Engineering stories
- Evitare il fallback nei sistemi distribuiti
- Distribuzione canary per gli esperimenti di ingegneria del caos

Video correlati:

- AWS re:Invent 2020: Testing resiliency using chaos engineering (ARC316)
- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world (CMY301)

Strumenti correlati:

- AWS Fault Injection Service
- Marketplace AWS: Gremlin Chaos Engineering Platform
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP05 Esecuzione regolare di GameDay

Conduci GameDay per esercitare regolarmente le tue procedure di risposta agli eventi e alle compromissioni che incidono sul carico di lavoro. Coinvolgi gli stessi team responsabili della gestione

degli scenari di produzione. Queste esercitazioni aiutano ad applicare le misure per prevenire l'impatto sugli utenti causato da eventi di produzione. Esercitando le procedure di risposta in condizioni realistiche, puoi identificare e risolvere eventuali lacune o punti deboli prima che si verifichi un evento reale.

I GameDay simulano eventi in ambienti simili a quelli di produzione per testare sistemi, processi e risposte dei team. Lo scopo è quello di eseguire le stesse azioni che verrebbero eseguite dal team se l'evento si verificasse realmente. Questi esercizi aiutano a capire dove è possibile apportare miglioramenti e contribuire a sviluppare l'esperienza organizzativa nel gestire eventi e compromissioni. Questi dovrebbero essere svolti regolarmente, in modo che il team sappia costruire abitudini radicate su come rispondere.

I GameDay preparano i team a gestire gli eventi di produzione con maggiore sicurezza. I team ben allenati sono più in grado di individuare e rispondere rapidamente ai vari scenari. Ciò si traduce in un significativo miglioramento della postura di prontezza e resilienza.

Risultato desiderato: conduci GameDay sulla resilienza in modo coerente e programmato. Questi GameDay sono visti come una parte normale e attesa dell'attività. La tua organizzazione ha costruito una cultura di preparazione e quando si verificano problemi di produzione, i team sono ben preparati a rispondere efficacemente, a risolvere i problemi in modo efficiente e a mitigare l'impatto sui clienti.

Anti-pattern comuni:

- Documenti le procedure, ma non le metti mai in pratica.
- Negli esercizi di prova escludi i responsabili delle decisioni aziendali.
- Organizzi un GameDay, ma non informi tutte le parti interessate.
- Ti concentri esclusivamente sugli errori tecnici, ma non coinvolgi le parti interessate aziendali.
- Non incorpori le lezioni apprese nei GameDay nei processi di recupero.
- Incolpi i team per errori o bug.

Vantaggi dell'adozione di questa best practice:

- Migliorare le capacità di risposta: nei GameDay, i team si esercitano a svolgere i propri compiti e a testare i meccanismi di comunicazione durante gli eventi simulati, creando una risposta più coordinata ed efficiente nelle situazioni di produzione.
- Identifica e risolvi le dipendenze: gli ambienti complessi spesso comportano dipendenze intricate tra vari sistemi, servizi e componenti. I GameDay possono aiutare a identificare e

risolvere queste dipendenze e a verificare che i sistemi e i servizi critici siano adeguatamente coperti dalle procedure del runbook e che possano essere aumentati verticalmente o ripristinati tempestivamente.

- Promuovere una cultura della resilienza: i GameDay possono aiutare a coltivare una mentalità di resilienza all'interno di un'organizzazione. Quando si coinvolgono team interfunzionali e parti interessate, questi esercizi promuovono la consapevolezza, la collaborazione e la comprensione condivisa dell'importanza della resilienza in tutta l'organizzazione.
- Miglioramento e adattamento continui: GameDay regolari aiutano a valutare e adattare continuamente le strategie di resilienza, in modo da mantenerle pertinenti ed efficaci di fronte a circostanze mutevoli.
- Aumentare la fiducia nel sistema: GameDay riusciti possono aiutare a creare fiducia nella capacità del sistema di resistere e riprendersi da interruzioni.

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

Guida all'implementazione

Una volta progettate e implementate le misure di resilienza necessarie, conduci un GameDay per verificare che tutto funzioni come previsto in produzione. Un GameDay, soprattutto il primo, deve coinvolgere tutti i membri del team. Tutte le parti interessate e i partecipanti devono essere informati in anticipo su data, ora e scenari simulati.

Durante il GameDay, i team coinvolti simulano vari eventi e potenziali scenari secondo le procedure prescritte. I partecipanti monitorano e valutano attentamente l'impatto di questi eventi simulati. Se il sistema funziona come previsto, i meccanismi automatici di rilevamento, dimensionamento e autoriparazione devono attivarsi e generare un impatto minimo o nullo sugli utenti. Se il team rileva un impatto negativo, esegue il rollback del test e corregge i problemi identificati, sia con mezzi automatici sia con interventi manuali documentati nei runbook applicabili.

Per migliorare continuamente la resilienza, è fondamentale documentare e incorporare le lezioni apprese. Questo processo è un ciclo di feedback che acquisisce sistematicamente le intuizioni dei GameDay e le utilizza per migliorare i sistemi, i processi e le capacità del team.

Per aiutare a riprodurre scenari reali in cui i componenti o i servizi del sistema possono generare errori imprevisti, si consiglia di iniettare errori simulati come esercizio del GameDay. I team possono testare la resilienza e la tolleranza agli errori dei loro sistemi e simulare i processi di risposta e di ripristino agli incidenti in un ambiente controllato.

In AWS, i GameDay possono essere realizzati con repliche dell'ambiente di produzione utilizzando infrastructure as code. Questo processo consente di eseguire i test in un ambiente sicuro e molto simile a quello di produzione. Prendi in considerazione il servizio AWS Fault Injection Service per creare diversi scenari di errore. Utilizza servizi come Amazon CloudWatch e AWS X-Ray per monitorare il comportamento del sistema durante i GameDay. Utilizza AWS Systems Manager per gestire ed eseguire i playbook e utilizza AWS Step Functions per orchestrare i flussi di lavoro ricorrenti del GameDay.

Passaggi dell'implementazione

- Stabilisci un programma per i GameDay: sviluppa un programma strutturato che definisce la frequenza, la portata e gli obiettivi dei GameDay. Coinvolgi le principali parti interessate e gli esperti in materia nella pianificazione e nello svolgimento di questi esercizi.
- Prepara il GameDay:
  1. Identifica i servizi chiave critici per l'azienda che sono al centro del GameDay. Cataloga e mappa le persone, i processi e le tecnologie che supportano tali servizi.
  2. Stabilisci il programma del GameDay e prepara i team coinvolti a partecipare all'evento. Prepara i servizi di automazione per simulare gli scenari pianificati ed esegui i processi di ripristino appropriati. I servizi AWS come AWS Fault Injection Service, AWS Step Functions e AWS Systems Manager possono aiutarti ad automatizzare vari aspetti dei GameDay, come l'iniezione di errori e l'avvio di azioni di ripristino.
- Esegui la simulazione: nel GameDay, esegui lo scenario pianificato. Osserva e documenta come le persone, i processi e le tecnologie reagiscono all'evento simulato.
- Conduci revisioni post-esercizio: dopo il GameDay, conduci una sessione retrospettiva per esaminare le lezioni apprese. Identifica le aree di miglioramento e le azioni necessarie per migliorare la resilienza operativa. Documenta gli esiti e tieni traccia delle eventuali modifiche necessarie per migliorare le strategie di resilienza e la preparazione al completamento.

Risorse

Best practice correlate:

- REL12-BP01 Utilizzo dei playbook per analizzare gli errori
- REL12-BP04 Test della resilienza tramite l'utilizzo dell'ingegneria del caos
- OPS04-BP01 Identificazione degli indicatori chiave di prestazione
- OPS07-BP03 Utilizzo di runbook per eseguire le procedure

- OPS10-BP01 Utilizzo di un processo per la gestione di eventi, incidenti e problemi

Documenti correlati:

- Che cos'è AWS GameDay?
- AWS Well-Architected Concepts - Game Day

Video correlati:

- AWS re:Invent 2023 - Practice like you play: How Amazon scales resilience to new heights

Esempi correlati:

- AWS Workshop - Navigate the storm: Unleashing controlled chaos for resilient systems
- Build Your Own Game Day to Support Operational Resilience

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 ed RPO sono gli 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 disaster recovery per un carico di lavoro.

Sia la disponibilità che il disaster recovery si affidano alle stesse best practice, come il monitoraggio degli errori, l'implementazione su più sedi e il failover automatico. Tuttavia, la disponibilità si focalizza su componenti del carico di lavoro, mentre il disaster recovery si concentra su copie singole dell'intero carico di lavoro. Il disaster recovery 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 disaster recovery per convalidare l'implementazione

- REL13-BP04 Gestione della deviazione di configurazione nel sito o nella regione del disaster recovery
- REL13-BP05 Automatizzazione del ripristino

REL13-BP01 Definizione degli obiettivi di ripristino in caso di downtime e perdita di dati

Guasti ed errori possono avere un impatto sull'attività in diversi modi. In primo luogo, possono causare l'interruzione del servizio (tempo di inattività). In secondo luogo, possono causare la perdita, l'incoerenza o il mancato aggiornamento dei dati. Per guidare le modalità di risposta e recupero dagli errori, definisci un Obiettivo del tempo di ripristino (RTO) e un Obiettivo del punto di ripristino (RPO) per ogni carico di lavoro. L'Obiettivo del tempo di ripristino (RTO) è il ritardo massimo accettabile tra l'interruzione del servizio e il ripristino del servizio. L'Obiettivo del punto di ripristino (RPO) è il tempo massimo accettabile dopo l'ultimo punto di ripristino dei dati.

Risultato desiderato: ogni carico di lavoro ha un RTO e un RPO designati in base a considerazioni tecniche e all'impatto aziendale.

Anti-pattern comuni:

- Non hai designato gli obiettivi di ripristino.
- Selezioni obiettivi di ripristino arbitrari.
- Selezioni obiettivi di ripristino troppo blandi e che non soddisfano gli obiettivi aziendali.
- Non hai valutato l'impatto del tempo di inattività e della perdita di dati.
- Scegli obiettivi di ripristino non realistici, come il tempo zero di ripristino o nessuna perdita di dati, che potrebbero non essere raggiungibili per la configurazione del carico di lavoro.
- Selezioni obiettivi di ripristino più severi rispetto agli obiettivi aziendali reali. Questo costringe a implementazioni di ripristino più costose e complicate rispetto alle esigenze del carico di lavoro.
- Selezioni obiettivi di ripristino incompatibili con quelli di un carico di lavoro dipendente.
- Non tieni conto dei requisiti normativi e di conformità.

Vantaggi dell'adozione di questa best practice: quando definisci gli RTO e gli RPO per i carichi di lavoro, stabilisci obiettivi chiari e misurabili per il ripristino in base alle esigenze aziendali. Una volta fissati questi obiettivi, puoi creare piani di disaster recovery (DR) su misura per raggiungerli.

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

Guida all'implementazione

Costruisci una matrice o un foglio di lavoro per guidare la pianificazione del disaster recovery. Nella matrice, crea diverse categorie o livelli di carico di lavoro in base al loro impatto sull'azienda (ad esempio, critico, alto, medio e basso) e i relativi RTO e RPO da raggiungere per ciascuno di essi. La matrice seguente fornisce un possibile esempio (nota che i valori RTO e RPO possono differire) da seguire:

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

Esempio di matrice di disaster recovery

Per ogni carico di lavoro, devi analizzare e comprendere l'impatto sull'azienda del tempo di inattività e della perdita di dati. L'impatto cresce tipicamente con il tempo di inattività e la perdita di dati, ma la forma dell'impatto può variare in base al tipo di carico di lavoro. Ad esempio, un tempo di inattività fino a un'ora potrebbe avere un impatto ridotto, ma in seguito l'impatto potrebbe intensificarsi rapidamente. L'impatto può assumere diverse forme, tra cui l'impatto finanziario (come la perdita di fatturato), l'impatto a livello di reputazione (tra cui la perdita di fiducia dei clienti), l'impatto operativo (come il mancato pagamento degli stipendi o la diminuzione della produttività) e il rischio normativo. Una volta completato, assegna il carico di lavoro al livello appropriato.

Considera le seguenti domande quando analizzi l'impatto del guasto o dell'errore:

1. Qual è il tempo massimo di indisponibilità del carico di lavoro prima che si verifichi un impatto inaccettabile sull'azienda?
2. Qual è l'intensità e il tipo di impatto che l'azienda subirà a causa di un'interruzione del carico di lavoro? Prendi in considerazione tutti i tipi di impatto, compresi quelli finanziari, a livello di reputazione, operativi e normativi.

3. Qual è la quantità massima di dati che può essere persa o non recuperata prima che si verifichi un impatto inaccettabile sull'azienda?
4. I dati persi possono essere ricreati da altre fonti (note anche come dati derivati)? In tal caso, considera anche gli RPO di tutti i dati di origine utilizzati per ricreare i dati del carico di lavoro.
5. Quali sono gli obiettivi di ripristino e le aspettative di disponibilità dei carichi di lavoro da cui questo dipende (a valle)? Gli obiettivi del carico di lavoro devono essere raggiungibili in base alle capacità di ripristino delle relative dipendenze a valle. Valuta possibili soluzioni alternative o mitigazioni delle dipendenze downstream che possono migliorare la capacità di ripristino di questo carico di lavoro.
6. Quali sono gli obiettivi di ripristino e le aspettative di disponibilità dei carichi di lavoro che dipendono da questo (upstream)? Gli obiettivi del carico di lavoro upstream possono richiedere che questo carico di lavoro disponga di capacità di ripristino più rigorose di quanto non sembri a prima vista.
7. Esistono obiettivi di recupero diversi in base al tipo di incidente? Ad esempio, si possono avere RTO e RPO diversi a seconda che l'incidente riguardi una zona di disponibilità o un'intera Regione.
8. Gli obiettivi di ripristino cambiano durante determinati eventi o periodi dell'anno? Ad esempio, si possono avere RTO e RPO diversi in base alle stagioni dello shopping, agli eventi sportivi, alle vendite speciali e al lancio di nuovi prodotti.
9. In che modo gli obiettivi di ripristino si allineano con la strategia di disaster recovery aziendale e organizzativa?
10. Ci sono implicazioni legali o contrattuali da considerare? Ad esempio, hai l'obbligo per contratto di fornire un servizio con un determinato RTO o RPO? In quali sanzioni potresti incorrere in caso di inadempienza?
11. Devi mantenere l'integrità dei dati per soddisfare i requisiti normativi o di conformità?

Il seguente foglio di lavoro può aiutarti a valutare ogni carico di lavoro. Puoi modificare questo foglio di lavoro per adattarlo alle tue esigenze specifiche, ad esempio aggiungendo 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

Passaggi dell'implementazione

1. Identifica le parti interessate aziendali e i team tecnici responsabili di ciascun carico di lavoro e collabora con loro.
2. Crea categorie o livelli di criticità per l'impatto del carico di lavoro nell'organizzazione. Le categorie di esempio sono: critico, alto, medio e basso. Per ogni categoria, scegli un RTO e un RPO che riflettano gli obiettivi e i requisiti aziendali.
3. Assegna a ciascun carico di lavoro una delle categorie di impatto create nel passaggio precedente. Per decidere in che modo un carico di lavoro rientra in una categoria, considera l'importanza del carico di lavoro per l'azienda e l'impatto di un'interruzione o di una perdita di dati e utilizza le domande di cui sopra come guida. Ne conseguono un RTO e un RPO per ogni carico di lavoro.
4. Considera l'RTO e l'RPO per ogni carico di lavoro determinato nel passaggio precedente. Coinvolgi i team aziendali e tecnici del carico di lavoro per determinare se gli obiettivi devono essere modificati. Ad esempio, le parti interessate aziendali potrebbero stabilire che siano necessari obiettivi più severi. In alternativa, i team di tecnici potrebbero decidere di modificare gli obiettivi per renderli raggiungibili con le risorse disponibili e i vincoli tecnologici.

Risorse

Best practice correlate:

- REL09-BP04 Ripristino periodico dei dati per verificare l'integrità e i processi di backup
- REL12-BP01 Utilizzo dei playbook per analizzare gli errori
- REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino
- REL13-BP03 Esecuzione di test sull'implementazione del disaster recovery per convalidare l'implementazione

Documenti correlati:

- AWS Architecture Blog: serie sul disaster recovery
- Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud (whitepaper di AWS)
- Managing resiliency policies with AWS Resilience Hub
- Partner APN: partner che possono assistere con il disaster recovery
- Marketplace AWS: prodotti utilizzabili per il disaster recovery

Video correlati:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- Disaster Recovery of Workloads on AWS

REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino

Definisci una strategia di disaster recovery 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: per ciascun carico di lavoro esiste una strategia di disaster recovery definita e implementata che consente a quel carico di lavoro di raggiungere gli obiettivi di disaster recovery. Le strategie di disaster recovery 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 disaster recovery ad-hoc quando si verifica un disastro.
- Assenza di piani per il disaster recovery.
- Dipendenza dalle operazioni del piano di controllo (control-plane) 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 disaster recovery nei carichi dl lavoro di loro proprietà.

Livello di rischio associato se questa best practice non fosse adottata: elevato. Senza una strategia di disaster recovery pianificata, implementata e testata, è poco probabile riuscire a raggiungere gli obiettivi di ripristino in caso di emergenze.

Guida all'implementazione

Una strategia di disaster recovery 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 disaster recovery (DR) su più zone di disponibilità (AZ) all'interno di un singolo Regione AWS può offrire la mitigazione rispetto a emergenze 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 disaster recovery di emergenza basata su più regioni.

Quando pianifichi una strategia di disaster recovery su più regioni, devi scegliere una delle seguenti strategie. Sono elencati in ordine crescente di costo e complessità e in ordine decrescente di RTO e RPO. La regione di ripristino indica una Regione AWS diversa da quella principale utilizzata per il carico di lavoro.

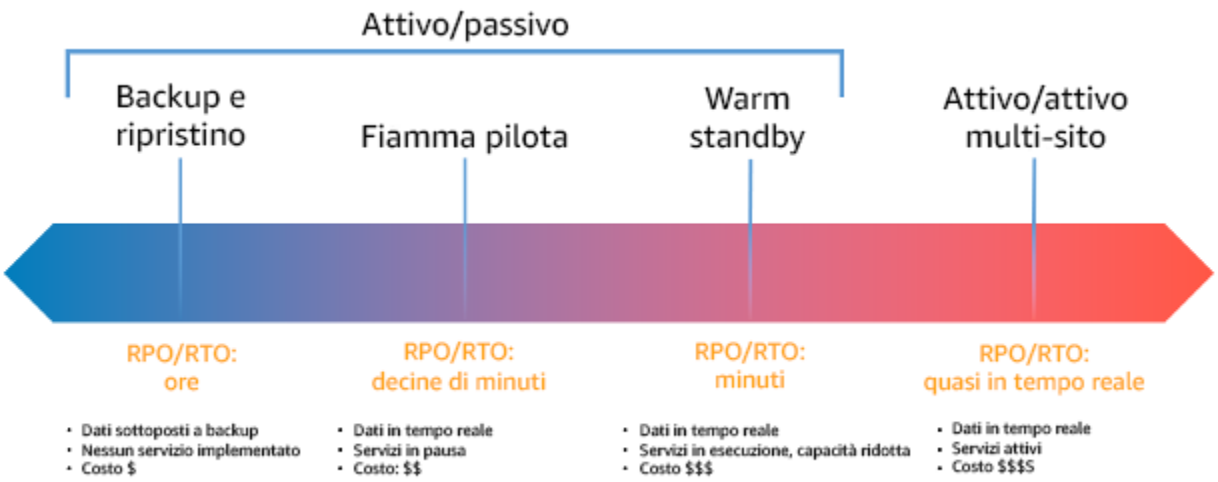


Figura 17: strategie di disaster recovery (DR)

- Backup e ripristino (RPO in ore, RTO in 24 ore o meno): esegui il backup dei dati e delle applicazioni nella regione di recupero. Adottando backup continui o automatizzati otterrai un ripristino point-in-time (PITR) 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 il modello Infrastructure as code per ridurre l'RTO), implementerai il codice e ripristinerai i dati del backup dopo un disastro nella regione di ripristino.
- Pilot light (RPO in minuti, RTO in decine di minuti): fornisci una copia dell'infrastruttura del carico di lavoro di base 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 in secondi, RTO in minuti): mantieni sempre una versione ridotta del carico di lavoro completamente funzionante in esecuzione nella regione di ripristino. I sistemi business critical sono completamente duplicati e sono sempre accesi, ma con un parco istanze ridotto verticalmente. I dati vengono replicati e si trovano nella regione di recupero. Al momento del ripristino, il sistema viene fatto aumentare verticalmente rapidamente per gestire il carico di produzione. Più si aumenterà verticalmente 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 vicino a zero, RTO uguale potenzialmente a zero): il carico di lavoro viene implementato in più Regioni AWS e serve attivamente il traffico da esse proveniente. 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'aumentare verticalmente, mentre Warm Standby richiede solo l'aumentare verticalmente (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 cloud-native (native del 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 disaster recovery in linea con i requisiti di ripristino di questo carico di lavoro.

   La scelta di una strategia di disaster recovery 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 disaster recovery Pilot Light o di Warm Standby soddisfano sia l'RTO sia i criteri per i costi.

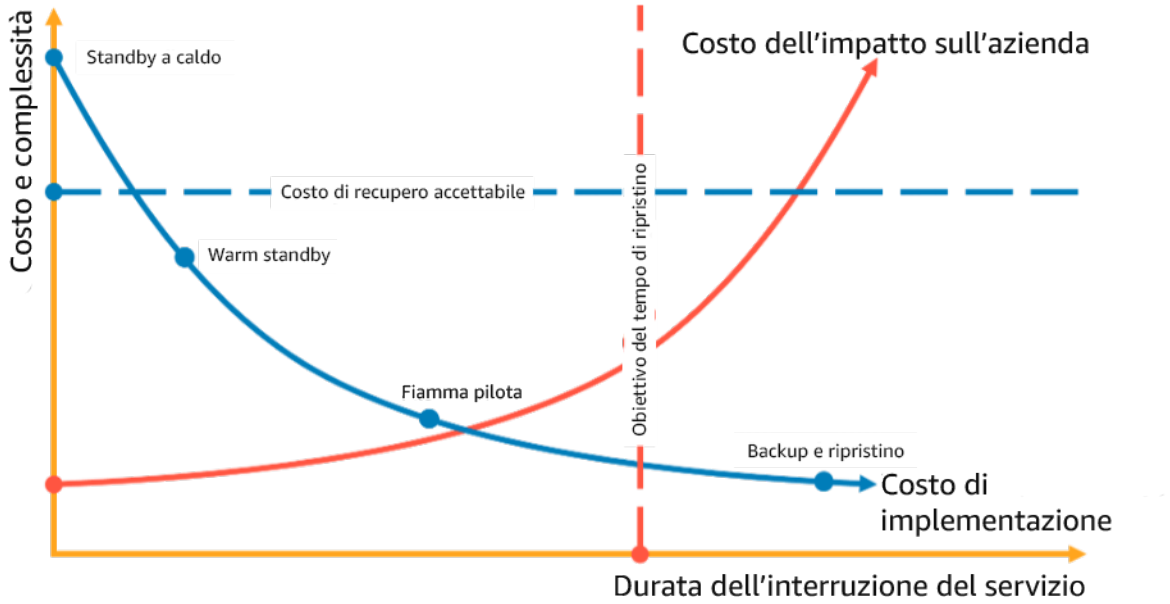


Figura 18: scegliere una strategia di disaster recovery in base all'RTO e ai costi

Per ulteriori informazioni, consulta Piano di continuità aziendale.

2. Esamina i modelli con cui la strategia di disaster recovery 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 disaster recovery, utilizzando aspetti di più strategie.

   Nei passaggi seguenti puoi applicare la strategia al carico di lavoro specifico.

   Backup e ripristino

   Backup ripristino è la strategia meno complessa da implementare, ma richiederà più tempo e impegno per ripristinare il carico di lavoro, generando così valori RTO e RPO più elevati. È buona pratica creare sempre backup dei dati e copiarli in un altro sito (ad esempio, un'altra Regione AWS).

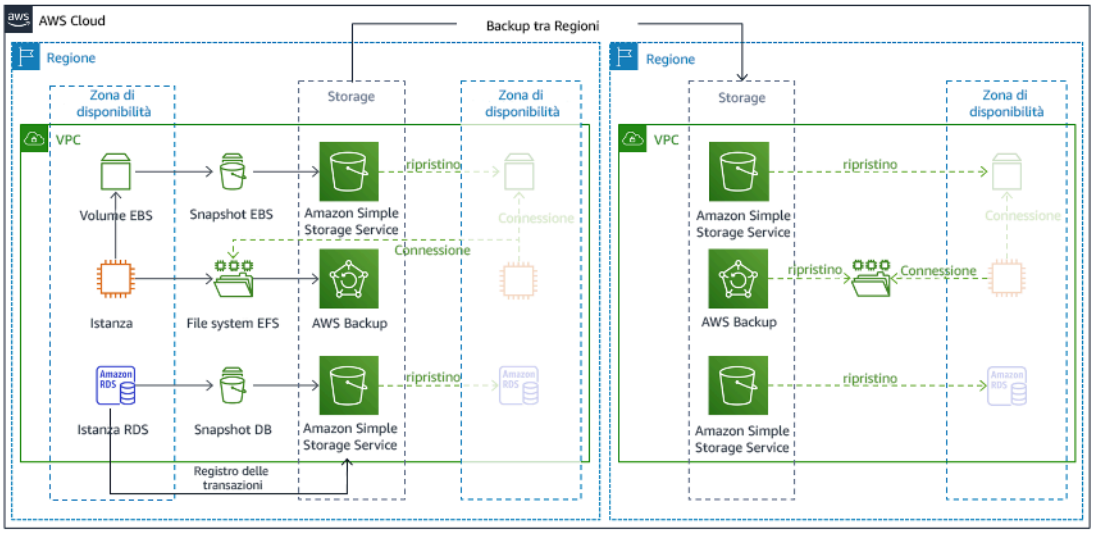


Figura 19: architettura di backup e ripristino

Per ulteriori dettagli su questa strategia, consulta [Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery].

Pilot light

Con l'approccio pilot light, replichi i dati dalla tua regione principale alla regione di ripristino. Le risorse di base utilizzate per l'infrastruttura del carico di lavoro vengono implementate 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.

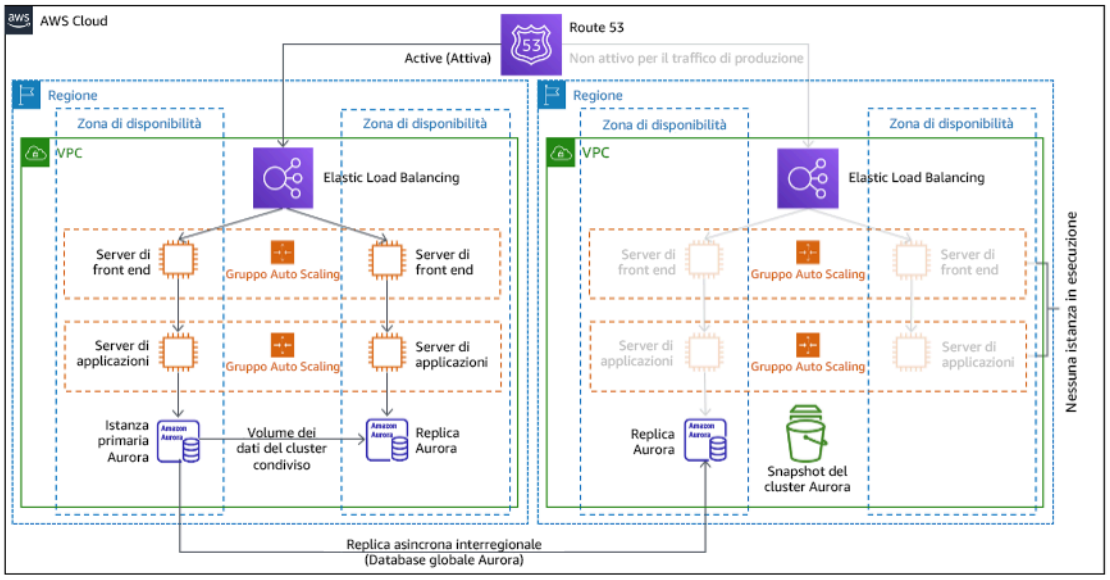


Figura 20: architettura pilot light

Per ulteriori informazioni su questa strategia, consulta Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby.

Warm standby

L'approccio warm standby implica la verifica della presenza di una copia ridotta verticalmente, ma comunque 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 ha raggiunto il massimo della capacità, allora viene definita come standby a caldo.

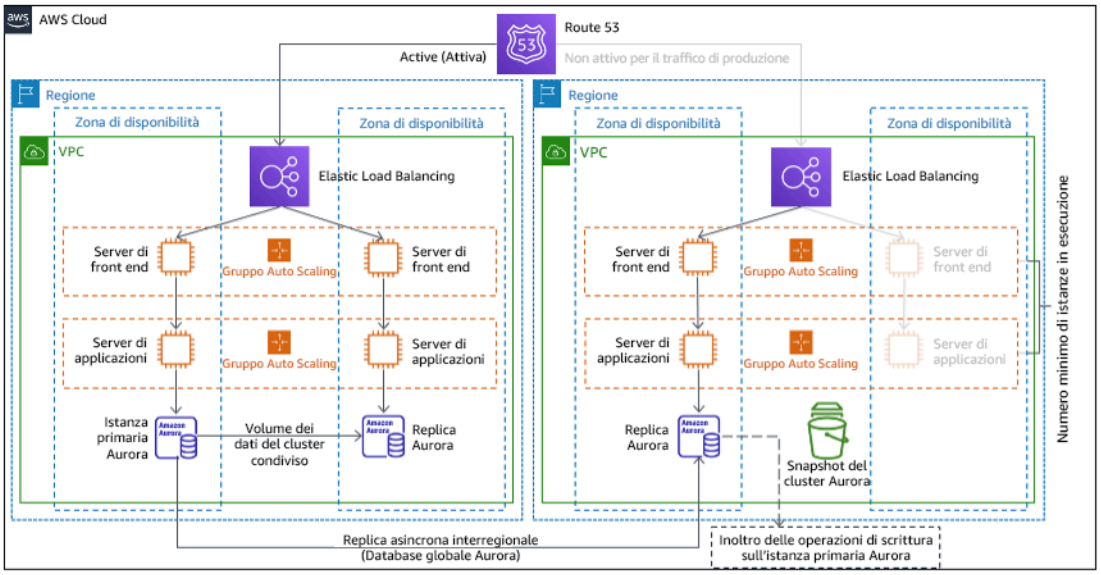


Figura 21: architettura warm standby

Se si utilizza Warm Standby o Pilot Light è necessario aumentare verticalmente le risorse nella regione di ripristino. Per verificare che sia disponibile capacità sufficiente quando necessario, valuta l'eventuale utilizzo delle prenotazioni della capacità per le istanze EC2. In caso di utilizzo di AWS Lambda, la concorrenza allocata può fornire ambienti di runtime pronti a rispondere immediatamente alle invocazioni della funzione.

Per ulteriori informazioni su questa strategia, consulta Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby.

Attivo/attivo multi-sito

Puoi eseguire il carico di lavoro simultaneamente in più regioni come parte di una strategia attivo/attivo multi-sito. 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 disaster recovery. 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 disaster recovery, 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. La strategia attivo/attivo multi-sito è la strategia di ripristino operativamente più complessa e dovrebbe essere selezionata solo quando lo richiedono i requisiti aziendali.

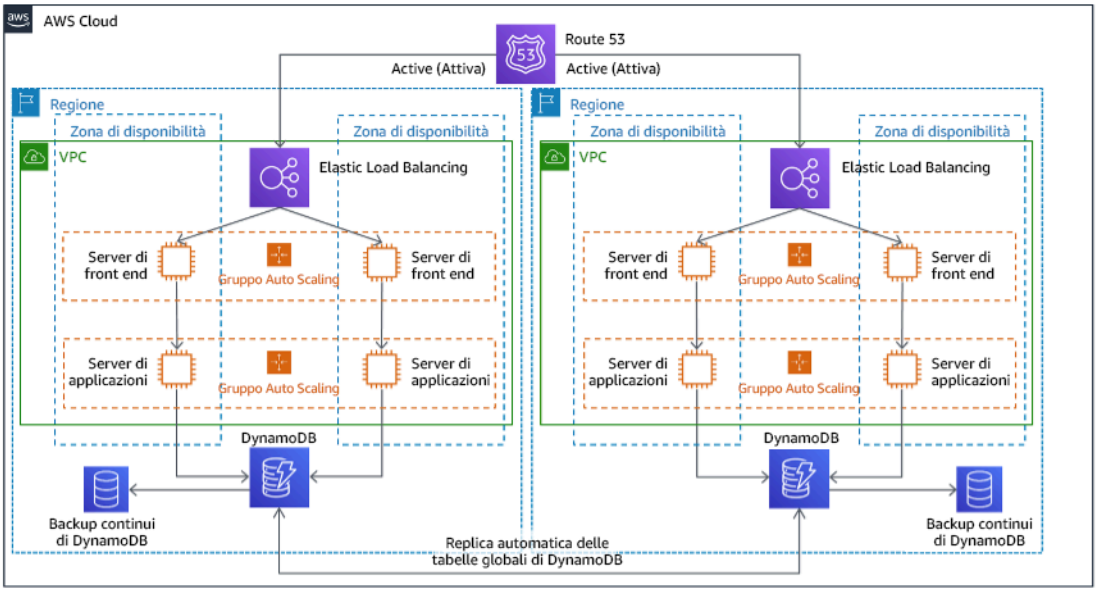


Figura 22: architettura attivo/attivo multi-sito

Per ulteriori informazioni su questa strategia, consulta [Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active].

AWS Elastic Disaster Recovery

Se stai prendendo in considerazione la strategia pilot light o warm standby per il disaster recovery, AWS Elastic Disaster Recovery potrebbe fornire un approccio alternativo con maggiori vantaggi. Elastic Disaster Recovery può offrire obiettivi RPO e RTO simili alla strategia warm standby, ma con l'approccio a basso costo della strategia pilot light. Elastic Disaster Recovery replica i dati dalla regione primaria a quella di ripristino, usando una protezione continua dei dati per conseguire un RPO misurato in secondi e un RTO misurabile 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.

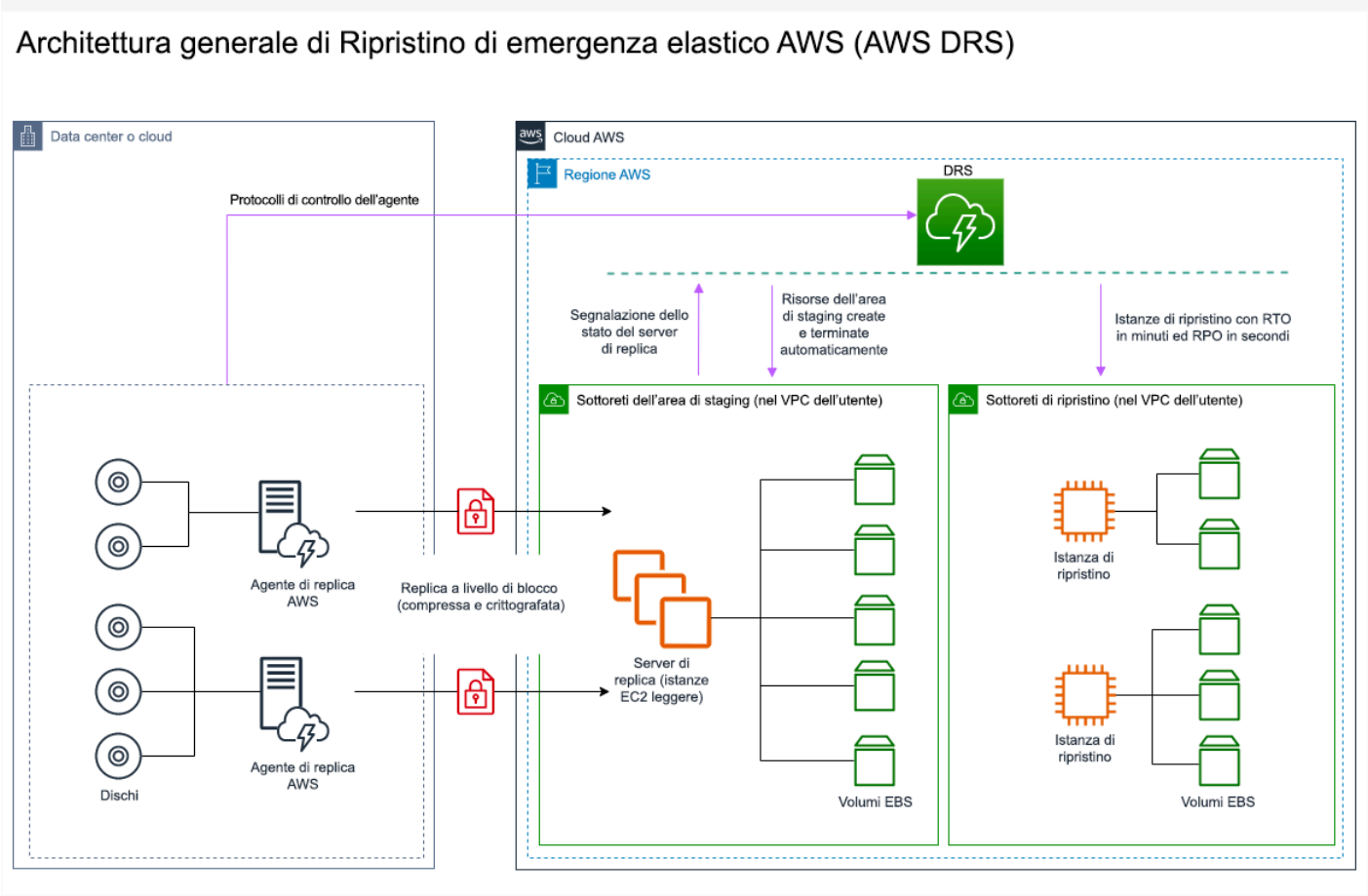


Figura 23: architettura AWS Elastic Disaster Recovery

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.

Utilizzo di più zone di disponibilità all'interno di una singola Regione AWS

Quando si usano più zone di disponibilità all'interno di un'unica regione, l'implementazione della strategia di disaster recovery 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 utilizza un approccio attivo/attivo multisito, in quanto le istanze Amazon EC2 ed Elastic Load Balancer dispongono di risorse implementate in più zone

di disponibilità, che gestiscono attivamente le richieste. L'architettura dimostra inoltre che lo standby a caldo, in cui in caso di errore dell'istanza Amazon RDS primaria (o della stessa zona di disponibilità), l'istanza di standby viene promossa a primaria.

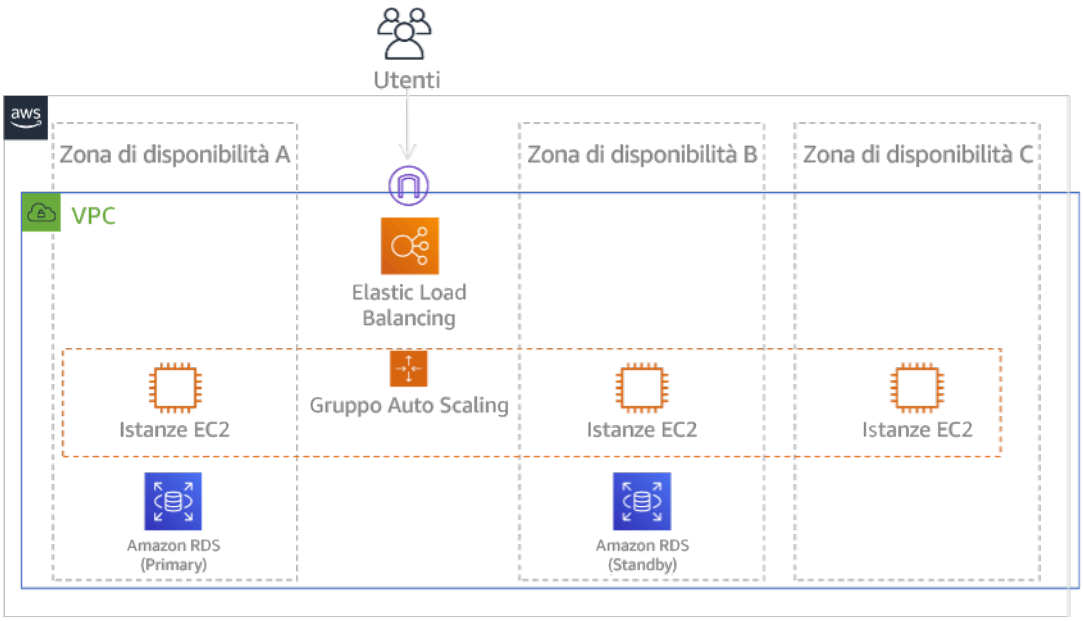


Figura 24: architettura con più zone di disponibilità

Oltre a questa architettura HA, devi aggiungere i backup di tutti i dati richiesti per eseguire il tuo carico di lavoro. Questo aspetto è di particolare importanza per i dati vincolati a una singola zona, come i volumi Amazon EBS o i cluster Amazon Redshift. In caso di errore di una zona di disponibilità, dovrai ripristinare i dati in un'altra zona di disponibilità. Laddove possibile, dovrai anche copiare i backup di dati su un'altra Regione AWS come forma di ulteriore protezione.

Un approccio alternativo meno comune alla singola Regione, ossia il disaster recovery multi-AZ, è presentato nel post del blog, Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack. 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 il modello Infrastructure as code, come AWS CloudFormation o strumenti di terze parti, come Hashicorp Terraform. Per l'implementazione in più account e regioni con una singola 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. I parametri e la logica condizionale di CloudFormation permettono di controllare se uno stack implementato è 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 disaster recovery richiedono l'esecuzione del backup delle origini dati all'interno della Regione AWS e la copia di tali backup nella regione di ripristino. AWS Backup offre una visualizzazione a livello centrale che consente di configurare, pianificare e monitorare i backup per tali risorse. Per Pilot Light, Warm Standby e Multi-sito attivo/attivo, devi anche replicare i dati dalla regione principale alle risorse di dati nella regione di ripristino, come le istanze DB di Amazon Relational Database Service (Amazon RDS) o le tabelle di 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 fra le 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'emergenza).

   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 di 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 implementare il codice. Puoi usare AWS Backup per ripristinare i dati nella regione di ripristino. Per ulteriori

dettagli, consulta REL09-BP01 Identificazione e backup di tutti i dati che richiedono un backup o riproduzione dei dati dalle origini. La ricostruzione dell'infrastruttura comprende la creazione di risorse come istanze EC2, oltre ad Amazon Virtual Private Cloud (Amazon VPC), sottoreti e gruppi di sicurezza necessari. Puoi automatizzare gran parte del processo di ripristino. Per scoprire come farlo, consulta questo post del blog.

5. Stabilisci e implementa le modalità con cui reindirizzerai il traffico al failover nel momento in cui sarà necessario (durante un'emergenza).

   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. Tra le opzioni, vi è l'utilizzo di 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 il failover avviato manualmente, puoi utilizzare Amazon Application Recovery Controller, che fornisce un'API del piano dati a elevata disponibilità per reinstradare il traffico verso la Regione di ripristino. Nella fase di implementazione del failover, usa le operazioni di piano dati ed evita quelle del piano di controllo (control-plane), come illustrato in REL11-BP04 Affidati al piano dati e non 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'emergenza è diminuita di intensità. Il provisioning di infrastruttura e codice alla regione principale in genere segue gli stessi passaggi usati inizialmente, affidandosi al modello Infrastructure as code e alle pipeline di implementazione 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. In caso di utilizzo delle tabelle globali Amazon DynamoDB, anche se la tabella nella regione principale era diventata non disponibile, quando torna di nuovo online, ripristina la propagazione di scritture in sospeso. Se utilizzi 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.

Dopo un failover, se puoi proseguire l'esecuzione nella tua regione di ripristino, valuta la possibilità di farlo nella tua regione principale. Effettueresti 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.

Se usi Elastic Disaster Recovery, il servizio fornirà assistenza per l'orchestrazione e l'automazione del processo di failback. Per ulteriori informazioni, consulta Performing a 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 Fai affidamento sul piano dati e non sul 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 disaster recovery
- Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud (whitepaper di AWS)
- Opzioni di disaster recovery 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 un fra le regioni
- Route 53: configurazione del failover DNS
- S3: replica tra regioni
- Che cos'è AWS Backup?
- What is Amazon Application Recovery Controller?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Get Started - AWS
- Partner APN: partner che possono assistere con il disaster recovery
- Marketplace AWS: prodotti utilizzabili per il disaster recovery

Video correlati:

- Disaster Recovery of Workloads on AWS
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services

REL13-BP03 Esecuzione di test sull'implementazione del disaster recovery 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: testare regolarmente il piano di disaster recovery verifica che funzioni quando necessario e che il tuo team sappia come eseguire la strategia.

Livello di rischio associato se questa best practice non fosse adottata: 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 trovarne le soluzioni prima del test successivo.
2. Per i carichi di lavoro basati su Amazon EC2, utilizza AWS Elastic Disaster Recovery per implementare e avviare istanze di esercitazione per la tua strategia di disaster recovery. AWS Elastic Disaster Recovery consente di eseguire esercitazioni in modo efficiente, per prepararsi per 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 il disaster recovery
- AWS Architecture Blog: serie sul disaster recovery
- Marketplace AWS: prodotti utilizzabili per il disaster recovery
- AWS Elastic Disaster Recovery
- Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud (whitepaper di AWS)
- AWS Elastic Disaster Recovery Preparing for Failover
- The Berkeley/Stanford recovery-oriented computing project
- What is AWS Fault Injection Simulator?

Video correlati:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS

REL13-BP04 Gestione della deviazione di configurazione nel sito o nella regione del disaster recovery

Per eseguire una procedura di disaster recovery (DR), il carico di lavoro deve essere in grado di riprendere le normali operazioni in modo tempestivo, senza perdite rilevanti di funzionalità o di dati, una volta che l'ambiente di DR è stato messo online. Per raggiungere questo obiettivo, è essenziale mantenere coerenti l'infrastruttura, i dati e le configurazioni tra l'ambiente di disaster recovery e l'ambiente primario.

Risultato desiderato: la configurazione e i dati del sito di disaster recovery sono identici a quelli del sito primario, il che facilita un ripristino rapido e completo in caso di necessità.

Anti-pattern comuni:

- Non riesci ad aggiornare le posizioni di ripristino quando vengono apportate modifiche alle posizioni primarie. Questo determina configurazioni obsolete che potrebbero ostacolare gli sforzi di ripristino.

- Non consideri le potenziali limitazioni, come le differenze di servizio tra le posizioni primaria e di ripristino, che possono portare a errori imprevisti durante il failover.
- Per l'aggiornamento e la sincronizzazione dell'ambiente di disaster recovery fai riferimento a processi manuali, il che aumenta il rischio di errore umano e di incoerenza.
- Non riesci a rilevare la deviazione di configurazione. Questo ti porta falsamente a pensare che il sito di disaster recovery sia pronto prima di un incidente.

Vantaggi dell'adozione di questa best practice: la coerenza tra l'ambiente di disaster recovery e l'ambiente primario migliora notevolmente le probabilità della riuscita di un ripristino dopo un incidente e riduce il rischio di errore della procedura di ripristino.

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

Guida all'implementazione

Un approccio completo alla gestione della configurazione e alla preparazione al failover può aiutarti a verificare che il sito di disaster recovery sia costantemente aggiornato e pronto a subentrare in caso di errore del sito primario.

Per ottenere la coerenza tra l'ambiente primario e quello di disaster recovery (DR), verifica che le pipeline di distribuzione distribuiscano le applicazioni sia al sito primario che a quello di disaster recovery. Implementa le modifiche ai siti di disaster recovery dopo un adeguato periodo di valutazione (noto anche come implementazioni distribuite) per rilevare i problemi nel sito primario e arrestare l'implementazione prima che si diffondano. Implementa il monitoraggio per rilevare la deviazione della configurazione e tenere traccia delle modifiche e della conformità negli ambienti. Esegui la correzione automatica nel sito di disaster recovery per mantenerlo completamente coerente e pronto a subentrare in caso di incidente.

Passaggi dell'implementazione

1. Verifica che la Regione di disaster recovery contenga i servizi AWS e le funzionalità richieste per una corretta esecuzione del piano di disaster recovery.
2. Utilizza infrastructure as code (IaC). Mantieni accurati i modelli di configurazione dell'infrastruttura di produzione e dell'applicazione e applicali regolarmente all'ambiente di disaster recovery. AWS CloudFormation è in grado di rilevare le deviazioni tra ciò che i modelli CloudFormation specificano e ciò che viene effettivamente distribuito.
3. Configura le pipeline CI/CD per distribuire le applicazioni e gli aggiornamenti dell'infrastruttura in tutti gli ambienti, compresi i siti primari e di disaster recovery. Soluzioni CI/CD come AWS

CodePipeline possono automatizzare il processo di implementazione, riducendo il rischio di deviazione della configurazione.

4. Implementazioni distribuite tra gli ambienti primario e di disaster recovery. Questo approccio consente di distribuire e testare inizialmente gli aggiornamenti nell'ambiente primario, isolando così i problemi nel sito primario prima che vengano propagati al sito di disaster recovery. Questo approccio impedisce che i difetti vengano inviati contemporaneamente alla produzione e al sito di disaster recovery e mantiene l'integrità dell'ambiente di disaster recovery.
5. Monitora costantemente le configurazioni delle risorse sia nell'ambiente primario che in quello di disaster recovery. Soluzioni come AWS Config possono aiutare a far rispettare la conformità della configurazione e a rilevare eventuali deviazioni, contribuendo a mantenere configurazioni coerenti tra gli ambienti.
6. Implementa meccanismi di avviso per monitorare e notificare qualsiasi deviazione della configurazione o interruzione o ritardo nella replica dei dati.
7. Automatizza la correzione delle deviazioni di configurazione rilevate.
8. Pianifica audit periodici e controlli di conformità per verificare l'allineamento continuo tra le configurazioni primaria e di disaster recovery. Le revisioni periodiche aiutano a mantenere la conformità con le regole definite e a identificare eventuali discrepanze che devono essere risolte.
9. Verifica eventuali discordanze nella capacità allocata da AWS, in Service Quotas, nelle limitazioni (della larghezza di banda della rete) e nelle discrepanze di configurazione e versione.

Risorse

Best practice correlate:

- REL01-BP01 Consapevolezza su Service Quotas e vincoli di servizio
- REL01-BP02 Gestione delle Service Quotas in più account e Regioni
- REL01-BP04 Monitoraggio e gestione delle quote
- REL13-BP03 Esecuzione di test sull'implementazione del disaster recovery per convalidare l'implementazione

Documenti correlati:

- Remediating Noncompliant AWS Resources by Regole di AWS Config
- AWS Systems Manager Automation

- AWS CloudFormation: Detecting unmanaged configuration changes to stacks and resources
- AWS CloudFormation: Detect Drift on an Entire CloudFormation Stack
- 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?
- Remediating Noncompliant AWS Resources by Regole di AWS Config

Video correlati:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

Esempi correlati:

- Registro AWS CloudFormation
- Quota Monitor for AWS
- Implement automatic drift remediation for AWS CloudFormation using Amazon CloudWatch and AWS Lambda
- AWS Architecture Blog: serie sul disaster recovery
- Marketplace AWS: prodotti utilizzabili per il disaster recovery
- Automatizzazione di distribuzioni pratiche e sicure

REL13-BP05 Automatizzazione del ripristino

Implementa meccanismi di ripristino testati e automatizzati che siano affidabili, osservabili e riproducibili per ridurre il rischio e l'impatto aziendale di guasti ed errori.

Risultato desiderato: hai implementato un flusso di lavoro di automazione ben documentato, standardizzato e accuratamente testato per i processi di ripristino. L'automazione del ripristino corregge automaticamente i problemi secondari che comportano un basso rischio di perdita di dati o di indisponibilità. Puoi invocare rapidamente i processi di ripristino per gli incidenti gravi, osservare il comportamento della correzione durante il loro funzionamento e terminare i processi se osservi situazioni pericolose o errori.

Anti-pattern comuni:

- Dipendi da componenti o meccanismi che si trovano in uno stato non riuscito o danneggiato come parte del piano di ripristino.
- I processi di ripristino richiedono un intervento manuale, come l'accesso alla console (noto anche come ClickOps).
- Avvii le procedure di ripristino automaticamente in situazioni che presentano un rischio elevato di perdita o indisponibilità dei dati.
- Non includi un meccanismo per interrompere una procedura di ripristino (come un cavo Andon o un grande pulsante rosso di arresto) che non funziona o che comporta rischi aggiuntivi.

Vantaggi dell'adozione di questa best practice:

- Maggiore affidabilità, prevedibilità e coerenza delle operazioni di ripristino.
- Capacità di soddisfare obiettivi di ripristino più rigorosi, tra cui Obiettivo del tempo di ripristino (RTO) e Obiettivo del punto di ripristino (RPO).
- Riduzione della probabilità di non riuscita del ripristino durante un incidente.
- Riduzione del rischio di errori associati a processi di ripristino manuali, soggetti a errori umani.

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

Guida all'implementazione

Per implementare il ripristino automatizzato, è necessario un approccio completo che utilizzi i servizi AWS e le best practice. Per iniziare, identifica i componenti critici e i potenziali punti di errore nel carico di lavoro. Sviluppa processi automatizzati in grado di ripristinare i carichi di lavoro e i dati in caso di errori senza l'intervento umano.

Sviluppa l'automazione del ripristino utilizzando i principi infrastructure as code (IaC). In questo modo l'ambiente di ripristino è coerente con l'ambiente di origine e consente il controllo delle versioni dei processi di ripristino. Per orchestrare flussi di lavoro di ripristino complessi, valuta soluzioni come AWS Systems Manager Automations o AWS Step Functions.

L'automazione dei processi di ripristino offre vantaggi significativi e può aiutare a raggiungere più facilmente Obiettivo del tempo di ripristino (RTO) e Obiettivo del punto di ripristino (RPO). Tuttavia, si possono verificare situazioni impreviste che possono causare un esito negativo o creare nuovi

rischi, come tempo di inattività aggiuntivo e perdita di dati. Per ridurre questo rischio, occorre offrire la possibilità di interrompere rapidamente un'automazione dei ripristino in corso. Una volta interrotta, si può indagare e adottare misure correttive.

Per i carichi di lavoro supportati, valuta soluzioni come AWS Elastic Disaster Recovery (AWS DRS) per fornire un failover automatico. AWS DRS replica continuamente le macchine (compresi sistema operativo, configurazione dello stato del sistema, database, applicazioni e file) in un'area di gestione temporanea nell'Account AWS di destinazione e nella Regione preferita. Se si verifica un incidente, AWS DRS automatizza la conversione dei server replicati in carichi di lavoro completamente allocati nella Regione di ripristino su AWS.

La manutenzione e il miglioramento del ripristino automatico sono un processo continuo. Verifica e perfeziona continuamente le procedure di ripristino in base alle lezioni apprese e rimani aggiornato sui nuovi servizi e funzionalità AWS che possono migliorare le capacità di ripristino.

Passaggi dell'implementazione

1. Pianifica il ripristino automatico
    a. Esegui una revisione approfondita dell'architettura, dei componenti e delle dipendenze del carico di lavoro per identificare e pianificare i meccanismi di ripristino automatico. Classifica le dipendenze del carico di lavoro in dipendenze hard e soft. Le dipendenze hard sono quelle senza le quali il carico di lavoro non può funzionare e per le quali non è possibile fornire un sostituto. Le dipendenze soft sono quelle utilizzate abitualmente dal carico di lavoro, ma che possono essere sostituite da sistemi o processi sostitutivi temporanei o che possono essere gestite con una degradazione regolare.
    b. Stabilisci processi per identificare e recuperare i dati mancanti o danneggiati.
    c. Definisci i passaggi per confermare lo stato stazionario ripristinato dopo il completamento delle azioni di ripristino.
    d. Prendi in considerazione tutte le azioni necessarie per rendere il sistema ripristinato pronto per il servizio completo, come il pre-riscaldamento e la compilazione delle cache.
    e. Considera i problemi che si potrebbero verificare durante il processo di ripristino e come individuarli e correggerli.
    f. Considera gli scenari in cui il sito primario e il relativo piano di controllo (control-plane) non sono accessibili. Verifica che le azioni di ripristino possano essere eseguite in modo indipendente senza ricorso al sito primario. Considera soluzioni come Amazon Application Recovery Controller (ARC) per reindirizzare il traffico senza dover modificare manualmente i record DNS.
2. Sviluppa un processo di ripristino automatico

a. Implementa il rilevamento automatico dei guasti e meccanismi di failover per un ripristino automatico. Crea dashboard, ad esempio con Amazon CloudWatch, per segnalare lo stato di avanzamento e lo stato di integrità delle procedure di ripristino automatiche. Includi procedure per convalidare le operazioni di ripristino riuscite. Fornisci un meccanismo per interrompere un ripristino in corso.

b. Crea playbook come processo di fallback per guasti che non possono essere ripristinati automaticamente e prendi in considerazione il piano di disaster recovery.

c. Esegui il test dei processi di ripristino come descritto in REL13-BP03.

3. Preparati per il ripristino

a. Valuta lo stato del sito di ripristino e distribuisci in anticipo i componenti critici. Per ulteriori dettagli, consulta REL13-BP04.

b. Definisci ruoli, responsabilità e processi decisionali chiari per le operazioni di ripristino, coinvolgendo le parti interessate e i team dell'organizzazione.

c. Definisci le condizioni per avviare i processi di ripristino.

d. Crea un piano per invertire il processo di ripristino e tornare al sito primario, se richiesto o dopo che è stato considerato sicuro.

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
- REL13-BP02 Utilizzo di strategie di ripristino definite per conseguire gli obiettivi di ripristino
- REL13-BP03 Esecuzione di test sull'implementazione del disaster recovery per convalidare l'implementazione
- REL13-BP04 Gestione della deviazione di configurazione nel sito o nella Regione del disaster recovery

Documenti correlati:

- AWS Architecture Blog: serie sul disaster recovery
- Ripristino di emergenza dei carichi di lavoro su AWS: ripristino nel cloud (whitepaper di AWS)
- Orchestrate Disaster Recovery Automation using Amazon Route 53 ARC and AWS Step Functions

- Build AWS Systems Manager Automation runbooks using AWS CDK
- Marketplace AWS: prodotti utilizzabili per il disaster recovery
- AWS Systems Manager Automation
- AWS Elastic Disaster Recovery
- Using Elastic Disaster Recovery for Failover and Failback
- Risorse di AWS Elastic Disaster Recovery
- Partner APN: partner che possono assistere con il disaster recovery

Video correlati:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2022: AWS On Air ft. AWS Failback for AWS Elastic Disaster Recovery

Conclusioni

Che tu non abbia alcune 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 messo in discussione 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 sfruttare i consigli di progettazione, operativi e orientati al ripristino qui offerti, nonché la conoscenza e l'esperienza dei nostri AWS Solution Architect. Ci piacerebbe sentire la tua opinione, in particolare le tue storie di successo nel raggiungere alti livelli di disponibilità su. AWS Contatta il team del tuo account o usa Contattaci tramite il nostro sito Web.

Collaboratori

Hanno collaborato alla stesura del presente documento:

- Michael Fischer, Principal Solutions Architect, Amazon Web Services
- 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, 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:

- AWS Well-Architected Framework
- AWS Architecture Center

Revisioni del documento

Per ricevere una notifica sugli aggiornamenti del presente whitepaper, iscriviti al feed RSS.

Modifica	Descrizione	Data
Linee guida sulle best practice aggiornate	Le best practice sono state aggiornate con nuove linee guida nelle seguenti aree: REL 1, REL 2, REL 4, REL 6, REL 7, REL 8, REL 10, REL 12 e REL 13. Le linee guida sono state ampliate e chiarite per l'intero pilastro. Le linee guida per REL10-BP02 e REL12-BP03 sono state unite in altre best practice. Sono state aggiornate le risorse per l'intero pilastro.	6 novembre 2024
Linee guida sulle best practice aggiornate	Aggiornamenti minori alle best practice in REL 2, 4, 5, 6, 7 e 8.	27 giugno 2024
Linee guida sulle best practice aggiornate	Best practice aggiornate con nuove linee guida nelle seguenti aree: Progettazione delle interazioni in un sistema distribuito per prevenire i guasti, Progettazione delle interazioni in un sistema distribuito per mitigare o resistere ai guasti, Monitoraggio delle risorse del carico di lavoro, Progettazione del carico di lavoro per adattarlo ai cambiamenti della	6 dicembre 2023

	domanda, Implementazione delle modifiche e Test dell'affidabilità.	
Linee guida sulle best practice aggiornate	Best practice aggiornate con nuove linee guida nelle seguenti aree: Monitoraggio delle risorse del carico di lavoro e Progettazione del carico di lavoro per resistere ai guasti dei componenti.	3 ottobre 2023
Linee guida sulle best practice aggiornate	Best practice 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 resistere ai guasti e Monitoraggio delle risorse del carico di lavoro.	13 luglio 2023
Aggiornamento secondario	Rimozione del linguaggio non inclusivo.	13 aprile 2023
Aggiornamenti per il nuovo framework	Best practice aggiornate con prontuario e nuove best practice aggiunte.	10 aprile 2023
Aggiornamento del whitepaper	Best practice aggiornate con nuova guida all'implementazione.	15 dicembre 2022
Aggiornamenti minori	Numeri delle figure aggiornati e piccole modifiche in tutto il documento.	17 novembre 2022

Aggiornamento del whitepaper	Ampliamento delle best practice e aggiunta dei piani di miglioramento.	20 ottobre 2022
Aggiornamento del whitepaper	Sono state aggiunte due nuove best practice relative al pilastro dell'affidabilità nelle sezioni Utilizzo dell'isolamento dei guasti per proteggere il carico di lavoro e Progettare il carico di lavoro per resistere ai guasti dei componenti.	5 maggio 2022
Aggiornamento del whitepaper	Aggiornamento delle linee guida sul disaster recovery per includere Route 53 Application Recovery Controller. Aggiunta di riferimenti a DevOps Guru. Aggiornamento di diversi collegamenti alle Risorse e altre modifiche editoriali minori.	26 ottobre 2021
Aggiornamento secondario	Aggiunta di informazioni su AWS Fault Injection Service (AWS FIS).	15 marzo 2021
Aggiornamento secondario	Aggiornamento minore del testo.	4 gennaio 2021

Aggiornamento del whitepaper	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 la spiegazione delle differenze tra disponibilità e disaster recovery 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 disaster recovery); 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	7 dicembre 2020
Aggiornamento secondario	Appendice A aggiornata per aggiungere l'obiettivo di progettazione della disponibilità per AWS Lambda	27 ottobre 2020
Aggiornamento secondario	Appendice A aggiornata per aggiungere l'obiettivo di progettazione della disponibilità per AWS Global Accelerator	24 luglio 2020

Aggiornamenti per il nuovo framework	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, con lo spostamento di Ripristino automatico del guasto prima di Test delle procedure di ripristino, aggiornamento di diagrammi e formati per le equazioni, eliminazione delle sezioni Servizi chiave e integrazione dei riferimenti ai servizi AWS chiave nelle best practice.	8 luglio 2020
Aggiornamento secondario	Correzione di un collegamento danneggiato	1° ottobre 2019

Aggiornamento del whitepaper	Appendice A aggiornata	1 Aprile 2019
Aggiornamento del whitepaper	Aggiunta di consigli specifici sulla rete AWS Direct Connect e di obiettivi di progettazione del servizio aggiuntivi	1 settembre 2018
Aggiornamento del whitepaper	Aggiunte le sezioni Principi di progettazione e Gestione dei limiti. Collegamenti aggiornati, rimossa ambiguità della terminologia a monte/a valle e aggiunti riferimenti espliciti ai restanti argomenti del pilastro dell'affidabilità negli scenari di disponibilità.	1° giugno 2018
Aggiornamento del whitepaper	Modificata la soluzione DynamoDB su più regioni con le tabelle globali DynamoDB. Aggiunta degli obiettivi di progettazione del servizio	1 marzo 2018
Aggiornamenti minori	Correzione minore al calcolo della disponibilità per includere la disponibilità dell'applicazione	1° dicembre 2017
Aggiornamento del whitepaper	Aggiornamento per fornire indicazioni su progetti ad alta disponibilità, inclusi concetti, best practice ed esempi di implementazione.	1° novembre 2017
Pubblicazione iniziale	Pubblicazione del pilastro dell'affidabilità: Framework AWS Well-Architected.	1° novembre 2016

Note

I clienti sono responsabili della propria valutazione indipendente delle informazioni contenute nel presente documento. Questo documento: (a) è solo a scopo informativo, (b) rappresenta le offerte e le pratiche attuali di AWS prodotti, che sono soggette a modifiche senza preavviso, e (c) non crea alcun impegno o assicurazione da parte dei suoi affiliati, AWS fornitori o licenzianti. AWS i prodotti o i servizi sono forniti «così come sono» senza garanzie, dichiarazioni o condizioni di alcun tipo, esplicite o implicite. Le responsabilità e le responsabilità dei AWS propri clienti sono regolate da AWS accordi e il presente documento non fa parte di, né modifica, alcun accordo tra AWS e i suoi clienti.

AWS Glossario

Per la AWS terminologia più recente, consultate il AWS glossario nella sezione Reference.Glossario AWS