

# Principio di base dell'affidabilità

Canone di architettura AWS

*Aprile 2020*



## Avvisi

I clienti sono responsabili della propria valutazione autonoma delle informazioni contenute in questo documento. Questo documento: (a) è solo a scopo informativo, (b) mostra le offerte e le pratiche attuali dei prodotti AWS soggette a modifiche senza preavviso, e (c) non crea alcun impegno o garanzia da parte di AWS e dei suoi affiliati, fornitori o licenziatari. I prodotti o servizi AWS sono forniti "così come sono" senza garanzie, dichiarazioni o condizioni di alcun tipo, sia esplicite che implicite. Le responsabilità e gli obblighi di AWS verso i propri clienti sono disciplinati dagli accordi AWS e il presente documento non fa parte né modifica alcun accordo tra AWS e i suoi clienti.

© 2020, Amazon Web Services, Inc. o sue affiliate. Tutti i diritti riservati.

# Sommario

Introduzione .....	1
Affidabilità .....	1
Principi di progettazione .....	1
Definizioni .....	2
Comprendere le esigenze di disponibilità .....	7
Fondamenti.....	8
Gestione di quote e vincoli di servizio.....	8
Pianifica la topologia di rete.....	10
Architettura del carico di lavoro.....	15
Progettazione dell'architettura del servizio di carico di lavoro .....	16
Progettare interazioni in un sistema distribuito per evitare errori .....	19
Progetta interazioni in un sistema distribuito per mitigare o sopportare gli errori .....	22
Gestione delle modifiche.....	28
Monitoraggio delle risorse del carico di lavoro .....	28
Progetta il carico di lavoro per adattarti alle variazioni della domanda .....	33
Implementazione della modifica .....	35
Gestione dei guasti .....	40
Esegui il backup dei dati .....	40
Utilizzo dell'isolamento dei guasti per proteggere il tuo carico di lavoro.....	42
Progettare il carico di lavoro per resistere ai guasti dei componenti.....	48
Affidabilità dei test.....	52
Piano per il disaster recovery (DR).....	56
Implementazioni di esempio per obiettivi di disponibilità.....	60
Selezione delle dipendenze .....	60
Scenari a regione singola.....	61
Scenari multi regione .....	68
Conclusioni.....	77
Collaboratori.....	77

Approfondimenti .....	77
Revisioni del documento .....	78
Appendice A: disponibilità prevista per determinati servizi AWS .....	80

## Riassunto

Il focus di questo documento è il pilastro dell'affidabilità del [Canone di architettura AWS](#). Esso fornisce linee guida per aiutare i clienti ad applicare le best practice nella progettazione, distribuzione e manutenzione di ambienti Amazon Web Services (AWS).

# Introduzione

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

Il Canone di architettura AWS si basa su cinque principi:

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

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

Questo documento è rivolto a coloro che ricoprono ruoli tecnologici, ad esempio direttori tecnologici (CTO), architetti, sviluppatori e membri del team operativo. Dopo aver letto questo documento, ti sarà possibile comprendere le best practice e le strategie di AWS da utilizzare per progettare architetture cloud per l'affidabilità. Questo documento include dettagli di implementazione di alto livello e modelli architetturali, nonché riferimenti a risorse aggiuntive.

## Affidabilità

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

## Principi di progettazione

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

- **Ripristino automatico in caso di guasto:** monitorando un carico di lavoro per gli indicatori chiave di prestazioni (KPI), puoi attivare l'automazione quando una soglia viene superata. Questi KPI dovrebbero essere una misura del valore aziendale e non degli aspetti tecnici del funzionamento del servizio. Ciò consente la notifica e il tracciamento automatici degli errori, nonché 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 in locale, spesso vengono eseguiti test per dimostrare che il carico di lavoro funziona in uno scenario specifico. I test non vengono generalmente utilizzati per convalidare le strategie di recupero. Nel cloud, puoi testare il modo in cui il carico di lavoro incorre nell'errore e convalidare le procedure di ripristino. È possibile utilizzare l'automazione per simulare diversi errori o per ricreare scenari che in precedenza hanno condotto 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.
- **Dimensionamento 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 su molteplici risorse più piccole per garantire che non condividano un punto di errore comune.
- **Smetti di indovinare la capacità:** una causa comune di guasti nei carichi di lavoro in locale è la saturazione delle risorse, quando le richieste assegnate ad 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 dei limiti, ma alcune quote possono essere controllate e altre possono essere gestite (consulta [Gestisci quote e vincoli di servizio](#)).
- **Gestione del cambiamento nell'automazione:** le modifiche all'infrastruttura dovrebbero essere apportate utilizzando l'automazione. Le modifiche che devono essere gestite includono le modifiche all'automazione, che possono quindi essere monitorate e revisionate.

## Definizioni

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

- Fondamenti
- Architettura del carico di lavoro

- Gestione delle modifiche
- Gestione dei guasti

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

## La resilienza e i componenti dell'affidabilità

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

- **La resilienza** è la capacità di un carico di lavoro di ripristinarsi a seguito di interruzioni dell'infrastruttura o del servizio, acquisire dinamicamente le risorse di elaborazione 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 valutazioni di prontezza operativa (ORR) per confermare che le applicazioni sono pronte per le operazioni di produzione.
- Sicurezza, che include la prevenzione di danni ai dati o all'infrastruttura da parte di malintenzionati, che comprometterebbero la disponibilità. Ad esempio, crittografare i backup per garantire la sicurezza dei dati.
- Efficienza delle prestazioni, che include la progettazione di tassi massimi di richiesta e la riduzione al minimo delle latenze per il carico di lavoro.
- Ottimizzazione dei costi, che include compromessi quali spendere di più sulle istanze EC2 per ottenere stabilità statica oppure fare affidamento sulla scalabilità automatica quando è necessaria una maggiore capacità.

La resilienza è l'obiettivo principale di questo whitepaper.

Anche gli altri quattro fattori sono importanti e sono illustrati nei rispettivi pilastri del [Canone di architettura AWS](#). Ci occuperemo di loro nelle best practice, ma il nostro obiettivo è la resilienza.

## Disponibilità

La *disponibilità* (nota anche come *disponibilità dei servizi*) è un parametro comunemente utilizzato per misurare quantitativamente l'affidabilità.

- **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 quando occorre.

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

- $Availability = \frac{Available\ for\ Use\ Time}{Total\ Time}$
- Una percentuale di tempo di attività (ad esempio il 99,9%) in un periodo di tempo (in genere un anno)
- L'abbreviazione comune si riferisce solo alle aggiunte di "numeri di 9", ad esempio "cinque nove" si traduce in 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 primo punto. Tuttavia, questa si rivela spesso una scelta sbagliata perché gli utenti potrebbero effettivamente voler utilizzare il servizio durante questi periodi.

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

Disponibilità	Indisponibilità massima (per anno)	Categorie dell'applicazione
<a href="#">99%</a>	3 giorni 15 ore	Elaborazione batch, estrazione dati, trasferimento e caricamento di lavori
<a href="#">99,9%</a>	8 ore 45 minuti	Strumenti interni come gestione delle conoscenze e monitoraggio dei progetti
<a href="#">99,95%</a>	4 ore 22 minuti	Commercio online, punto di vendita
<a href="#">99,99%</a>	52 minuti	Carichi di lavoro di video e trasmissione
<a href="#">99,999%</a>	5 minuti	Transazioni bancomat, carichi di lavoro di telecomunicazione

**Calcolo della disponibilità con forti dipendenze.** Molti sistemi dipendono in misura elevata da altri sistemi, dove un'interruzione in un sistema dipendente si traduce direttamente in un'interruzione

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

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

$$99.99\% \times 99.99\% \times 99.99\% = 99.97\%$$

È quindi importante comprendere le dipendenze e i relativi obiettivi di progettazione di disponibilità quando si calcolano i propri.

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



$$Avail_{workload} = Avail_{MAX} - \left( (100\% - Avail_{dependency}) \times (100\% - Avail_{dependency}) \right)$$

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

Ma cosa succede se non conosco la disponibilità di una dipendenza?

**Calcolo della disponibilità delle dipendenze.** Alcune dipendenze forniscono linee guida sulla loro disponibilità, inclusi gli obiettivi di progettazione della disponibilità per molti servizi AWS (consulta [Appendice A: Progettazione per la disponibilità di servizi AWS selezionati](#)). 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 errori (MTBF)** e il **tempo medio di ripristino (MTTR)**. Una stima della disponibilità può essere stabilita da:

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

Ad esempio, se il MTBF è di 150 giorni e il MTTR è di 1 ora, la disponibilità stimata è del 99,97%.

Per ulteriori dettagli, consulta [questo documento \(Calcolo della disponibilità totale del sistema\)](#), che può aiutarti a calcolare la disponibilità.

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

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

## Recovery Time Objective (RTO) e Recovery Point Objective (RPO)

Questi termini sono più spesso associati al disaster recovery (DR), un insieme di obiettivi e strategie per ripristinare la disponibilità del carico di lavoro in caso di emergenza.

**Recovery Time Objective (RTO)** è il periodo di tempo complessivo durante il quale i componenti di un carico di lavoro possono trovarsi nella fase di ripristino e quindi non disponibili, prima di influire negativamente sulla missione o i processi aziendali dell'organizzazione.

**Recovery Point Objective (RPO)** è il periodo di tempo complessivo durante il quale i dati di un carico di lavoro possono essere non disponibili, prima di influire negativamente sulla missione o sui processi aziendali dell'organizzazione.

L'RPO deve essere inferiore all'RTO nel caso più comune, in cui la disponibilità dipende dai dati del carico di lavoro.

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.

## Comprendere le esigenze di disponibilità

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

### Suggerimento

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

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

Molti clienti AWS adottano un approccio analogo alla valutazione critica delle loro applicazioni e all'identificazione di componenti secondari con diverse esigenze di disponibilità. Gli obiettivi di progettazione della disponibilità vengono quindi adattati ai diversi aspetti e vengono compiuti gli appropriati sforzi di lavoro per progettare il sistema. AWS ha un'esperienza significativa nella progettazione di applicazioni con una serie di obiettivi di progettazione della disponibilità, inclusi servizi con una disponibilità del 99,999% o superiore. Gli AWS Solution Architects (SA) possono aiutarti a progettare in modo appropriato in base ai tuoi obiettivi di disponibilità. Il coinvolgimento di AWS nelle prime fasi del processo di progettazione migliora la nostra capacità di aiutarti a raggiungere i tuoi obiettivi di disponibilità. La pianificazione della disponibilità

non viene eseguita solo prima dell'avvio del carico di lavoro. Viene anche eseguita continuamente per perfezionare la progettazione man mano che acquisisci esperienza operativa, impari da eventi reali e superi errori di diversi tipi. È quindi possibile applicare lo sforzo di lavoro appropriato per migliorare l'implementazione.

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

## Fondamenti

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

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

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

- Gestione di quote e vincoli di servizio
- Effettuare il provisioning della topologia di rete

## Gestione di quote e vincoli di servizio

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

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

**Consapevolezza su quote e vincoli di servizio:** conosci le quote predefinite e le richieste di aumento delle quote per l'architettura del carico di lavoro. Inoltre, sai quali vincoli delle risorse, ad esempio disco o rete, sono potenzialmente influenti.

Service Quotas è un servizio AWS che ti aiuta a gestire le quote per oltre 100 servizi AWS da un'unica posizione. Oltre a cercare i valori delle quote, puoi anche richiedere e monitorare gli aumenti delle quote dalla console delle quote di servizio o tramite il kit SDK AWS. 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 sono indicate anche nella documentazione AWS di ciascun servizio. Consulta ad esempio [quote di Amazon VPC](#). I limiti di velocità sulle API con throttling vengono impostati all'interno del gateway API stesso configurando un piano di utilizzo. Altre restrizioni impostate come configurazione per i rispettivi servizi includono Provisioned IOPS, storage RDS allocato e allocazioni di volumi EBS. Amazon Elastic Compute Cloud (Amazon EC2) dispone di un proprio pannello di controllo sulle restrizioni dei servizi che consente di gestire l'istanza, Amazon Elastic Block Store (Amazon EBS) e i limiti degli indirizzi IP elastici. Se hai un caso d'uso in cui le quote di servizio influiscono sulle prestazioni della tua applicazione e non sono adattabili alle tue esigenze, contatta AWS Support per vedere se sono possibili mitigazioni.

**Gestione delle quote tra account e regioni:** se utilizzi più account AWS o regioni AWS, assicurati di richiedere le quote appropriate in tutti gli ambienti in cui vengono eseguiti i carichi di lavoro di produzione.

Le quote di servizio vengono monitorate per account. Salvo diversa indicazione, ogni quota è specifica alla regione AWS.

Oltre agli ambienti di produzione, gestisci anche le quote in tutti gli ambienti non di produzione applicabili, in modo che i test e lo sviluppo non siano ostacolati.

**Soddisfa le quote di servizio fisse e i vincoli tramite l'architettura:** considera le quote di servizio immutabili e le risorse fisiche e progetta per evitare che queste compromettano l'affidabilità.

Alcuni esempi includono larghezza di banda di rete, dimensioni di payload di AWS Lambda, velocità di ottimizzazione del throttling per API Gateway e connessioni utente simultanee a un cluster Amazon Redshift.

**Monitoraggio e gestione delle quote:** valuta il tuo utilizzo potenziale e aumenta le quote in modo appropriato per una crescita pianificata dell'utilizzo.

Per i servizi supportati, puoi gestire le quote configurando gli allarmi CloudWatch affinché monitorino l'utilizzo e ti inviino una notifica in caso di raggiungimento delle quote. Questi allarmi possono essere attivati dalle quote di servizio o da Trusted Advisor. Puoi anche utilizzare i filtri dei parametri su CloudWatch Logs per cercare ed estrarre modelli nei log al fine di determinare se l'utilizzo è vicino alle soglie delle quote.

**Automazione della gestione delle quote:** implementa strumenti che ti avvisino quando vengono raggiunte le soglie. Utilizzando le API delle quote di servizio, puoi automatizzare le richieste di aumento delle quote.

Se integri il tuo database di gestione della configurazione (CMDB) o il sistema di ticketing con le quote di servizio, puoi automatizzare il monitoraggio delle richieste di aumento delle quote e delle quote correnti. Oltre al kit SDK AWS, le quote di servizio offrono automazione utilizzando gli strumenti a riga di comando di AWS.

**Assicurati che esista un intervallo sufficiente tra le quote correnti e l'utilizzo massimo per gestire il failover:** quando una risorsa ha esito negativo, potrebbe comunque essere conteggiata rispetto alle quote finché non viene terminata correttamente. Assicurati che le quote coprano la sovrapposizione di tutte le risorse non riuscite con sostituzioni prima che le risorse non riuscite vengano terminate. Nel calcolo di questo intervallo dovresti considerare un errore nella zona di disponibilità.

## Risorse

### Video

- [AWS Live re:Inforce 2019 - Quote di servizio](#)

### Documentazione

- [Che cosa sono le quote di servizio?](#)
- [Quote di servizio AWS](#) (precedentemente definite restrizioni dei servizi)
- [Restrizioni dei servizi di Amazon EC2](#)
- [Controlli di best practice di AWS Trusted Advisor](#) (consulta la sezione **Restrizioni dei servizi**)
- [AWS Limit Monitor su AWS Answers](#)
- [AWS Marketplace: prodotti CMDB che aiutano a monitorare i limiti](#)
- [Partner APN: partner che possono aiutare nella gestione della configurazione](#)

## Pianifica la topologia di rete

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

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

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

**Utilizzo della connettività di rete ad alta disponibilità per gli endpoint pubblici del carico di lavoro:** questi endpoint e il relativo routing devono essere altamente disponibili. Per ottenere questo risultato, utilizza DNS ad alta disponibilità, reti di distribuzione di contenuti (CDN), API Gateway, bilanciamento del carico o proxy inversi.

Amazon Route 53, AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway ed Elastic Load Balancing (ELB) forniscono tutti endpoint pubblici altamente disponibili. Puoi anche scegliere di valutare le appliance software di AWS Marketplace per il bilanciamento del carico e il proxy.

I consumatori del servizio fornito dal carico di lavoro, che siano utenti finali o altri servizi, effettuano richieste su questi endpoint del servizio. Sono disponibili diverse risorse AWS che ti consentono di fornire endpoint ad alta disponibilità.

Elastic Load Balancing fornisce bilanciamento del carico tra le zone di disponibilità, esegue l'instradamento di livello 4 (TCP) o livello 7 (http/https) e si integra con AWS WAF e con AWS Auto Scaling per contribuire a creare un'infrastruttura con riparazione automatica e assorbire gli aumenti di traffico mentre rilascia risorse quando il traffico diminuisce.

Amazon Route 53 è un servizio DNS (Domain Name System) scalabile e altamente disponibile che collega le richieste degli utenti all'infrastruttura in esecuzione in AWS, come istanze Amazon EC2, sistemi di bilanciamento del carico Elastic Load Balancing o bucket Amazon S3, e può essere utilizzato anche per instradare gli utenti a un'infrastruttura esterna ad AWS.

AWS Global Accelerator è un servizio a livello di rete che puoi utilizzare per indirizzare il traffico verso endpoint ottimali sulla rete globale AWS.

Gli attacchi DDoS (Distributed Denial of Service) rischiano di chiudere il traffico legittimo e ridurre la disponibilità per gli utenti. AWS Shield fornisce protezione automatica contro questi attacchi senza costi aggiuntivi per gli endpoint del servizio AWS sul tuo carico di lavoro. Puoi potenziare queste caratteristiche con appliance virtuali dei partner APN e di AWS Marketplace per soddisfare le tue esigenze.

**Provisioning di connettività ridondante tra reti private nel cloud e ambienti in locale:** utilizza più connessioni AWS Direct Connect (DX) o tunnel VPN tra reti private distribuite separatamente. Utilizza più ubicazioni DX per un'elevata disponibilità. Se utilizzi più regioni AWS, garantisci la ridondanza in almeno due di esse. È possibile valutare le appliance AWS Marketplace che terminano le VPN. Se utilizzi appliance di AWS Marketplace, distribuisci le istanze ridondanti per la disponibilità elevata in diverse zone di disponibilità.

Direct Connect è un servizio cloud che semplifica la creazione di una connessione di rete dedicata dall'ambiente in locale ad AWS. Utilizzando il gateway Direct Connect, il data center in locale può essere collegato a più VPC AWS distribuiti in più regioni AWS.

Questa ridondanza risolve possibili errori che condizionano la resilienza della connettività:

- In che modo sarai resiliente agli errori nella topologia?
- Cosa succede se configuri qualcosa in modo errato e rimuovi la connettività?
- Sarai in grado di gestire un aumento imprevisto del traffico e dell'utilizzo dei tuoi servizi?
- Sarai in grado di assorbire un tentativo di attacco DDoS (Distributed Denial of Service)?

Quando si connette il VPC al data center in locale tramite VPN, si devono considerare i requisiti di resilienza e larghezza di banda necessari quando si seleziona la dimensione del fornitore e dell'istanza su cui è necessario eseguire l'appliance. Se si utilizza un'appliance VPN non resiliente nella sua implementazione, è necessario disporre di una connessione ridondante tramite una seconda appliance. Per tutti questi scenari, è necessario definire un orario accettabile per il ripristino e il test per garantire che sia possibile soddisfare tali requisiti.

Se scegli di connettere il VPC al data center utilizzando una connessione Direct Connect e hai bisogno che questa connessione sia altamente disponibile, predisponi connessioni DX ridondanti da ogni data center. La connessione ridondante dovrebbe utilizzare una seconda connessione DX da una posizione diversa rispetto alla prima. Se disponi di più data center, assicurati che le connessioni terminino in posizioni diverse. Utilizza il [kit di strumenti di resilienza di Direct Connect](#) per semplificare la configurazione.

Se scegli di eseguire il failover sul VPN su Internet utilizzando AWS VPN, è importante capire che supporta fino a 1,25 Gbps di throughput per tunnel VPN, ma non supporta Equal Cost Multi Path (ECMP) per il traffico in uscita nel caso di più tunnel VPN gestiti da AWS che terminano sullo stesso gateway virtuale. Non è consigliabile utilizzare AWS Managed VPN come backup per le connessioni Direct Connect, a meno che non sia possibile tollerare velocità inferiori a 1 Gbps durante il failover.

Puoi anche utilizzare gli endpoint VPC per connettere privatamente il tuo VPC ai servizi AWS supportati e ai servizi endpoint VPC basati su AWS PrivateLink senza dover attraversare la rete Internet pubblica. Gli endpoint sono dispositivi virtuali. Sono componenti VPC a scalabilità orizzontale, ridondanti e ad alta disponibilità. Consentono la comunicazione tra le istanze nel VPC e i servizi senza imporre rischi di disponibilità o vincoli di larghezza di banda sul traffico di rete.

**Assicurati che gli account di allocazione delle sottoreti IP siano espandibili e disponibili:** gli intervalli di indirizzi IP di Amazon VPC devono essere abbastanza grandi da soddisfare i requisiti del carico di lavoro, tra cui la fattorizzazione nella futura espansione e l'allocazione di indirizzi IP alle sottoreti nelle zone di disponibilità. Sono inclusi sistemi di bilanciamento del carico, istanze EC2 e applicazioni basate su container.

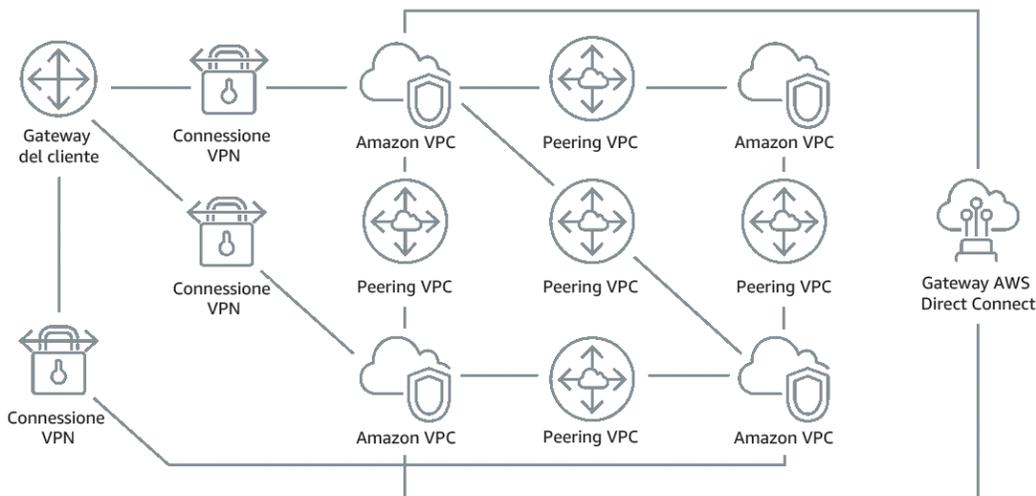
Quando pianifichi la topologia di rete, la prima fase consiste nel definire lo spazio stesso degli indirizzi IP. Gli intervalli di indirizzi IP privati (secondo le linee guida RFC 1918) dovrebbero essere allocati per ogni VPC. Nell'ambito di questo processo, soddisfa i seguenti requisiti:

- Lascia spazi per indirizzi IP per più di un VPC per Regione.
- All'interno di un VPC, lascia spazio per più sottoreti che coprano più zone di disponibilità.
- Lascia sempre spazio per un blocco CIDR inutilizzato all'interno di un VPC per un'espansione futura.
- Assicurati che sia disponibile spazio per gli indirizzi IP, al fine di soddisfare le esigenze di qualsiasi parco istanze EC2 transitorio che puoi utilizzare, ad esempio parchi istanze Spot per il machine learning, cluster Amazon EMR o cluster Amazon Redshift.
- Tieni presente che i primi quattro indirizzi IP e l'ultimo indirizzo IP in ogni blocco CIDR della sottorete sono riservati e non disponibili per l'uso.

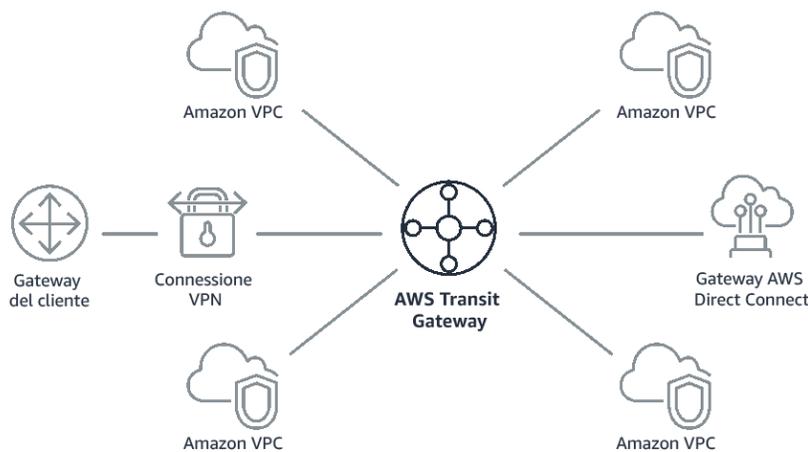
È consigliabile pianificare la distribuzione di blocchi CIDR VPC di grandi dimensioni. Tieni presente che il blocco CIDR VPC iniziale allocato al VPC non può essere modificato o eliminato, ma puoi aggiungere ulteriori blocchi CIDR non sovrapposti al VPC. I CIDR IPv4 della sottorete non possono essere modificati, mentre ciò è possibile con i CIDR IPv6. Tieni presente che la distribuzione del VPC più grande possibile (/16) genera oltre 65.000 indirizzi IP. Solo nello spazio degli indirizzi IP di base 10.x.x.x potresti effettuare il provisioning di 255 VPC di questo tipo. Pertanto, dovresti peccare per eccesso piuttosto che per difetto per semplificare la gestione dei VPC.

**Preferisci topologie hub-and-spoke rispetto a mesh multi-a-molti:** se più di due spazi di indirizzi di rete (ad esempio, VPC e reti in locale) sono connessi tramite peering VPC, AWS Direct Connect o VPN, utilizza un modello hub-and-spoke, come quelli forniti da AWS Transit Gateway.

Se disponi solo di due reti di questo tipo, puoi semplicemente connetterle tra loro, tuttavia, man mano che il numero di reti cresce, la complessità di tali connessioni mesh diventa insostenibile. AWS Transit Gateway offre un modello hub-and-spoke di facile manutenzione, consentendo l'instradamento del traffico su più reti.



*Senza AWS Transit Gateway: è necessario eseguire il peering tra VPC Amazon e tra ciascuna ubicazione locale utilizzando una connessione VPN, che può diventare complessa man mano che si ricalibra.*



*Con AWS Transit Gateway: è sufficiente connettere ciascun VPC o VPN ad AWS Transit Gateway e instradare il traffico da e verso ogni VPC o VPN.*

**Applica intervalli di indirizzi IP privati non sovrapposti in tutti gli spazi di indirizzi privati in cui sono connessi:** gli intervalli di indirizzi IP di ogni VPC non devono sovrapporsi quando collegati in peering o connessi tramite VPN. Analogamente, è necessario evitare conflitti di indirizzi IP tra un VPC e ambienti in locale o con altri provider di servizi cloud utilizzati. Bisogna inoltre disporre di un modo per allocare gli intervalli di indirizzi IP privati quando necessario.

Un sistema di gestione degli indirizzi IP (IPAM) può aiutarti in questo. Su AWS Marketplace sono disponibili diversi IPAM.

## Risorse

### Video

- [AWS re:Invent 2018: una progettazione VPC avanzata e nuove funzionalità per Amazon VPC \(NET303\)](#)
- [AWS re:Invent 2019: architetture di riferimento di AWS Transit Gateway per molti VPC \(NET406-R1\)](#)

### Documentazione

- [Cos'è un Transit Gateway?](#)
- [Cos'è Amazon VPC?](#)
- [Lavorare con gateway Direct Connect](#)
- [Utilizzo del kit di strumenti di resilienza di Direct Connect per iniziare](#)
- [Connettività di rete ad elevata disponibilità per data center multipli](#)
- [Cos'è AWS Global Accelerator?](#)
- [Utilizzo di connessioni VPN da sito a sito ridondanti per fornire il failover](#)
- [Endpoint VPC e servizi di endpoint VPC \(AWS PrivateLink\)](#)
- [Whitepaper sulle opzioni di connettività di Amazon Virtual Private Cloud](#)
- [AWS Marketplace per infrastruttura di rete](#)
- [Partner APN: partner che possono aiutarti a pianificare le tue reti](#)

## Architettura del carico di lavoro

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

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

- Implementazione dell'architettura del servizio appropriata
- Progettazione di un software in un sistema distribuito per evitare guasti
- Progettazione di un software in un sistema distribuito per mitigare gli errori

## Progettazione dell'architettura del servizio di carico di lavoro

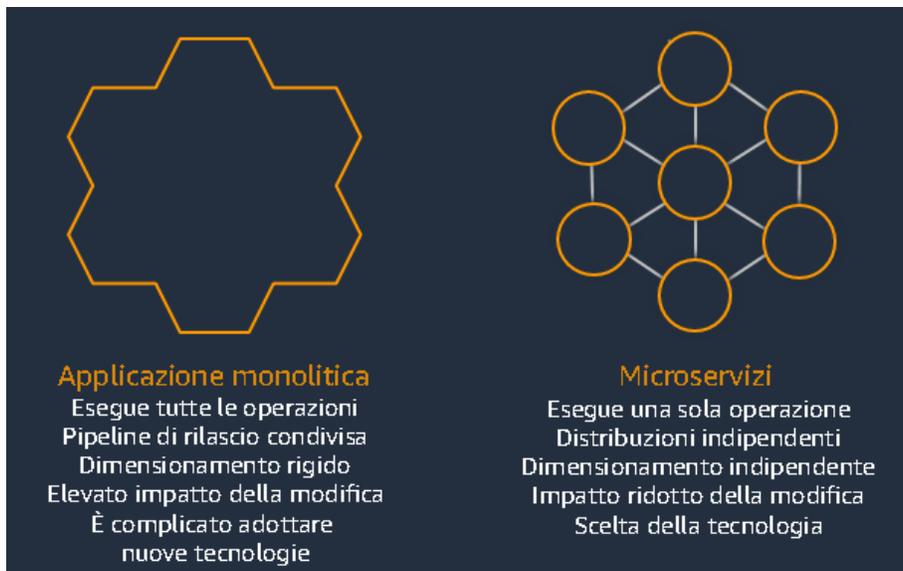
Creazione di carichi di lavoro altamente scalabili e affidabili utilizzando un'architettura orientata ai servizi (SOA) o un'architettura di microservizi. L'architettura orientata ai servizi (SOA) è la pratica di rendere i componenti software riutilizzabili tramite interfacce di servizio. L'architettura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

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

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

**Scegliere come segmentare il carico di lavoro:** l'architettura monolitica dovrebbe essere evitata. Al contrario, è consigliabile scegliere tra SOA e microservizi. Quando effettui ogni scelta, bilancia i vantaggi con le complessità: ciò che è giusto per un nuovo prodotto che corre al primo lancio è diverso da ciò che necessita un carico di lavoro creato per ricalibrare le risorse dall'inizio. I vantaggi dell'utilizzo di segmenti più piccoli includono maggiore agilità, flessibilità organizzativa e scalabilità. Le complessità includono un eventuale aumento della latenza, un debug più complesso e un maggiore carico operativo.

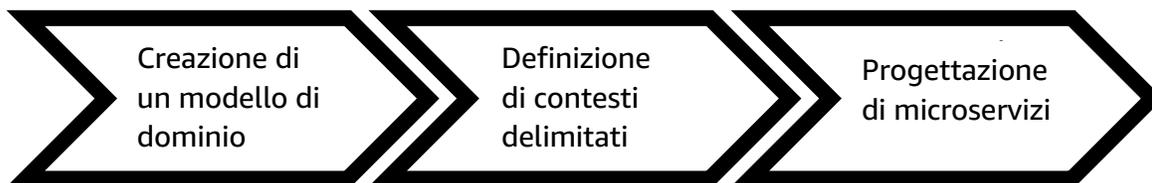
Anche se scegli di iniziare con un'architettura monolitica, devi assicurarti che sia modulare e abbia la capacità di evolversi in SOA o microservizi man mano che il prodotto si dimensiona con l'adozione dell'utente. La SOA e i microservizi offrono rispettivamente una segmentazione più piccola, preferita come architettura moderna scalabile e affidabile, ma ci sono [compromessi da considerare](#) soprattutto quando si distribuisce un'architettura di microservizi. Uno di questi 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. AWS X-Ray può essere utilizzato per risolvere questo problema. Un altro effetto da considerare è l'aumento della complessità operativa man mano che si moltiplica il numero di applicazioni che gestisci, che richiede la distribuzione di più componenti di indipendenza.



*Architettura monolitica e architettura di microservizi*

**Creazione di servizi incentrati su domini e funzionalità aziendali specifici:** la SOA crea servizi con funzioni ben delineate definite dalle esigenze aziendali. I microservizi utilizzano modelli di dominio e contesto delimitato per restringere ulteriormente questa operazione, in modo che ogni servizio esegua *una sola operazione*. Focalizzarsi su funzionalità specifiche consente di differenziare i requisiti di affidabilità dei diversi servizi e mirare agli investimenti in modo più specifico. Un problema aziendale conciso e un piccolo team associato a ciascun servizio facilitano il dimensionamento organizzativo.

Nella progettazione di un'architettura di microservizi, è utile utilizzare Domain-Driven Design (DDD) per modellare il problema aziendale utilizzando le entità. Ad esempio, le entità Amazon.com possono includere pacchetti, consegna, pianificazione, prezzo, sconto e valuta. Quindi il modello viene ulteriormente suddiviso in modelli più piccoli utilizzando [Contesto delimitato](#), dove le entità che condividono caratteristiche e attributi simili vengono raggruppate insieme. Pertanto, utilizzando il pacchetto di esempio di Amazon, la consegna e la pianificazione sarebbero parte del contesto di spedizione, mentre il prezzo, lo sconto e la valuta fanno parte del contesto dei prezzi. Con il modello diviso in contesti, emerge un modello su come delimitare i microservizi.



**Fornitura di contratti di servizio per API:** i contratti di servizio sono accordi documentati tra i team sull'integrazione dei servizi e includono una definizione API leggibile dal computer, limiti

di velocità e aspettative di prestazioni. Una strategia di controllo delle versioni consente ai client di continuare a utilizzare l'API esistente e migrare le applicazioni all'API più recente quando sono pronte. La distribuzione può avvenire in qualsiasi momento, purché il contratto non venga violato. Il team del fornitore di servizi può utilizzare lo stack tecnologico scelto per soddisfare il contratto API. Analogamente, l'utente del servizio può utilizzare la propria tecnologia.

I microservizi portano il concetto di SOA al punto di creare servizi con un set minimo di funzionalità. Ogni servizio pubblica un'API e obiettivi di progettazione, limiti e altre considerazioni per l'utilizzo del servizio. Ciò stabilisce un "contratto" con le applicazioni chiamanti. Questo comporta tre vantaggi principali:

- Il servizio possiede un problema aziendale conciso e un piccolo team a cui appartiene il problema aziendale. Questo consente un miglior ridimensionamento organizzativo.
- Il team può effettuare la distribuzione in qualsiasi momento purché soddisfi le proprie API e altri requisiti "contrattuali"
- Il team può utilizzare qualsiasi stack tecnologico a condizione che soddisfi le proprie API e altri requisiti di "contratto".

Amazon API Gateway è un servizio completamente gestito che semplifica agli sviluppatori la creazione, la pubblicazione, la manutenzione, il monitoraggio e la protezione delle API su qualsiasi scala. Gestisce tutte le attività coinvolte nell'accettazione e nell'elaborazione di fino a centinaia di migliaia di chiamate API simultanee, tra cui la gestione del traffico, il controllo delle autorizzazioni e degli accessi, il monitoraggio e la gestione delle versioni delle API. Utilizzando OpenAPI Specification (OAS), precedentemente noto come Swagger Specification, è possibile definire il contratto API e importarlo in API Gateway. Con API Gateway, puoi eseguire la versione e la distribuzione delle API.

## Risorse

### Documentazione

- Amazon API Gateway: [configurazione di un'API REST con OpenAPI](#)
- [Implementazione di microservizi in AWS](#)
- [Microservizi in AWS](#)

### Collegamenti esterni

- [Microservizi: una definizione di questo nuovo termine di architettura](#)
- [I compromessi dei microservizi](#)
- [Contesto delimitato](#) (un modello centrale in Domain-Driven Design)

## Progettare interazioni in un sistema distribuito per evitare errori

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

**Identificazione del tipo di sistema distribuito necessario:** i sistemi distribuiti hard real-time richiedono risposte che devono essere fornite in modo sincrono e rapido, mentre i sistemi soft real-time hanno una finestra temporale più generosa di minuti o più per la risposta. I sistemi offline gestiscono le risposte tramite elaborazione in batch o asincrona. I sistemi distribuiti hard real-time hanno i requisiti di affidabilità più severi.

Le [difficoltà maggiori con i sistemi distribuiti](#) riguardano i sistemi distribuiti hard real-time, noti anche come servizi di richiesta/risposta. La difficoltà sta nel fatto che le richieste arrivino in modo imprevedibile e le risposte debbano essere fornite rapidamente (ad esempio, il cliente è attivamente in attesa della risposta). Alcuni esempi includono i server Web front-end, la pipeline degli ordini, le transazioni con carte di credito, ogni API AWS e la telefonia.

**Implementazione delle dipendenze con accoppiamento debole:** le dipendenze come sistemi di accodamento, sistemi di streaming, flussi di lavoro e sistemi di bilanciamento del carico hanno 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.

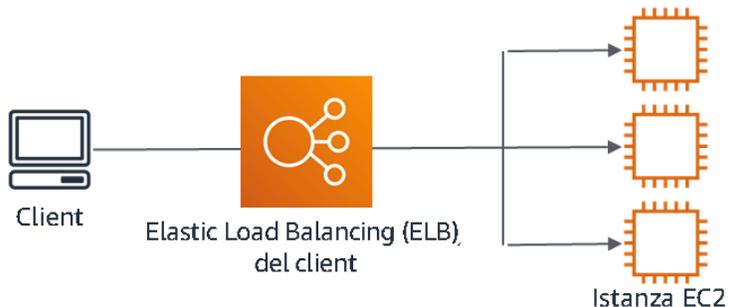
Se i cambiamenti apportati a un componente forzano la modifica anche di altri componenti basati sullo stesso, allora si parla di accoppiamento *stretto*. L'accoppiamento *debole* interrompe questa dipendenza, in modo che i componenti dipendenti debbano conoscere solo l'interfaccia con versione e pubblicata. 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 aggiungere liberamente ulteriore codice o caratteristiche a un componente, riducendo al minimo i rischi per i componenti che dipendono da esso. Inoltre, la scalabilità è migliorata in quanto è possibile dimensionare orizzontalmente o persino modificare l'implementazione sottostante della dipendenza.

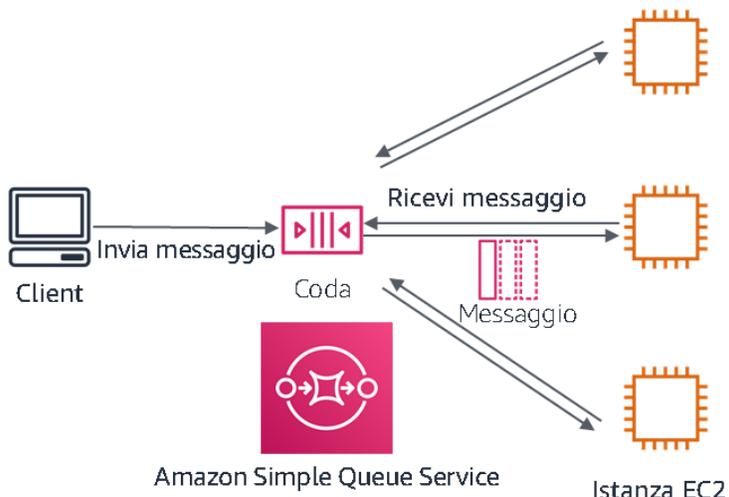
Per migliorare ulteriormente la resilienza tramite accoppiamento debole, rendi le interazioni dei componenti asincrone laddove possibile. Questo modello è idoneo a qualsiasi interazione che non richieda una risposta immediata e laddove la conferma della registrazione di una richiesta sia sufficiente.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



**Accoppiamento stretto (sincrono)**  
 Un errore nell'istanza EC2 influisce direttamente sul client



**Accoppiamento debole (sincrono)**  
 ELB instrada il traffico solo alle istanze EC2 integre, mitigando un guasto su una qualsiasi di esse



**Accoppiamento debole (asincrono)**  
 I "lavoratori" dell'istanza EC2 raccolgono le richieste come messaggi in coda. Se si verificano errori, il messaggio rimane e un altro lavoratore può elaborarlo (oppure può ricevere un'elaborazione speciale in una coda DLQ)  
 Se il tasso di richiesta supera la capacità di elaborarle, le richieste possono comunque essere soddisfatte e vengono memorizzate nell'unità di coda elaborata.

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

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

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

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

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

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

## Risorse

### Video

- [AWS re:Invent 2019: passare alle architetture basate sugli eventi \(SVS308\)](#)
- [AWS re:Invent 2018: circuiti chiusi e menti aperte: come controllare i sistemi, grandi e piccoli ARC337](#) (include accoppiamento debole, lavoro costante e stabilità statica)
- [Summit AWS di New York 2019: introduzione alle architetture basate su eventi e ad Amazon EventBridge \(MAD205\)](#) (si parla di EventBridge, SQS, SNS)

## Documentazione

- [Servizi AWS che pubblicano parametri CloudWatch](#)
- [Cos'è Amazon Simple Queue Service?](#)
- Amazon EC2: [Garantire l'idempotenza](#)
- Amazon Builders' Library: [sfide con i sistemi distribuiti](#)
- [Soluzione di registrazione centralizzata](#)
- [AWS Marketplace: prodotti che possono essere utilizzati per il monitoraggio e gli avvisi](#)
- [Partner APN: partner che possono aiutarti con il monitoraggio e l'accesso](#)

## Progetta interazioni in un sistema distribuito per mitigare o sopportare gli errori

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

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

**Implementazione di un degrado elegante per trasformare le dipendenze forti applicabili in dipendenze deboli:** quando le dipendenze di un componente non sono integre, il componente stesso può comunque funzionare, anche se in modo degradato. Ad esempio, quando una chiamata di dipendenza non riesce, utilizza invece una risposta statica predeterminata.

Considera un servizio B chiamato dal servizio A che a sua volta chiama il servizio C.



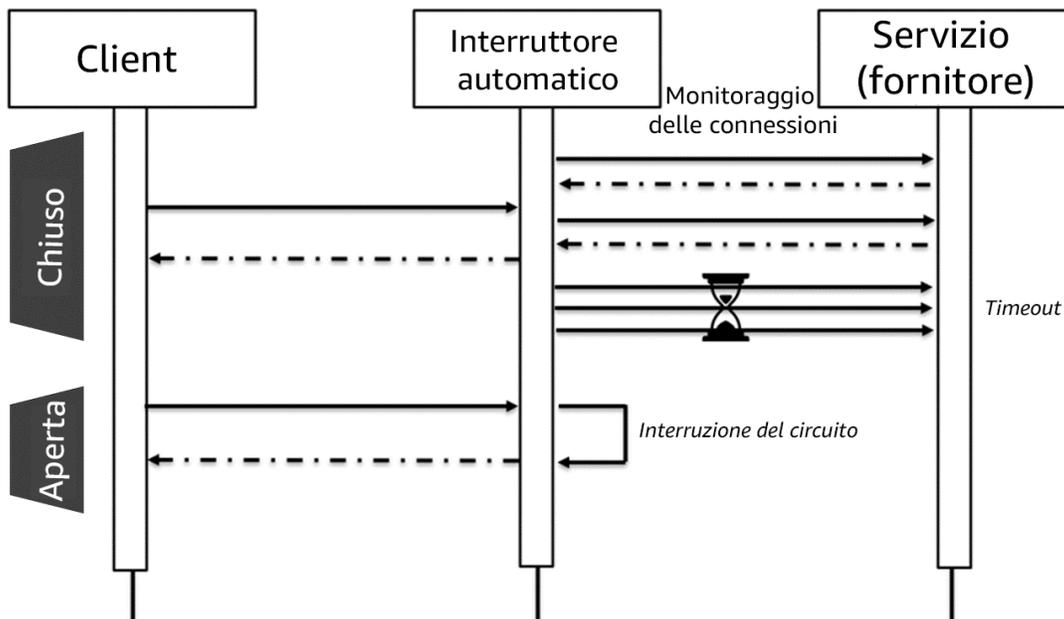
*Il servizio C ha esito negativo quando viene chiamato dal servizio B. Il servizio B restituisce una risposta degradata al servizio A.*

Quando il servizio B chiama il servizio C, ha ricevuto da quest'ultimo un errore o un timeout. Il servizio B, senza una risposta dal servizio C (e dai dati che contiene) restituisce invece ciò

che può. Questo può essere l'ultimo valore buono memorizzato nella cache oppure il servizio B può sostituire una risposta statica predeterminata a ciò che avrebbe ricevuto dal servizio C. Può quindi restituire una risposta degradata all'intermediario, il servizio A. Senza questa risposta statica, l'errore nel servizio C si propagherebbe attraverso il servizio B fino al servizio A, causando una perdita di disponibilità.

Secondo il fattore moltiplicativo nell'equazione di disponibilità per le dipendenze forti (consulta [Calcolo della disponibilità con dipendenze forti](#)), qualsiasi calo della disponibilità di C influisce notevolmente sulla disponibilità effettiva di B. Restituendo il servizio di risposta statica B mitiga l'errore in C e, sebbene degradato, rende la disponibilità del servizio C simile alla disponibilità del 100% (presupponendo che restituisca in modo affidabile la risposta statica in condizioni di errore). La risposta statica è una semplice alternativa alla restituzione di un errore e non è un tentativo di ricalcolare la risposta utilizzando metodi diversi. Tali tentativi a livello di un meccanismo completamente diverso che cercano di ottenere lo stesso risultato sono chiamati comportamento di fallback e sono un anti-modello da evitare.

Un altro esempio di degrado elegante è il *modello dell'interruttore*. Le strategie di ripetizione devono essere utilizzate quando l'errore è transitorio. Quando non è il caso e l'operazione potrebbe non riuscire, il modello dell'interruttore impedisce al client di eseguire una richiesta che potrebbe non riuscire. Quando le richieste vengono elaborate normalmente, l'interruttore viene chiuso e le richieste scorrono. Quando il sistema remoto inizia a restituire errori o presenta una latenza elevata, l'interruttore si apre e la dipendenza viene ignorata o i risultati vengono sostituiti con risposte ottenute più semplicemente, ma meno complete (che potrebbero essere semplicemente una cache di risposta). Periodicamente, il sistema tenta di chiamare la dipendenza per determinare se è stata ripristinata. In questo caso, l'interruttore viene chiuso.



*L'interruttore che mostra gli stati chiusi e aperti.*

Oltre agli stati chiusi e aperti mostrati nel diagramma, dopo un periodo di tempo configurabile nello stato aperto, l'interruttore può passare allo stato semiaperto. In questo stato, tenta periodicamente di chiamare il servizio a una velocità molto inferiore rispetto al normale. Questa indagine viene utilizzata per controllare lo stato del servizio. Dopo un certo numero di successi nello stato semiaperto, l'interruttore passa allo stato chiuso e le normali richieste riprendono.

**Richieste di throttling:** si tratta di un modello di mitigazione per rispondere a un aumento imprevisto della domanda. Alcune richieste vengono soddisfatte, ma quelle che superano un limite definito vengono rifiutate e restituiscono un messaggio che indica che sono state sottoposte a throttling. L'aspettativa per i client è che si ritirino e abbandonino la richiesta o riprovino a una velocità più lenta.

I servizi devono essere progettati per una capacità nota di richieste che ogni nodo o cella è in grado di elaborare. Ciò può essere stabilito mediante test di carico. È quindi necessario tenere traccia del tasso di arrivo delle richieste e se il tasso di arrivo temporaneo supera questo limite, la risposta appropriata è segnalare che la richiesta è stata limitata. Ciò consente all'utente di riprovare, potenzialmente su un nodo/cella diverso che potrebbe avere capacità disponibile. Amazon API Gateway fornisce metodi per limitare le richieste. Amazon SQS e Amazon Kinesis possono eseguire il buffer delle richieste, livellando il tasso di richiesta e alleggerendo la necessità di throttling per le richieste che possono essere gestite in modo asincrono.

**Controllo e limitazione delle chiamate di ripetizione dei tentativi:** utilizza il backoff esponenziale per eseguire nuovi tentativi dopo intervalli progressivamente più lunghi. Introduci il jitter per randomizzare gli intervalli di ripetizione e limitare il numero massimo di tentativi.

I componenti tipici di un sistema software distribuito includono server, sistemi di bilanciamento del carico, database e server DNS. Durante il funzionamento, e sempre soggetti ad anomalie, uno qualsiasi tra questi componenti può iniziare a generare errori. La tecnica predefinita per gestire gli errori consiste nell'implementare nuovi tentativi lato client. Questa tecnica aumenta l'affidabilità e la disponibilità dell'applicazione. Tuttavia, su vasta scala, e se i client cercano di eseguire nuovamente l'operazione non riuscita non appena si verifica un errore, la rete può rapidamente diventare satura di richieste nuove e ritirate, ciascuna in competizione per la larghezza di banda di rete. Ciò può causare una *tempesta di ripetizione dei tentativi*, che ridurrà la disponibilità del servizio. Questo modello potrebbe continuare finché non si verifica un errore completo del sistema.

Per evitare tali scenari, dovrebbero essere utilizzati [algoritmi di backoff](#) quali il **backoff esponenziale** comune. Gli algoritmi di backoff esponenziale riducono gradualmente la velocità con cui vengono eseguiti i nuovi tentativi, evitando così la congestione della rete.

Molti SDK e librerie software, inclusi quelli di AWS, implementano una versione di questi algoritmi. Tuttavia, non dare mai per scontato che esista un algoritmo di backoff, esegui sempre test e verificane la presenza.

Il backoff semplice da solo non è sufficiente perché nei sistemi distribuiti tutti i client possono eseguire simultaneamente il backoff, creando cluster di chiamate ripetute. Nel suo post di blog [Backoff esponenziale e Jitter](#), Marc Better spiega come modificare la funzione `wait()` nel backoff esponenziale per evitare cluster di chiamate di ripetizione dei tentativi. La soluzione consiste nell'aggiungere **jitter** nella funzione `wait()`. Per evitare di eseguire nuovi tentativi per troppo tempo, le implementazioni dovrebbero limitare il backoff a un valore massimo.

Infine, è importante configurare un numero massimo di tentativi o di tempo trascorso, dopo il quale i nuovi tentativi semplicemente falliranno. Gli SDK AWS lo implementano per impostazione predefinita e può essere configurato. Per i servizi più bassi nello stack, un limite massimo di nuovi tentativi pari a zero o uno limiterà il rischio, ma sarà comunque efficace poiché i nuovi tentativi vengono delegati a servizi più elevati nello stack.

**Guasti veloci e limitazioni delle code:** se il carico di lavoro non è in grado di rispondere correttamente a una richiesta, ha esito negativo rapidamente. Ciò consente il rilascio delle risorse associate a una richiesta e permette al servizio di recuperare se le risorse sono in esaurimento. Se il carico di lavoro è in grado di rispondere correttamente, ma la frequenza delle richieste è troppo elevata, utilizza una coda per eseguire il buffer delle richieste. Tuttavia, non consentire code lunghe che possono comportare l'elaborazione di richieste obsolete a cui il client ha già rinunciato.

Questa best practice si applica al lato server, o ricevitore, della richiesta.

Tieni presente che le code possono essere create a più livelli di un sistema e possono compromettere notevolmente la possibilità di recuperare rapidamente quando le richieste obsolete (che non necessitano più di una risposta) vengono elaborate prima di richieste più recenti che hanno bisogno di una risposta. Fai attenzione ai luoghi in cui sono presenti code. Spesso si nascondono nei flussi di lavoro o nel lavoro registrato in un database.

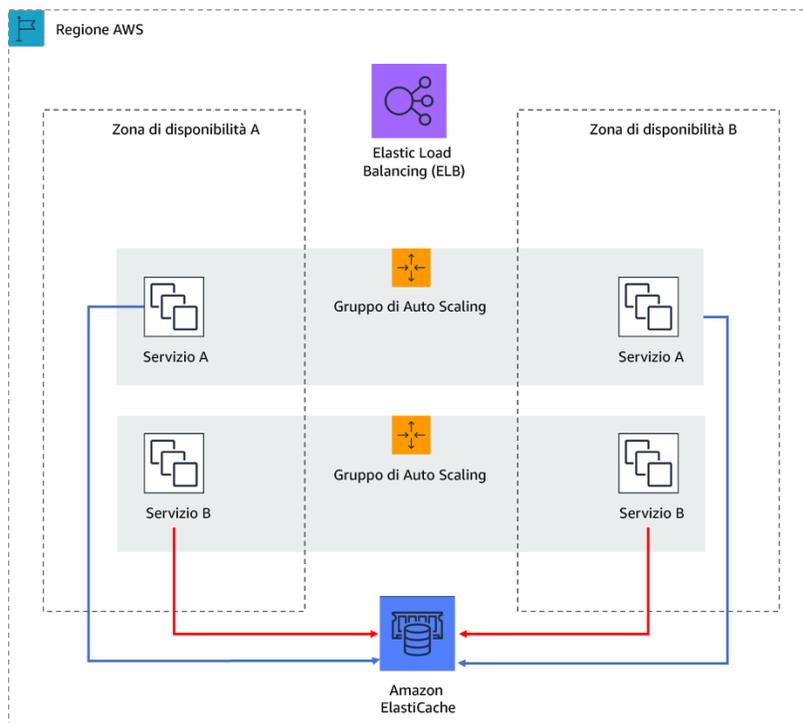
**Impostazione dei timeout del client:** imposta i timeout in modo appropriato, verificali sistematicamente e non fare affidamento sui valori predefiniti, poiché per impostazione sono generalmente troppo alti

Questa best practice si applica al lato client, o al mittente, della richiesta.

Imposta sia un timeout di connessione che un timeout di richiesta su qualsiasi chiamata remota e, generalmente, su qualsiasi chiamata tra i processi. Molti framework offrono funzionalità di timeout integrate, ma fai attenzione perché molti hanno valori predefiniti infiniti o troppo alti. Un valore troppo elevato riduce l'utilità del timeout perché le risorse continuano a essere consumate mentre il client attende che si verifichi il timeout. Un valore troppo basso può generare un aumento del traffico sul back-end e una maggiore latenza perché vengono ritentate troppe richieste. In alcuni casi, questo può portare a interruzioni complete perché tutte le richieste vengono ritentate.

Per ulteriori informazioni su come Amazon utilizza timeout, nuovi tentativi e backoff con jitter, consulta la [Builder's Library: timeout, nuovi tentativi e backoff con jitter](#).

**Rendi i servizi stateless laddove possibile:** i servizi dovrebbero o non richiedere lo stato o 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. In questo modo i server possono essere sostituiti a piacimento senza compromettere la disponibilità. Amazon ElastiCache o Amazon DynamoDB sono ottime destinazioni per lo stato di offload.



*In questa applicazione Web stateless, viene eseguito l'offload dello stato della sessione in Amazon ElastiCache.*

Quando gli utenti o i servizi interagiscono con un'applicazione, spesso eseguono una serie di interazioni che formano una sessione. Una sessione è un dato univoco per gli utenti che persistono tra le richieste mentre utilizzano l'applicazione. Un'applicazione stateless è un'applicazione che non richiede la conoscenza delle interazioni precedenti e non memorizza le informazioni sulla sessione.

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

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

**Implementazione delle leve di emergenza:** si tratta di processi rapidi che possono mitigare l'impatto sulla disponibilità sul carico di lavoro. Possono essere utilizzati in assenza di una causa principale. Una leva di emergenza ideale riduce a zero il carico cognitivo dei resolver fornendo criteri di attivazione e disattivazione completamente deterministici. Le leve di esempio includono il blocco di tutto il traffico del robot o la distribuzione di una risposta statica. Le leve sono spesso manuali, ma possono anche essere automatizzate.

Suggerimenti per l'implementazione e l'utilizzo di leve di emergenza:

- Quando le leve sono attivate, fai di meno, non di più
- Rendi le cose semplici, evita comportamenti bimodali
- Testa periodicamente le leve

Di seguito sono elencati alcuni esempi di operazioni che NON rappresentano leve di emergenza:

- Aggiunta di capacità
- Chiamare i proprietari dei servizi dei client che dipendono dal tuo servizio e chiedere loro di ridurre le chiamate
- Apportare una modifica al codice e rilasciarlo

## Risorse

### Video

- [Nuovi tentativi, backoff e jitter: AWS re:Invent 2019: un'introduzione ad Amazon Builders' Library \(DOP328\)](#)

### Documentazione

- [Ripetizione dei tentativi in caso di errore e backoff esponenziale in AWS](#)
- Amazon API Gateway: [throttling delle richieste API per migliorare il throughput](#)

- Amazon Builders' Library: [timeout, nuovi tentativi e backoff con jitter](#)
- Amazon Builders' Library: [evitare fallback nei sistemi distribuiti](#)
- Amazon Builders' Library: [evitare backlog di coda insormontabili](#)
- Amazon Builders' Library: [sfide e strategie di caching](#)

### Corsi

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

### Collegamenti esterni

- [CircuitBreaker](#) (riepilogo dell'interruttore dal libro "Release It!")

### Libri

- Michael Nygard "[Release It! Design and Deploy Production-Ready Software](#)"

## Gestione delle modifiche

Le modifiche apportate al carico di lavoro o al relativo ambiente devono essere anticipate e sistemate per ottenere un funzionamento affidabile 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 distribuzioni delle caratteristiche e le patch di sicurezza.

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

- Monitoraggio delle risorse
- Progettazione del carico di lavoro per adattarsi ai cambiamenti della domanda
- Implementazione della modifica

## Monitoraggio delle risorse del carico di lavoro

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

Il monitoraggio è essenziale per accertarti di soddisfare i requisiti di disponibilità. Il monitoraggio deve rilevare efficacemente gli errori. La modalità di errore peggiore è l'errore "silenzioso", in cui la funzionalità non è più attiva, ma non c'è modo di rilevarla se non indirettamente. I tuoi clienti

se ne accorgono prima di te. L'avviso in caso di problemi è uno dei principali motivi per cui esegui il monitoraggio. I tuoi avvisi devono essere disaccoppiati dal tuo sistema il più possibile. Se l'interruzione del servizio elimina la possibilità di inviare avvisi, avrai un periodo di interruzione più lungo.

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

Il monitoraggio ad 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. Storage e analisi

**Generazione – monitora tutti i componenti per il carico di lavoro:** monitora i componenti del carico di lavoro con Amazon CloudWatch o strumenti di terze parti. Monitora i servizi AWS con Personal Health Dashboard.

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 di creazione per gli eventi di allarme corrispondenti

Il monitoraggio nel cloud offre nuove opportunità. La maggior parte dei provider di servizi cloud ha sviluppato hook e approfondimenti personalizzabili su più livelli del carico di lavoro.

AWS rende disponibili numerose informazioni di monitoraggio e registrazione, che possono essere utilizzate per definire i processi di modifica della domanda. Di seguito è riportato solo un elenco parziale di servizi e caratteristiche che generano dati di log e parametri.

- Amazon ECS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling e Amazon EMR pubblicano parametri per le medie di CPU, I/O di rete e I/O su disco.
- Amazon CloudWatch Logs può essere abilitato per Amazon Simple Storage Service (Amazon S3), Classic Load Balancer e Application Load Balancer.
- I log di flusso VPC possono essere abilitati per analizzare il traffico di rete da e verso un VPC.

- AWS CloudTrail registra le attività dell'account AWS, incluse le azioni eseguite tramite la Console di gestione AWS, gli SDK AWS e gli strumenti a riga di comando.
- Amazon EventBridge distribuisce un flusso di eventi di sistema in tempo reale che riflette le modifiche nei servizi AWS.
- AWS fornisce strumenti per raccogliere log a livello di sistema operativo e trasmetterli in streaming nei CloudWatch Logs.
- I parametri personalizzati di Amazon CloudWatch possono essere utilizzati per parametri di qualsiasi dimensione.
- Amazon ECS e AWS Lambda eseguono lo streaming dei dati di log in CloudWatch Logs.
- Amazon Machine Learning (Amazon ML), Amazon Rekognition, Amazon Lex e Amazon Polly forniscono parametri per le richieste riuscite e non riuscite.
- AWS IoT fornisce parametri per il numero di esecuzioni di regole, nonché parametri specifici di esito positivo e negativo relativi alle regole.
- Amazon API Gateway fornisce parametri per il numero di richieste, richieste errate e latenza per le tue API.
- Personal Health Dashboard offre una visualizzazione personalizzata delle prestazioni e della disponibilità dei servizi AWS sottostanti alle risorse AWS.

Inoltre, monitora tutti gli endpoint esterni da posizioni remote per assicurarti 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 effettuate dai consumatori dell'applicazione. Mantieni questi elementi di breve durata e assicurati di non sovraccaricare il flusso di lavoro durante il test. Amazon CloudWatch Synthetics consente di [creare canary](#) per monitorare endpoint e API. Puoi anche combinare i nodi client sintetici Canary con la console AWS X-Ray per individuare quali Canary sintetiche stanno riscontrando problemi con errori, guasti o velocità di throttling per l'intervallo di tempo selezionato.

**Aggregazione – definisci e calcola i parametri:** archivia i dati di log e applica filtri, se necessario, per calcolare i parametri, ad esempio i conteggi di un evento di log specifico o la latenza calcolata dai timestamp degli eventi di log.

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

**Elaborazione in tempo reale e allarmi – invia notifiche:** le organizzazioni che devono essere messe al corrente ricevono notifiche quando si verificano eventi significativi.

Gli avvisi possono anche essere inviati ad argomenti Amazon Simple Notification Service (Amazon SNS) e quindi inviati a qualsiasi numero di sottoscrittori. Ad esempio, Amazon SNS può inoltrare avvisi a un alias e-mail in modo che il personale tecnico possa rispondere.

**Elaborazione e allarmi in tempo reale – risposte automatiche:** utilizza l'automazione per intervenire quando viene rilevato un evento, ad esempio per sostituire i componenti guasti.

Gli avvisi possono attivare eventi AWS Auto Scaling, in modo che i cluster reagiscano alle variazioni nella domanda. Gli avvisi possono essere inviati ad Amazon Simple Queue Service (Amazon SQS), che può fungere da punto di integrazione per i sistemi di ticket di terze parti. AWS Lambda può anche sottoscrivere avvisi, fornendo agli utenti un modello serverless asincrono che reagisce alle modifiche in modo dinamico. AWS Config monitora e registra in modo continuo le configurazioni delle risorse AWS e può attivare [AWS Systems Manager Automation](#) per risolvere i problemi.

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

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

Sono disponibili diversi strumenti forniti da partner e terze parti che consentono aggregazione, elaborazione, storage e analisi. Questi strumenti includono New Relic, Splunk, Loggly, Logstash, CloudHealth e Nagios. Tuttavia, la generazione esterna di log di sistema e applicazioni è univoca per ciascun provider di servizi cloud e spesso per ciascun servizio.

Una parte spesso trascurata del processo di monitoraggio è la gestione dei dati. È necessario determinare i requisiti di conservazione per il monitoraggio dei dati, quindi applicare le policy del ciclo di vita di conseguenza. Amazon S3 supporta la gestione del ciclo di vita a livello di bucket S3. Questa gestione del ciclo di vita può essere applicata in modo diverso ai diversi percorsi nel bucket. Verso la fine del ciclo di vita, è possibile trasferire i dati 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.

**Conduci revisioni regolarmente:** esamina frequentemente il modo in cui il monitoraggio del carico di lavoro viene implementato e aggiornalo in base a eventi e modifiche significativi.

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

L'audit del monitoraggio consente di sapere quando un'applicazione sta raggiungendo gli obiettivi di disponibilità. Le analisi della causa principale richiedono la possibilità di scoprire cosa è successo quando si verificano errori. AWS offre servizi che ti consentono di monitorare lo stato dei tuoi servizi durante un incidente:

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

Ad AWS, organizziamo una riunione settimanale per esaminare le prestazioni operative e condividere quanto appreso tra i team. Poiché ci sono così tanti team ad AWS, abbiamo creato [The Wheel](#) (la ruota) per scegliere casualmente un carico di lavoro da esaminare. Stabilire una cadenza regolare per le revisioni delle prestazioni operative e la condivisione delle conoscenze migliora la capacità di ottenere prestazioni più elevate dai team operativi.

**Monitoraggio del tracciamento end-to-end delle richieste attraverso il tuo sistema:** utilizza AWS X-Ray o strumenti di terze parti per consentire agli sviluppatori di analizzare ed eseguire il debug di sistemi distribuiti in modo più semplice per comprendere le prestazioni delle loro applicazioni e dei relativi servizi sottostanti.

## Risorse

### Documentazione

- [Utilizzo dei parametri di Amazon CloudWatch](#)
- [Utilizzo di Canary](#) (Amazon CloudWatch Synthetics)
- [Query di esempio di Amazon CloudWatch Logs Insights](#)
- [AWS Systems Manager Automation](#)
- [Cos'è AWS X-Ray?](#)
- [Debug con Amazon CloudWatch Synthetics e AWS X-Ray](#)
- Amazon Builders' Library: [dotazione dei sistemi distribuiti per la visibilità operativa](#)

## Progetta il carico di lavoro per adattarti alle variazioni della domanda

Un carico di lavoro scalabile fornisce elasticità per aggiungere o rimuovere risorse automaticamente, in modo che corrispondano strettamente alla domanda corrente in qualsiasi momento.

**Utilizzo dell'automazione quando si ottengono o si dimensionano le risorse:** quando si sostituiscono le risorse danneggiate o si dimensiona il carico di lavoro, è possibile automatizzare il processo utilizzando servizi AWS gestiti, quali Amazon S3 e AWS Auto Scaling. Puoi anche utilizzare strumenti di terze parti ed SDK AWS per automatizzare il dimensionamento.

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

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

Durante il dimensionamento di istanze EC2 o container Amazon ECS ospitati su istanze EC2, assicurati di utilizzare più zone di disponibilità (preferibilmente almeno tre) e di aggiungere o rimuovere capacità per mantenere il bilanciamento tra queste zone di disponibilità.

Quando si utilizza AWS Lambda, le risorse vengono ricalibrate automaticamente. Ogni volta che viene ricevuta una notifica di evento per la funzione, AWS Lambda individua rapidamente la capacità libera all'interno del parco istanze di calcolo ed esegue il codice fino alla simultaneità allocata. Devi assicurarti che la simultaneità necessaria sia configurata sulla Lambda specifica e nelle tue quote di servizio.

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

Configura e utilizza Amazon CloudFront o una rete di distribuzione di contenuti affidabile. Una rete per la distribuzione di contenuti (CDN) può fornire tempi di risposta più rapidi agli utenti finali e può dar seguito a richieste di contenuti che potrebbero causare un dimensionamento superfluo dei carichi di lavoro.

**Otteni risorse dopo il rilevamento della compromissione di un carico di lavoro:** dimensiona le risorse in modo reattivo quando necessario se la disponibilità è influenzata, in modo da ripristinare la disponibilità del carico di lavoro.

Devi prima configurare i controlli dello stato e i criteri su questi controlli per indicare quando la disponibilità è influenzata dalla mancanza di risorse. Quindi notificare al personale appropriato di dimensionare 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 istanze EC2 in un gruppo Auto Scaling o modificando il throughput di una tabella DynamoDB tramite la console o l'interfaccia a riga di comando AWS. Tuttavia, l'automazione dovrebbe essere utilizzata ogni volta che è possibile (consulta **Utilizzo dell'automazione durante il dimensionamento di un carico di lavoro**).

**Otteni risorse dopo il rilevamento della necessità di più risorse per un carico di lavoro:** dimensiona le risorse in modo proattivo per soddisfare la domanda ed evitare l'impatto sulla disponibilità.

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

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

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

$$L = \lambda W$$

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

$\lambda$  = velocità media alla quale arrivano le richieste (richieste/sec)

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

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

**Esecuzione di un test del carico di lavoro:** adotta una metodologia 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 individuare il punto di interruzione e testare le prestazioni del carico di lavoro. AWS semplifica la configurazione di

ambienti di test temporanei che modellino il dimensionamento del carico di lavoro di produzione. Nel cloud, puoi creare un ambiente di test su scala di produzione on demand, completare i test e disattivare le risorse. Poiché paghi solo per l'ambiente di test quando è in esecuzione, puoi simulare l'ambiente reale a una frazione del costo dei test in locale.

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

## Risorse

### Documentazione

- AWS Auto Scaling: [come funzionano i piani di dimensionamento](#)
- [Cos'è Amazon EC2 Auto Scaling?](#)
- [Gestione automatica della capacità di throughput con Auto Scaling di DynamoDB](#)
- [Cos'è Amazon CloudFront?](#)
- [Test di carico distribuito su AWS](#): simula migliaia di utenti connessi
- [AWS Marketplace: prodotti che possono essere utilizzati con Auto Scaling](#)
- [Partner APN: partner che possono aiutarti a creare soluzioni di calcolo automatizzate](#)

### Collegamenti esterni

- [Raccontare storie sulla legge di Little](#)

## Implementazione della modifica

Le modifiche controllate sono necessarie per distribuire nuove funzionalità e per garantire che i carichi di lavoro e l'ambiente operativo eseguano software noti con patch corrette. Se queste modifiche non sono controllate, risulta difficile prevedere l'effetto di tali modifiche o risolvere i problemi che si verificano a causa delle stesse.

**Utilizzo dei runbook per attività standard come la distribuzione:** i runbook sono le fasi predefinite per ottenere risultati specifici. Utilizza i runbook per eseguire attività standard, eseguite manualmente o automaticamente. Alcuni esempi includono la distribuzione di un carico di lavoro, l'applicazione di patch o di modifiche DNS.

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

Per le procedure runbook, inizia con un processo manuale valido, implementalo nel codice e attiva l'esecuzione automatizzata, se necessario.

Anche per carichi di lavoro sofisticati e altamente automatizzati, i runbook sono ancora utili per l'[esecuzione di game day](#) 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.

**Integrazione di test funzionali come parte della distribuzione:** i test funzionali vengono eseguiti come parte della distribuzione automatizzata. Se i criteri di esito positivo non vengono soddisfatti, la pipeline viene arrestata o ripristinata.

Questi test vengono eseguiti in un ambiente di pre-produzione, gestito per fasi prima della produzione nella pipeline. Idealmente, questa operazione viene eseguita come parte di una pipeline di distribuzione.

**Integrazione dei test di resilienza come parte della distribuzione:** i test di resilienza (come parte della progettazione del caos) vengono eseguiti come parte della pipeline di distribuzione automatizzata in un ambiente di pre-produzione.

Questi test vengono gestiti per fasi ed eseguiti nella pipeline prima della produzione. Dovrebbero anche essere eseguiti in produzione, ma come parte di [Game Day](#).

**Distribuzione attraverso un'infrastruttura immutabile:** questo è un modello che impone che non vengano eseguiti aggiornamenti, patch di sicurezza o modifiche di configurazione *sul posto* sui sistemi di produzione. Quando è necessaria una modifica, l'architettura viene costruita su una nuova infrastruttura e distribuita in produzione.

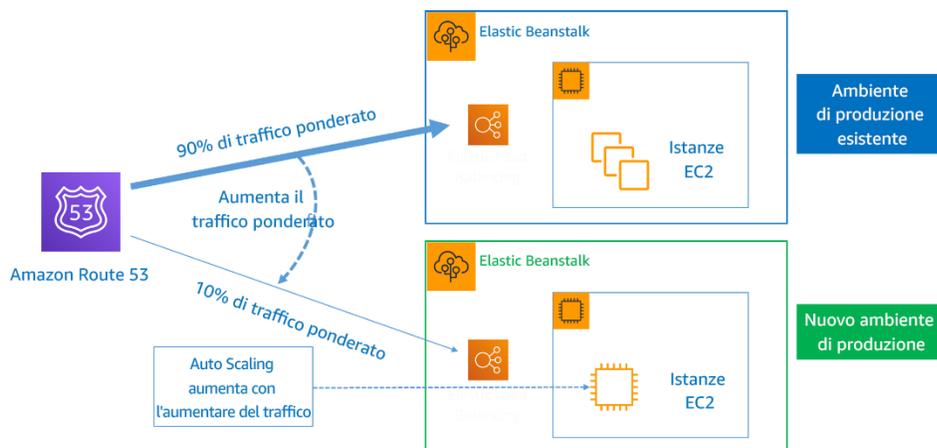
L'implementazione più comune del paradigma dell'infrastruttura immutabile è il *server immutabile*. Ciò significa che se un server necessita di un aggiornamento o di una correzione, vengono distribuiti nuovi server invece di aggiornare quelli già in uso. Pertanto, invece di accedere al server tramite SSH e aggiornare la versione del software, ogni modifica nell'applicazione inizia con un push del software al repository di codice, ad esempio git push. Poiché non sono consentite modifiche nell'infrastruttura immutabile, puoi essere sicuro dello stato del sistema distribuito. Le infrastrutture immutabili sono intrinsecamente più coerenti, affidabili e prevedibili e semplificano molti aspetti dello sviluppo e delle operazioni di software.

Utilizza una distribuzione Canary o blue/green durante la distribuzione di applicazioni in infrastrutture immutabili.

La [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 la distribuzione viene completata correttamente, puoi continuare a distribuire alla velocità

desiderata, monitorando le modifiche alla ricerca di errori, fino a quando la distribuzione non sarà completata. AWS CodeDeploy può essere configurato con una configurazione di distribuzione che abilita una distribuzione Canary.

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



*Distribuzione blue/green con AWS Elastic Beanstalk e Amazon Route 53*

Vantaggi dell'infrastruttura immutabile:

- Riduzione delle deviazioni di configurazione: sostituendo frequentemente i server da una configurazione di base, nota e controllata dalla versione, l'infrastruttura viene reimpostata a uno stato noto, evitando deviazioni di configurazione.
- Distribuzioni semplificate: le distribuzioni sono semplificate perché non devono supportare gli aggiornamenti. Gli aggiornamenti sono solo nuove distribuzioni.
- Distribuzioni atomiche affidabili: le distribuzioni o vengono completate correttamente oppure non vengono apportate modifiche. Offre maggiore fiducia nel processo di distribuzione.
- Distribuzioni più sicure con processi di rollback e ripristino rapidi: le distribuzioni sono più sicure perché la versione funzionante precedente non viene modificata. Puoi eseguire il rollback se vengono rilevati errori.
- Ambienti di test e debug coerenti: poiché tutti i server utilizzano la stessa immagine, non ci sono differenze tra gli ambienti. Una build viene distribuita in più ambienti. Inoltre, evita ambienti incoerenti e semplifica test e debug.

- Maggiore scalabilità: poiché i server utilizzano un'immagine di base, sono coerenti e ripetibili, la scalabilità automatica è banale.
- Toolchain semplificata: la toolchain è semplificata poiché è possibile eliminare gli strumenti di gestione della configurazione che gestiscono gli aggiornamenti del software di produzione. Non vengono installati altri strumenti o agenti sui server. Le modifiche vengono apportate all'immagine di base, testate e implementate.
- Maggiore sicurezza: negando tutte le modifiche ai server, puoi disabilitare SSH sulle istanze e rimuovere le chiavi. Questo riduce il vettore di attacco, migliorando l'assetto di sicurezza dell'organizzazione.

**Distribuzione delle modifiche attraverso l'automazione:** le distribuzioni e l'applicazione di patch sono automatizzate per eliminare un impatto negativo.

Apportare modifiche ai sistemi di produzione è una delle aree di rischio più grandi per molte organizzazioni. Riteniamo che le distribuzioni siano un problema prioritario da risolvere insieme ai problemi aziendali affrontati dal software. Oggi, questo significa l'uso dell'automazione ovunque sia pratico nelle operazioni, tra cui il test e la distribuzione delle modifiche, l'aggiunta o la rimozione di capacità e la migrazione dei dati. AWS CodePipeline consente di gestire le fasi necessarie per rilasciare il carico di lavoro. Questo include uno stato di distribuzione che utilizza AWS CodeDeploy per automatizzare la distribuzione del codice dell'applicazione su istanze Amazon EC2, istanze in locale, funzioni Lambda serverless o servizi Amazon ECS.

#### Suggerimento

Anche se la prassi convenzionale suggerisce di includere le persone per le procedure operative più difficili, ti consigliamo di automatizzare le procedure più complesse proprio per questo motivo.

## Modelli di distribuzione aggiuntivi per ridurre al minimo i rischi:

[I flag delle funzionalità \(noti anche come interruttori funzione\)](#) sono opzioni di configurazione in un'applicazione. È possibile distribuire 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 la distribuzione presenta problemi, è possibile semplicemente disattivare la funzionalità senza eseguire il rollback.

[Distribuzione zonale isolata con errori:](#) una delle regole più importanti che AWS ha stabilito per le proprie distribuzioni è evitare di toccare contemporaneamente più zone di disponibilità all'interno di una regione. Questo è fondamentale per garantire che le zone di disponibilità siano indipendenti ai fini dei calcoli di disponibilità. Si consiglia di utilizzare considerazioni simili nelle distribuzioni.

## 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 delle applicazioni ai suoi contratti di servizio e fornire dati in caso di interruzione o di altre anomalie operative. Una ORR formale viene condotta prima della distribuzione iniziale di produzione. AWS ripeterà periodicamente le ORR (una volta all'anno o prima dei periodi di prestazione critici) per garantire che non ci siano state "deviazioni" dalle previsioni operative. Per ulteriori informazioni sulla prontezza operativa, consulta il [Pilastro dell'eccellenza operativa](#) del [Canone di architettura AWS](#).

### Suggerimento

Effettua una ORR per le applicazioni prima dell'utilizzo iniziale in produzione e successivamente periodicamente.

## Risorse

### Video

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

### Documentazione

- [Che cos'è AWS CodePipeline?](#)
- [Cos'è CodeDeploy?](#)
- [Panoramica di una distribuzione blue/green](#)
- [Distribuzione graduale di applicazioni serverless](#)
- Amazon Builders' Library: [garantire la sicurezza del rollback durante le distribuzioni](#)
- Amazon Builders' Library: [velocizzare la distribuzione continua](#)
- [AWS Marketplace: prodotti per l'automazione delle distribuzioni](#)
- [Partner APN: partner per la creazione di soluzioni di distribuzione automatizzate](#)

### Corsi

- Corso Well-Architected: [livello 300: Testing for Resiliency of EC2 RDS and S3](#)

### Collegamenti esterni

- [ReleaseCanary](#)

## Gestione dei guasti

I guasti sono da mettere in conto e con il tempo tutto finisce per causare guasti: dai router ai dischi rigidi, dai sistemi operativi alle unità di memoria che danneggiano i pacchetti TCP, dagli errori temporanei agli errori permanenti. 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 devono essere risolti ogni giorno in un data center in locale. Nel cloud, tuttavia, devi essere protetto dalla maggior parte di questi tipi di guasti. Ad esempio, i volumi Amazon EBS vengono collocati in una zona di disponibilità specifica in cui vengono automaticamente replicati per proteggerti dai guasti di un singolo componente. Tutti i volumi EBS sono progettati per garantire 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, è necessario adottare misure per implementare la resilienza se è necessario che il tuo carico di lavoro sia affidabile.

Un prerequisito per l'applicazione delle best practice discusse qui è la necessità di accertarsi che le persone che progettano, implementano e gestiscono i tuoi carichi di lavoro siano consapevoli degli obiettivi aziendali e degli obiettivi di affidabilità per raggiungerli. Queste persone devono essere informate e addestrate per questi requisiti di affidabilità.

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

- Esegui il backup dei dati
- Utilizza l'isolamento dei guasti per proteggere il carico di lavoro
- Progetta il tuo carico di lavoro per resistere ai guasti dei componenti
- Testa la resilienza
- Pianificazione in funzione del disaster recovery (DR)

### Esegui il backup dei dati

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

**Identifica ed esegui il backup di tutti i dati di cui è necessario eseguire il backup o riproduci i dati dalle origini:** Amazon S3 può essere utilizzato come destinazione di backup per più origini dati. I servizi AWS come Amazon EBS, Amazon RDS e Amazon DynamoDB hanno funzionalità

integrate per la creazione di backup. In alternativa, è possibile utilizzare software di backup di terze parti. In alternativa, se per soddisfare gli RPO è possibile riprodurre i dati dalle origini, il backup può non essere necessario.

È possibile eseguire il backup dei dati in locale nell'AWS Cloud utilizzando i bucket Amazon S3 e AWS Storage Gateway. I dati di backup possono essere archiviati utilizzando Amazon S3 Glacier o S3 Glacier Deep Archive per uno storage cloud conveniente e senza limiti di tempo.

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.

**Backup sicuro e crittografato:** rileva l'accesso tramite autenticazione e autorizzazione come AWS Identity and Access Management (IAM) e rileva la compromissione dell'integrità dei dati utilizzando la crittografia.

Amazon S3 supporta diversi metodi di crittografia dei dati inattivi. Utilizzando la crittografia lato server, Amazon S3 accetta gli oggetti come dati non crittografati, quindi li crittografa prima di conservarli. Utilizzando la crittografia lato client, l'applicazione del carico di lavoro è responsabile della crittografia dei dati prima che vengano inviati a S3. Entrambi i metodi ti consentono di utilizzare AWS Key Management Service (AWS KMS) per creare e archiviare la chiave di dati oppure di fornire una chiave personalizzata (di cui sei responsabile). Utilizzando AWS KMS, è possibile impostare policy utilizzando AWS IAM su chi può o non può accedere alle chiavi dei dati e ai dati decrittografati.

Per Amazon RDS, se hai scelto di crittografare i database, anche i backup vengono crittografati. I backup di DynamoDB sono sempre crittografati.

**Esegui il backup dei dati automaticamente:** configura i backup in modo che vengano eseguiti automaticamente in base a una pianificazione periodica o mediante modifiche nel set di dati. Le istanze RDS, i volumi EBS, le tabelle DynamoDB e gli oggetti S3 possono essere configurati per il backup automatico. È anche possibile utilizzare soluzioni AWS Marketplace o soluzioni di terze parti.

Amazon Data Lifecycle Manager può essere utilizzato per automatizzare gli snapshot EBS. Amazon RDS e Amazon DynamoDB consentono il backup continuo con ripristino ad un punto temporale specifico. La funzione Versioni multiple di Amazon S3, una volta abilitata, è automatica.

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 in locale utilizzando AWS Storage Gateway.

Oltre alla funzione Versioni multiple, Amazon S3 offre la replica. L'intero bucket S3 può essere replicato automaticamente in un altro bucket in una regione AWS diversa.

**Esegui il ripristino periodico dei dati per verificare l'integrità e i processi dei backup:** convalida che l'implementazione del processo di backup soddisfi l'obiettivo del tempo di ripristino (RTO) e il Recovery Point Objective (RPO) eseguendo un test di ripristino.

Con AWS, è possibile creare un ambiente di test e ripristinare i backup per valutare le funzionalità RTO e RPO ed eseguire test su contenuto e integrità dei dati.

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

## Risorse

### Video

- [AWS re:Invent 2019: approfondimento su AWS Backup, ft. Rackspace \(STG341\)](#)

### Documentazione

- [Cos'è AWS Backup?](#)
- [Amazon S3: protezione dei dati tramite la crittografia](#)
- [Crittografia per i backup in AWS](#)
- [Backup e ripristino on demand per Dynamo DB](#)
- [EFS-to-EFS Backup](#)
- [AWS Marketplace: prodotti che possono essere usati per il backup](#)
- [Partner APN: partner che possono essere d'aiuto con il backup](#)

### I corsi

- Corso Well-Architected, [livello 200: test di backup e ripristino dei dati](#)

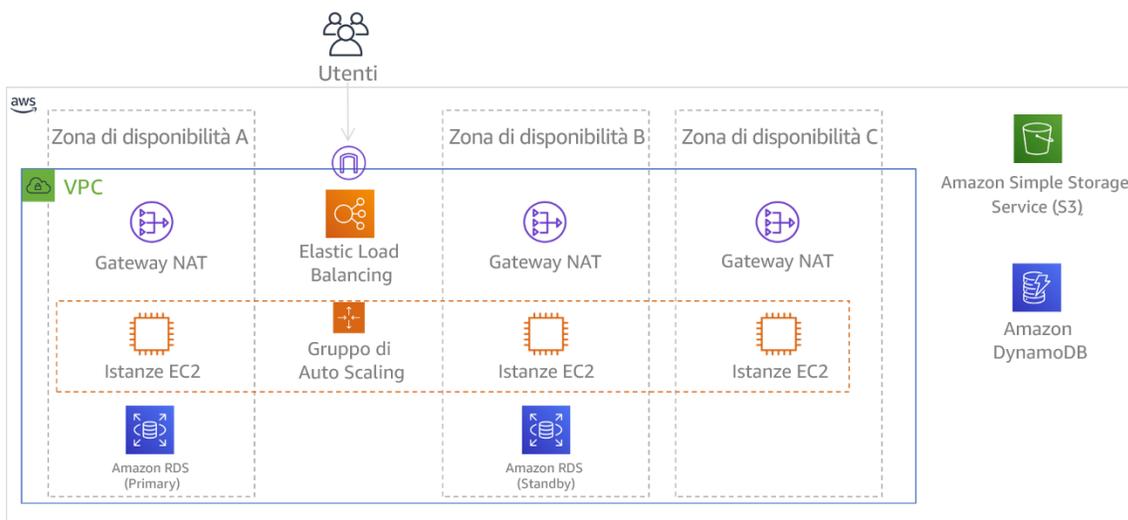
## Utilizzo dell'isolamento dei guasti per proteggere il tuo carico di lavoro

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

**Distribuisci il carico di lavoro in più ubicazioni:** distribuisci i dati e le risorse del carico di lavoro su più zone di disponibilità o, se necessario, su diverse regioni AWS. Questi luoghi possono essere diversi a seconda delle necessità.

Uno dei principi fondamentali per la progettazione di servizi su AWS è l'eliminazione dei punti critici di errore nell'infrastruttura fisica sottostante. Questo ci spinge a creare software e sistemi che utilizzano più zone di disponibilità e sono resistenti ai guasti di una singola zona. Allo stesso modo, i sistemi sono costruiti per resistere ai guasti di un singolo nodo di calcolo, singolo volume di archiviazione o singola istanza di un database. Quando si costruisce un sistema che si basa su componenti ridondanti, è importante garantire che i componenti funzionino in modo indipendente e, nel caso delle regioni AWS, in modo autonomo. I vantaggi ottenuti dai calcoli di disponibilità teorica con componenti ridondanti sono validi solo se questo continua a essere vero.

**Regioni di disponibilità:** le regioni AWS sono composte da almeno due zone di disponibilità progettate per essere indipendenti. Ogni zona di disponibilità è separata da una grande distanza fisica da altre zone per evitare scenari di guasto correlati dovuti a rischi ambientali come incendi, inondazioni e tornado. Ogni zona di disponibilità ha un'infrastruttura fisica indipendente: connessioni dedicate di alimentazione di rete, fonti di alimentazione di backup autonome, servizi meccanici indipendenti e connettività di rete indipendente all'interno e all'esterno della zona di disponibilità. Nonostante siano geograficamente separate, le zone di disponibilità si trovano nella stessa area regionale. Ciò consente la replica sincrona dei dati (ad esempio, tra database) senza impatto eccessivo sulla latenza delle applicazioni. Ciò consente ai clienti di utilizzare le zone di disponibilità in una configurazione attiva/attiva o attiva/standby. Le zone di disponibilità sono indipendenti e pertanto la disponibilità delle applicazioni aumenta quando vengono utilizzate più zone di disponibilità. Alcuni servizi AWS (incluso il piano dati dell'istanza EC2) sono distribuiti come servizi rigorosamente zonal con un comportamento simile alla zona di disponibilità nel suo insieme. Questi servizi vengono utilizzati per gestire in modo indipendente risorse (istanze, database e altre infrastrutture) all'interno della zona di disponibilità specifica. AWS ha da tempo offerto più zone di disponibilità nelle nostre regioni.



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

Mentre i piani di controllo AWS in genere offrono la possibilità di gestire le risorse all'interno dell'intera regione (più zone di disponibilità), alcuni piani di controllo (inclusi Amazon EC2 e Amazon EBS) hanno la capacità di filtrare i risultati in una singola zona di disponibilità. Con questo approccio, la richiesta viene elaborata solo nella zona di disponibilità specificata, riducendo l'esposizione a interruzioni in altre zone di disponibilità. I servizi AWS regionali, invece, utilizzano internamente più zone di disponibilità in una configurazione attiva/attiva per raggiungere gli obiettivi di progettazione della disponibilità che stabiliamo.

### Suggerimento

Quando l'applicazione si basa sulla disponibilità delle API del piano di controllo durante un'interruzione di una **zona di disponibilità**, utilizza i filtri API per richiedere i risultati per una singola zona di disponibilità con ciascuna richiesta API (ad esempio, con DescribeInstances.)

**Le zone locali di AWS** agiscono in modo analogo alle zone di disponibilità all'interno della rispettiva regione AWS in quanto possono essere selezionate come ubicazione di posizionamento per le risorse AWS zonali come sottoreti e istanze EC2. Ciò che li rende speciali è che non si trovano nella regione AWS associata, ma vicino a grandi popolazioni, settori e centri IT in cui al momento non esiste alcuna regione AWS. Tuttavia, mantengono una connessione sicura e a larghezza di banda elevata tra i carichi di lavoro locali nella zona locale e quelli in esecuzione nella regione AWS. È consigliabile utilizzare le zone locali di AWS per distribuire carichi di lavoro più vicini agli utenti per requisiti a bassa latenza.

**Amazon Global Edge Network** è composto da edge location in città di tutto il mondo. Amazon CloudFront utilizza questa rete per distribuire contenuti agli utenti finali a bassa latenza.

AWS Global Accelerator consente di creare endpoint di carico di lavoro in queste edge location per fornire l'onboarding alla rete globale AWS vicina ai tuoi utenti. Amazon API Gateway consente endpoint API ottimizzati per i confini utilizzando una distribuzione CloudFront per facilitare l'accesso client attraverso la edge location più vicina.

**Regioni AWS:** le regioni sono progettate per essere autonome, pertanto, per utilizzare un approccio multi-regione, puoi distribuire copie dedicate dei servizi in ciascuna regione.

### Suggerimento

La maggior parte degli obiettivi di affidabilità per un carico di lavoro può essere raggiunta utilizzando una strategia di AZ multiple all'interno di una singola regione AWS. Solo per i carichi di lavoro che devono essere multi-regione, è bene considerare un'architettura multi-regione.

AWS offre ai clienti la possibilità di gestire servizi in più regioni. Ad esempio, Amazon Aurora Global Database, le tabelle globali Amazon DynamoDB, la replica tra regioni per Amazon S3, le repliche di lettura tra regioni con Amazon RDS e la possibilità di copiare vari snapshot e Amazon Machine Image (AMI) in altre regioni. Tuttavia, lo facciamo in modi che tutelano l'autonomia della regione. Esistono pochissime eccezioni a questo approccio, inclusi i nostri servizi edge globali (come Amazon CloudFront e Amazon Route 53), insieme al piano di controllo per il servizio AWS Identity and Access Management (IAM). La stragrande maggioranza dei servizi opera interamente all'interno di una singola regione.

**Data center in locale:** per carichi di lavoro eseguiti in un data center in locale, progetta un'esperienza ibrida quando possibile. AWS Direct Connect fornisce una connessione di rete dedicata dalla tua sede ad AWS che consente l'esecuzione in entrambi.

Un'altra opzione consiste nell'eseguire l'infrastruttura e i servizi AWS in locale utilizzando AWS Outposts. AWS Outposts è un servizio completamente gestito che estende l'infrastruttura AWS, i servizi, le API e gli strumenti AWS al tuo data center. La stessa infrastruttura hardware utilizzata nell'AWS Cloud viene installata nel data center. Gli Outposts vengono quindi collegati alla regione AWS più vicina. Puoi quindi utilizzare Outposts per supportare i carichi di lavoro che hanno requisiti di bassa latenza o di elaborazione dei dati locali.

**Automatizza il ripristino dei componenti vincolati a una singola posizione:** se i componenti del carico di lavoro possono essere eseguiti solo in una singola zona di disponibilità o in un data center in locale, è necessario implementare la capacità di eseguire una ricostruzione completa del carico di lavoro entro gli obiettivi di ripristino definiti.

Se la best practice per distribuire il carico di lavoro in più posizioni non è possibile a causa di vincoli tecnologici, 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.

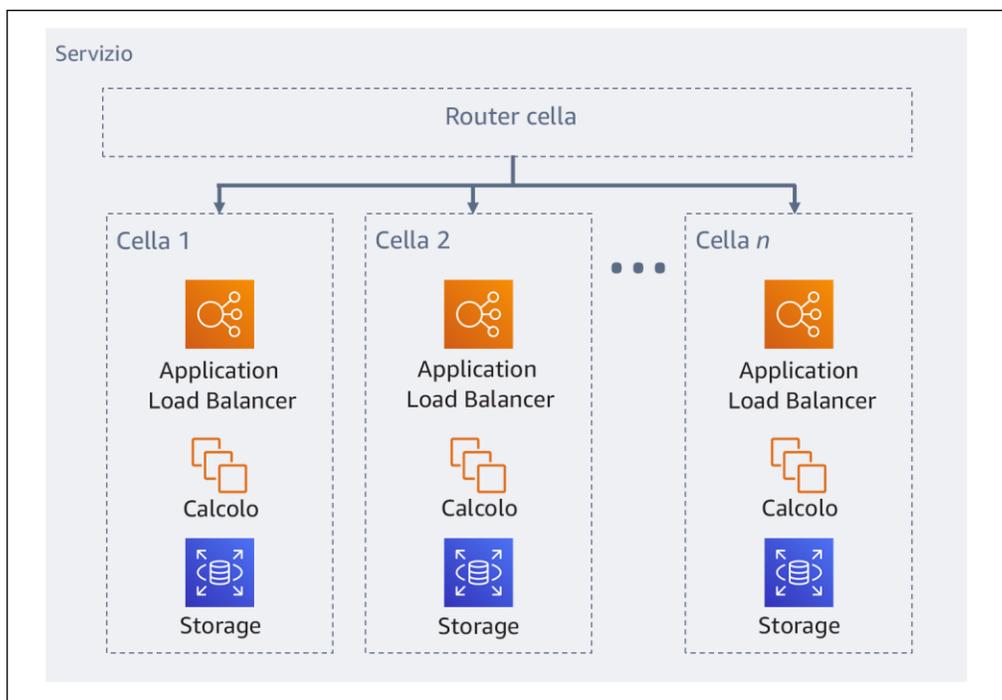
Ad esempio, Amazon EMR lancia tutti i nodi per un determinato cluster nella stessa zona di disponibilità perché l'esecuzione di un cluster nella stessa zona migliora le prestazioni dei flussi

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 ridistribuire 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 master](#). Utilizzando [EMR File System \(EMRFS\)](#), 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 per Amazon Redshift, per impostazione predefinita effettua il provisioning del cluster in una zona di disponibilità selezionata casualmente all'interno della regione AWS selezionata. Tutti i nodi del cluster vengono assegnati nella stessa zona.

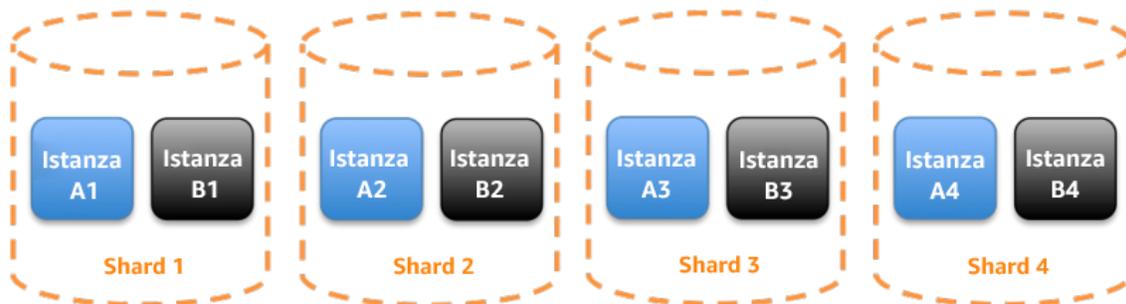
**Utilizzare architetture paratie:** come le paratie su una nave, questo modello garantisce che un guasto sia contenuto a un piccolo sottoinsieme di richieste/utenti, in modo che il numero di richieste danneggiate sia limitato e la maggior parte possa continuare senza errori. Le paratie per i dati sono in genere chiamate *partizioni o shard*, mentre le paratie per i servizi sono note come *celle*.

In un' *architettura basata su celle*, ogni cella è un'istanza completa e indipendente del servizio e ha una dimensione massima fissa. Con l'aumentare del carico, i carichi di lavoro aumentano aggiungendo più celle. Una chiave di partizione viene utilizzata sul traffico in entrata per determinare quale cella elaborerà la richiesta. Qualsiasi guasto è contenuto nella singola cella in cui si verifica, in modo che il numero di richieste danneggiate sia limitato man mano che le altre celle continuano senza errori. È importante identificare la chiave di partizione corretta per ridurre al minimo le interazioni tra celle ed evitare la necessità di coinvolgere servizi di mappatura complessi in ogni richiesta. I servizi che richiedono una mappatura complessa finiscono semplicemente per spostare il problema ai servizi di mappatura, mentre i servizi che richiedono interazioni tra celle riducono l'indipendenza delle celle (e quindi i presunti miglioramenti della disponibilità che ne deriverebbero).



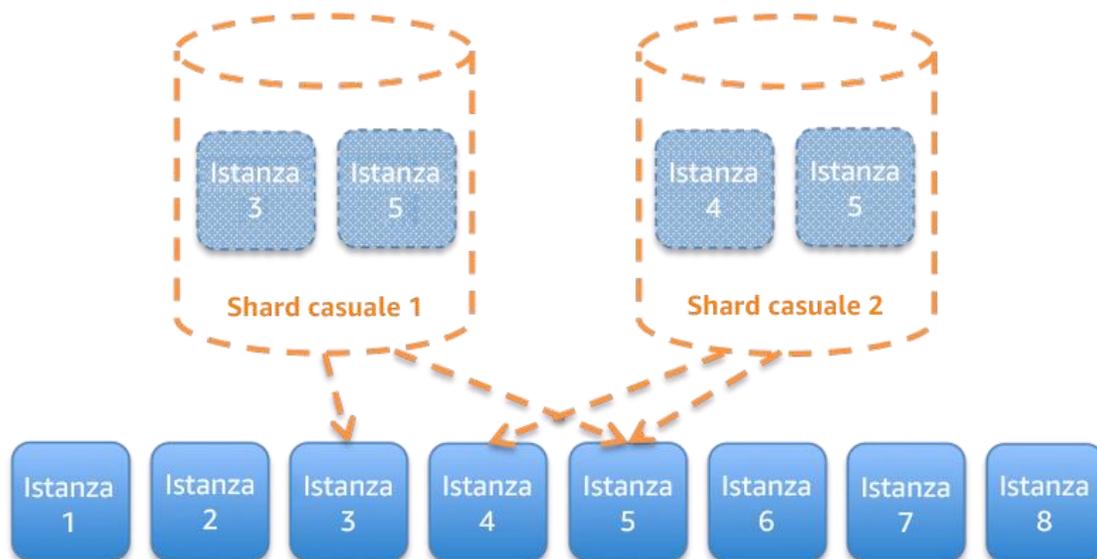
Architettura basata su celle

Nel suo post del blog AWS, Colm MacCarthaigh spiega in che modo Amazon Route 53 utilizza il concetto di [sharding casuale](#) per isolare le richieste dei clienti negli shard. Uno shard in questo caso è costituito da due o più celle. In base alla chiave di partizione, il traffico da un cliente (o risorse o qualsiasi altra cosa desideri isolare) viene instradato allo shard assegnato. Nel caso di otto celle con due celle per shard e clienti divisi tra i quattro shard, il 25% dei clienti riscontrebbe un impatto in caso di problema.



*Servizio diviso in quattro shard tradizionali di due celle ciascuno*

Con lo sharding casuale, puoi creare shard virtuali di due celle ciascuno e assegnare i clienti a uno di questi shard virtuali. Quando si verifica un problema, puoi comunque perdere un quarto dell'intero servizio, ma il modo in cui vengono assegnati i clienti o le risorse significa che l'ambito dell'impatto con lo sharding casuale è notevolmente inferiore al 25%. Con otto celle, ci sono 28 combinazioni univoche di due celle, il che significa che ci sono 28 possibili shard casuali (shard virtuali). Se disponi di centinaia o migliaia di clienti e assigni ogni cliente a uno shard casuale, l'impatto causato da un problema è di solo 1/28. Questo è sette volte superiore rispetto allo sharding normale.



*Servizio diviso in 28 shard casuali (shard virtuali) di due celle ciascuno (vengono mostrati solo due shard casuali su 28 possibili)*

Uno shard può essere utilizzato per server, code o altre risorse oltre alle celle.

## Risorse

### Video

- [AWS re:Invent 2018: Modelli architetturali per applicazioni attive-attive su più regioni \(ARC209-R2\)](#)
- [Sharding casuale: AWS re:Invent 2019: Introduzione alla Amazon Builders' Library \(DOP328\)](#)
- [AWS re:Invent 2018: come AWS riduce al minimo il raggio di esplosione degli guasti \(ARC338\)](#)
- [AWS re:Invent 2019: innovazione e funzionamento dell'infrastruttura di rete globale di AWS \(NET339\)](#)

### Documentazione

- [Cos'è AWS Outposts?](#)
- [Global Tables: replica multi-regione con DynamoDB](#)
- [Domande frequenti sulle zone locali di AWS](#)
- [Infrastruttura globale AWS](#)
- [Regioni, zone di disponibilità e zone locali](#)
- The Amazon Builders' Library: [isolamento del carico di lavoro tramite sharding-virtuale](#)

## Progettare il carico di lavoro per resistere ai guasti 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.

**Monitorare tutti i componenti del carico di lavoro per rilevare i guasti:** monitora continuamente lo stato del carico di lavoro, in modo che tu e i tuoi sistemi automatizzati siano consapevoli del deterioramento o del guasto completo non appena questi si verificano. Monitora gli indicatori chiave di prestazioni (KPI, key performance indicators) in base al valore aziendale.

Tutti i meccanismi di ripristino e correzione devono iniziare con la capacità 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, perciò questa deve essere una misura fondamentale della strategia di rilevamento e correzione.

**Failover verso risorse integre in percorsi non danneggiati:** assicurati che se si verifica un guasto in una posizione, i dati e le risorse provenienti da posizioni integre possano continuare a servire le richieste. Ciò è più semplice per i carichi di lavoro multi-zona, poiché i servizi AWS,

come Elastic Load Balancing e AWS Auto Scaling, aiutano a distribuire il carico tra le zone di disponibilità. Per i carichi di lavoro multi-regione, questa operazione è più complicata. Ad esempio, le repliche di lettura tra regioni consentono di distribuire i dati in più regioni AWS, ma è comunque necessario promuovere la replica di lettura per dominare e indirizzare il traffico verso di essa in caso di guasto di una posizione primaria. Amazon Route 53 e AWS Global Accelerator possono anche aiutare a instradare il traffico tra regioni AWS.

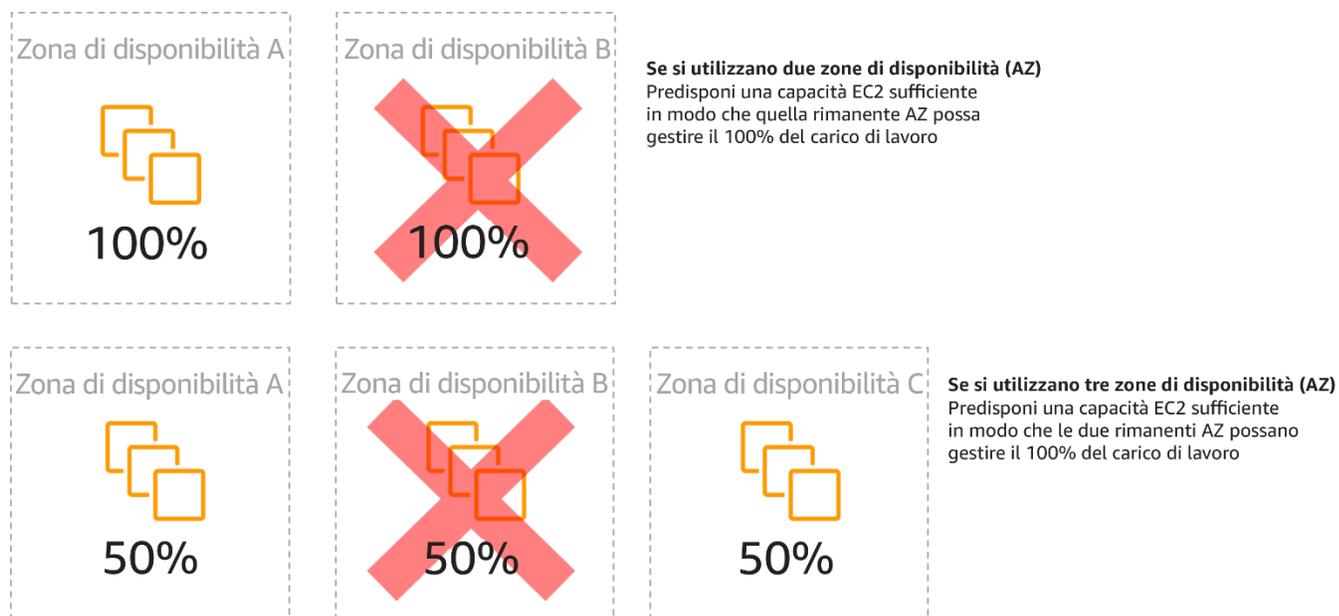
Se il carico di lavoro utilizza servizi AWS, ad esempio Amazon S3 o Amazon DynamoDB, questi vengono automaticamente distribuiti in più zone di disponibilità. In caso di guasto, il piano di controllo AWS instrada automaticamente il traffico verso le posizioni integre per te. Per Amazon RDS, è necessario scegliere AZ multiple come opzione di configurazione, quindi in caso di guasto, AWS indirizza automaticamente il traffico verso l'istanza integra. Per le istanze Amazon EC2 o le attività Amazon ECS, sei tu a scegliere le zone di disponibilità in cui distribuire. Elastic Load Balancing fornisce quindi la soluzione per rilevare le istanze in zone non integre e instradare il traffico verso quelle integre. Elastic Load Balancing può anche instradare il traffico verso componenti nel tuo data center in locale.

Per gli approcci multi-regione (che potrebbero includere anche data center in locale), Amazon Route 53 offre un modo per definire domini Internet e assegnare policy di instradamento che possono includere controlli dello stato per garantire che il traffico venga instradato verso regioni integre. In alternativa, AWS Global Accelerator fornisce indirizzi IP statici che fungono da punto di ingresso fisso alla tua applicazione, quindi instrada verso endpoint nelle regioni AWS di tua scelta, utilizzando la rete globale AWS anziché Internet per migliorare le prestazioni e l'affidabilità.

AWS si avvicina alla progettazione dei servizi pensando al ripristino degli errori. Progettiamo servizi per ridurre al minimo i tempi di ripristino dei 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. Questi servizi e risorse includono Amazon Aurora, Amazon Relational Database Service (Amazon RDS), istanze DB ad AZ multiple, Amazon S3, Amazon DynamoDB, Amazon Simple Queue Service (Amazon SQS) e Amazon Elastic File System (Amazon EFS). Questi sono costruiti per utilizzare l'isolamento basato su celle e utilizzare l'indipendenza delle 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.

**Utilizza la stabilità statica per prevenire un comportamento bimodale:** si ha un comportamento bimodale 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 ha esito negativo. Devi invece creare sistemi che siano staticamente stabili e operino in una sola modalità. In questo caso, effettua il provisioning di istanze sufficienti in ciascuna zona per gestire il carico di lavoro se una zona è stata rimossa, quindi utilizza Elastic Load Balancing o i controlli dello stato di Amazon Route 53 per spostare il carico dalle istanze danneggiate.

La stabilità statica per la distribuzione di calcolo (ad esempio istanze EC2 o container) determinerà la massima affidabilità. Questa operazione deve essere valutata in base ai problemi relativi ai costi. Eseguire il provisioning di minore capacità di elaborazione e affidarsi all'avvio di nuove istanze in caso di guasto è meno costoso. Tuttavia, per i guasti su larga scala (ad esempio un errore nella zona di disponibilità), questo approccio è meno efficace perché si basa sulla reazione ai guasti nel momento in cui si verificano, piuttosto che prepararsi a tali problemi prima che accadano. La soluzione deve valutare l'affidabilità rispetto alle esigenze di costo per il carico di lavoro. Utilizzando più zone di disponibilità, la quantità di elaborazione aggiuntiva necessaria per la stabilità statica diminuisce.



Dopo il trasferimento del traffico, utilizza AWS Auto Scaling per sostituire in modo asincrono le istanze dalla zona interessata dal guasto e avviarle nelle zone integre.

Un altro esempio di comportamento bimodale potrebbe essere un timeout di rete che potrebbe causare un tentativo di aggiornamento dello stato di configurazione dell'intero sistema. Ciò aggiungerebbe un carico imprevisto a un altro componente, che potrebbe quindi causare un errore, innescando altre conseguenze impreviste. Questo loop di feedback negativo influisce sulla disponibilità del tuo carico di lavoro. Al contrario, è necessario creare sistemi che siano staticamente stabili e funzionino in una sola modalità. Un progetto staticamente stabile sarebbe quello di eseguire un [lavoro costante](#) e aggiornare sempre, con cadenze fisse, lo stato di configurazione. Quando una chiamata non riesce, il carico di lavoro utilizza il valore precedentemente memorizzato nella cache e attiva 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 soddisfi le esigenze del client, ma non dovrebbe essere consentita perché modifica in modo significativo le richieste sul carico di lavoro e potrebbe causare guasti.

**Automatizza il ripristino su tutti i livelli:** al rilevamento di un guasto, utilizza funzionalità automatizzate per eseguire azioni da correggere.

*La capacità di riavvio* è uno strumento importante per risolvere i guasti. Come illustrato in precedenza per i sistemi distribuiti, una best practice consiste nel rendere i servizi stateless laddove possibile. In questo modo si evita la perdita di dati o la disponibilità al riavvio. Nel cloud, puoi (e generalmente dovresti) sostituire l'intera risorsa (ad esempio, l'istanza EC2 o la funzione Lambda) 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. Anziché creare nuovi meccanismi per intrappolare, identificare e correggere ciascuno dei diversi tipi di guasto, mappa diverse categorie di guasto alla stessa strategia di ripristino. Un'istanza può restituire un guasto causato da un guasto hardware, da un bug del sistema operativo, da una memory leak o da altre cause. Anziché creare una correzione personalizzata per ogni situazione, considera una di esse come un guasto dell'istanza. Termina l'istanza e consenti ad AWS Auto Scaling di sostituirla. In un secondo momento, esegui l'analisi sulla risorsa guasta fuori banda.

Un altro esempio è la possibilità di riavviare una richiesta 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 back-off e un jitter esponenziali.

*La capacità di riavvio* è un meccanismo di ripristino presente nelle architetture di cluster ROC (Recovery Oriented Computing) e ad alta disponibilità.

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 attivare AWS Lambda (o altri target) per eseguire una logica di remediation personalizzata sul 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. Se utilizzi AWS OpsWorks, puoi configurare il ripristino automatico delle istanze EC2 a livello del layer.

Per le sostituzioni su larga scala (ad esempio la perdita di un'intera zona di disponibilità), [la stabilità statica](#) è preferibile per l'alta disponibilità anziché cercare di ottenere più nuove risorse contemporaneamente.

**Invia notifiche quando gli eventi influiscono sulla disponibilità:** le notifiche vengono inviate al rilevamento di eventi significativi, anche se il problema causato dall'evento è stato risolto automaticamente.

Il ripristino automatizzato consente l'affidabilità del carico di lavoro. Tuttavia, può anche oscurare i 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. Gli allarmi di Amazon CloudWatch possono essere attivati in base ai guasti che si verificano. Possono anche attivarsi in base alle operazioni di ripristino automatizzato eseguite. 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.

## Risorse

### Video

- [Stabilità statica in AWS: AWS re:Invent 2019: Introduzione ad Amazon Builders' Library \(DOP328\)](#)

### Documentazione

- AWS OpsWorks: [utilizzare il ripristino automatico per sostituire le istanze con errori](#)
- [Cos'è Amazon EventBridge?](#)
- [Amazon Route 53: scelta di una policy di instradamento](#)
- [Cos'è AWS Global Accelerator?](#)
- Amazon Builders' Library: [stabilità statica usando le zone di disponibilità](#)
- Amazon Builders' Library: [implementazione dei controlli dello stato](#)
- [AWS Marketplace: prodotti utilizzabili per tolleranza ai guasti](#)
- [Partner APN: partner che possono essere d'aiuto con l'automazione della tolleranza ai guasti](#)

### Corsi

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

### Collegamenti esterni

- [Il progetto di informatica orientata al ripristino \(ROC, Recovery-Oriented Computing\) Berkeley/Stanford](#)

## Affidabilità dei test

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, perché bug o colli di bottiglia delle prestazioni possono influire sull'affidabilità del tuo carico di lavoro. Testa la resilienza del tuo carico di lavoro per trovare bug latenti che emergono solo in produzione. Esegui questi test regolarmente.

**Usa i playbook per analizzare gli errori:** abilita risposte coerenti e tempestive a scenari di guasto che non sono ben compresi, documentando il processo di analisi nei playbook. I playbook sono le fasi predefinite eseguite per identificare i fattori che contribuiscono a uno scenario di guasto. I risultati provenienti da qualsiasi fase del processo vengono utilizzati per stabilire i passaggi da intraprendere successivamente fino all'identificazione o alla risoluzione del problema.

Il playbook è una pianificazione proattiva che è necessario eseguire, in modo da poter intraprendere azioni reattive in modo efficace. Quando durante la produzione si verificano scenari di guasto non coperti dal playbook, risolvi innanzitutto il problema (spegni l'incendio). Quindi torna indietro e osserva le fasi intraprese per risolvere il problema e utilizzale per aggiungere una nuova voce al playbook.

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

**Esegui analisi post-imprevisto:** esamina gli eventi che hanno avuto un impatto sui clienti e identifica i fattori che vi hanno contribuito e gli elementi delle azioni preventive. Utilizza queste informazioni per sviluppare modi per limitare o prevenire il ripetersi degli imprevisti. Sviluppa procedure per attivare risposte rapide ed efficaci. Comunica i fattori che hanno contribuito al presentarsi dell'imprevisto e le azioni correttive secondo necessità, specificamente mirate per il pubblico di destinazione.

Valuta perché i test esistenti non hanno individuato il problema. Aggiungi i test per questo caso se i test non esistono già.

**Requisiti funzionali dei test:** includono test delle unità e test di integrazione che convalidano la funzionalità richiesta.

Puoi ottenere i migliori risultati quando questi test vengono eseguiti automaticamente come parte delle operazioni di sviluppo e distribuzione. Ad esempio, utilizzando AWS CodePipeline, gli sviluppatori affidano le modifiche a un repository di origine in cui CodePipeline rileva automaticamente le modifiche. Queste modifiche vengono create e vengono eseguiti test. Una volta completati i test, il codice creato viene distribuito ai server temporaneo per il test. Dal server temporaneo, CodePipeline esegue più test, 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.

Inoltre, l'esperienza dimostra che 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. Amazon CloudWatch Synthetics consente di [creare canary](#) per monitorare endpoint e API.

**Requisiti di dimensionamento e prestazioni dei test:** includono il test del carico per verificare che il carico di lavoro soddisfi i requisiti di dimensionamento e prestazioni.

Nel cloud, puoi creare un ambiente di test su scala di produzione on demand per il tuo carico di lavoro. Se esegui questi test su un'infrastruttura ridotta, devi dimensionare i risultati osservati in base a ciò che pensi accadrà in produzione. I test di carico e prestazioni possono essere eseguiti anche in produzione se si fa attenzione a non influire sugli utenti effettivi e si contrassegna con tag i dati di test in modo da non utilizzare dati utente reali e non danneggiare le statistiche di utilizzo o i report di produzione.

Con i test, assicurati che le risorse di base, le impostazioni di dimensionamento, le quote di servizio e la progettazione di resilienza funzionino come previsto sotto carico.

**Testa la resilienza utilizzando l'ingegneria del caos:** esegui test che inseriscono regolarmente guasti negli ambienti di pre-produzione e produzione. Ipotizza il modo in cui il carico di lavoro reagirà al guasto, quindi confronta la tua ipotesi con i risultati del test ed esegui l'iterazione se non corrispondono. Assicurati che il test di produzione non influisca sugli utenti.

Nel cloud, puoi testare il modo in cui il carico di lavoro incorre nell'errore e convalidare le procedure di ripristino. È possibile utilizzare l'automazione per simulare diversi errori o per ricreare scenari che in precedenza hanno condotto a errori. Questo espone percorsi di guasto che è possibile testare e correggere prima che si verifichi uno scenario di guasto reale, riducendo così il rischio.

L'ingegneria del caos è la disciplina che sperimenta un sistema per creare fiducia nella capacità del sistema di affrontare condizioni turbolenti nella produzione. - [Principi di ingegneria del caos](#)

Negli ambienti di pre-produzione e test, l'ingegneria del caos deve essere eseguita regolarmente e far parte del ciclo CI/CD. In produzione, i team devono fare attenzione a non interrompere la disponibilità e utilizzare *giornate di gioco* (game days) per controllare i rischi dell'ingegneria del caos nella produzione.

Le attività di test devono essere commisurate ai tuoi obiettivi di disponibilità. Eseguire test per accertarti di poter soddisfare i tuoi obiettivi di disponibilità è l'unico modo per essere certo di poter conseguire tali obiettivi.

Testa i guasti dei componenti ai quali il carico di lavoro è stato progettato per essere resiliente. Queste includono la perdita di istanze EC2, il guasto dell'istanza database principale di Amazon RDS e le interruzioni della zona di disponibilità.

Verifica l'indisponibilità delle dipendenze esterne. La resilienza della tua applicazione ai guasti temporanei delle dipendenze dovrebbe essere testata per tempi che variano da meno di un secondo a diverse ore.

Altre modalità di degrado possono causare funzionalità ridotte e risposte lente, spesso con conseguente diminuzione delle prestazioni dei servizi. Le fonti comuni di questo degrado sono una maggiore latenza nei servizi critici e una comunicazione di rete inaffidabile (pacchetti persi). Potresti voler usare la possibilità di inserire guasti nel tuo sistema, inclusi gli effetti di rete, come latenza e messaggi persi ed errori DNS, come l'impossibilità di risolvere un nome o di non essere in grado di stabilire connessioni a servizi dipendenti.

Sono disponibili diverse opzioni di terze parti per l'introduzione di guasti. Queste includono opzioni open source quali [Netflix Simian Military](#), [The Chaos Toolkit](#) e [Shopify Toxiproxy](#), nonché opzioni commerciali come [Gremlin](#). Consigliamo l'utilizzo di script autogestiti per le indagini iniziali su come implementare l'ingegneria del caos. In questo modo i team di ingegneri possono sentirsi a proprio agio con il modo in cui il caos viene introdotto nei loro carichi di lavoro. Per esempi di questi, consulta [Testing for Resiliency of EC2 RDS and S3](#) (Test per la resilienza di EC2 RDS e S3) utilizzando più linguaggi, ad esempio Bash, Python, Java e PowerShell. È inoltre necessario implementare l'[inserimento di caos in Amazon EC2 utilizzando AWS Systems Manager](#), che consente di simulare i cali di prestazioni e le condizioni di CPU elevate utilizzando i documenti di AWS Systems Manager.

**Conduci giornate di gioco con regolarità:** utilizza le giornate di gioco per esercitare regolarmente le procedure di guasto il più vicino possibile alla produzione (anche negli ambienti di produzione) con le persone coinvolte in scenari di errore reali. Le giornate di gioco applicano misure per garantire che i test di produzione non influiscano sugli utenti.

Esegui test sulle prestazioni della tua architettura e dei tuoi processi pianificando regolarmente *giornate di gioco* per simulare eventi della produzione. Questo ti aiuta a capire dove puoi apportare dei miglioramenti e a sviluppare un'esperienza organizzativa nella gestione degli eventi.

Quando la progettazione per la resilienza è in loco ed è stata testata in ambienti non di produzione, una giornata di gioco è il modo per garantire che tutto funzioni come pianificato in produzione. Una giornata di gioco, soprattutto la prima, è un'attività di duro lavoro per tutti, in cui tutti gli ingegneri e i team operativi vengono informati in merito a quando accadrà e cosa accadrà. I playbook sono in loco. Il guasto viene quindi inserito nei sistemi di produzione nel modo stabilito e l'impatto viene valutato. Se tutti i sistemi funzionano come progettato, il rilevamento e la correzione automatica avvengono con un impatto minimo o nullo. Tuttavia, se si osserva un impatto negativo, viene eseguito il rollback del test e i problemi relativi al carico di lavoro vengono risolti, se necessario manualmente (utilizzando il playbook). Poiché le giornate di gioco hanno luogo in produzione, è necessario prendere tutte le precauzioni per garantire che non vi sia alcun impatto sulla disponibilità per i clienti.

## Risorse

### Video

- [AWS re:Invent 2019: migliorare la resilienza con l'ingegneria del caos \(DOP309-R1\)](#)

### Documentazione

- [Distribuzione continua e integrazione continua](#)
- [Utilizzo di Canary](#) (Amazon CloudWatch Synthetics)
- [Utilizza CodePipeline con AWS CodeBuild per testare codice ed eseguire build](#)
- [Automatizza i playbook operativi con AWS Systems Manager](#)
- [AWS Marketplace: prodotti utilizzabili per integrazione continua](#)
- [Partner APN: partner che possono essere d'aiuto nell'implementazione di una pipeline di integrazione continua](#)

### Corsi

- Corso Well-Architected: [livello 300: Testing for Resiliency of EC2 RDS and S3](#)

### Collegamenti esterni

- [Principi dell'ingegneria del caos](#)
- [Resilience Engineering: imparare a trarre vantaggio dai guasti](#)
- [Apache JMeter](#)

### Libri

- Casey Rosenthal, Lorin Hochstein, Aaron Blohowiak, Nora Jones, Ali Basiri. ["Chaos Engineering"](#) (Ingegneria del caos) (August 2017)

## Piano per il disaster recovery (DR)

Avere backup e componenti del carico di lavoro ridondanti in loco è l'inizio della strategia di disaster recovery. RTO e RPO sono i tuoi obiettivi per il ripristino della disponibilità. 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.

**Definisci gli obiettivi di ripristino per i tempi di inattività e la perdita di dati:** il carico di lavoro ha un obiettivo di tempo di ripristino (RTO) e un Recovery Point Objective (RPO).

L'obiettivo del tempo di ripristino (RTO) è il periodo di tempo complessivo in cui i componenti del carico di lavoro possono trovarsi nella fase di ripristino e quindi non disponibili, prima di influire negativamente sulla missione o sui processi aziendali dell'organizzazione.

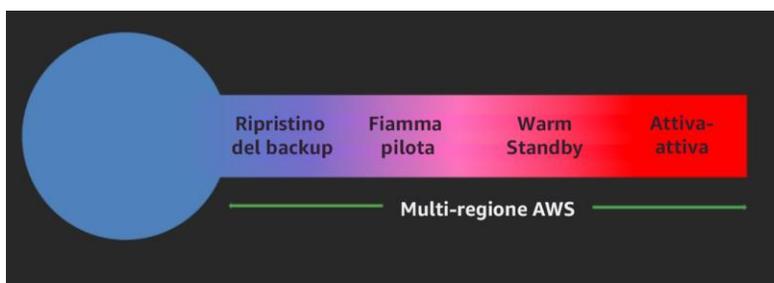
Recovery Point Objective (RPO) è il periodo di tempo complessivo di indisponibilità dei dati di un carico di lavoro, prima di influire negativamente sulla missione o sui processi aziendali dell'organizzazione.

Necessariamente, l'RPO deve essere inferiore all'RTO, poiché la disponibilità non può essere ripristinata con dati critici.

**Utilizzare strategie di ripristino definite per raggiungere gli obiettivi di ripristino:** è stata definita una strategia di disaster recovery (DR) per soddisfare gli obiettivi del carico di lavoro.

A meno che non sia necessaria una strategia multi-regione, ti consigliamo di raggiungere gli obiettivi di ripristino in AWS utilizzando più zone di disponibilità all'interno di una regione AWS.

Se necessario, quando si progetta una strategia multi-regione per il carico di lavoro, è necessario scegliere una delle seguenti strategie. Sono elencati in ordine crescente di complessità e in ordine decrescente di RTO e RPO. La *regione DR* si riferisce a una regione AWS diversa da quella utilizzata per il carico di lavoro (o a qualsiasi regione AWS se il carico di lavoro è locale).



- **Backup e ripristino** (RPO in ore, RTO in 24 ore o meno): esegui il backup dei dati e delle applicazioni nella regione DR. Ripristina questi dati quando necessario per il ripristino in caso di emergenza.
- **Fiamma pilota** (RPO in minuti, RTO in ore): mantieni una versione minima di un ambiente che esegue sempre gli elementi principali più critici del sistema nella regione DR. Quando si rende necessario il ripristino, è possibile effettuare rapidamente il provisioning di un ambiente di produzione completo partendo da questi elementi di base più critici.
- **Warm standby** (RPO in secondi, RTO in minuti): mantieni una versione ridotta di un ambiente completamente funzionante sempre in esecuzione nella regione DR. I sistemi business critical sono completamente duplicati e sono sempre accesi, ma con un parco istanze ridimensionato. Quando viene il momento del ripristino, il sistema viene dimensionato rapidamente per gestire il carico di produzione.

- **Active-active multi-regione** (RPO è none o possibilmente secondi, RTO in secondi): il carico di lavoro viene distribuito e serve attivamente il traffico da più regioni AWS. Questa strategia richiede la sincronizzazione di utenti e dati tra le regioni in uso. Quando si rende necessario il ripristino, utilizza servizi come Amazon Route 53 o AWS Global Accelerator per instradare il traffico degli utenti verso il punto in cui il carico di lavoro è integro.

### Suggerimento

La differenza tra fiamma pilota e warm standby può talvolta essere difficile da comprendere. Entrambe le opzioni includono un ambiente in esecuzione nella regione DR. Tra queste, se la strategia di disaster recovery prevede la distribuzione di un'infrastruttura aggiuntiva, utilizza fiamma pilota. Se implica solo il dimensionamento e il ridimensionamento dell'infrastruttura esistente, utilizza warm standby. Scegli tra queste opzioni in base alle tue esigenze di RTO e RPO.

**Testare l'implementazione di disaster recovery per convalidare l'implementazione:** testare regolarmente il failover su DR per garantire che RTO e RPO siano raggiunti.

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 funziona. Nell'esempio appena discusso, è necessario eseguire il failover regolarmente in standby, indipendentemente dalle necessità.

**Gestisci la deviazione di configurazione nel sito o nella regione del DR:** assicurati che l'infrastruttura, i dati e la configurazione siano in base alle esigenze nel sito o nella regione del DR. Ad esempio, controlla che le AMI e le quote di servizio siano aggiornate.

AWS Config monitora e registra in modo continuo le configurazioni delle tue risorse AWS. È in grado di rilevare le deviazioni e attivare [AWS Systems Manager Automation](#) per risolverle e attivare allarmi. AWS CloudFormation è inoltre in grado di rilevare le deviazioni negli stack distribuiti.

**Automatizza il ripristino:** utilizza AWS o strumenti di terze parti per automatizzare il ripristino del sistema e instradare il traffico verso il sito o la regione DR.

In base ai controlli dello stato configurati, i servizi AWS, come Elastic Load Balancing e AWS Auto Scaling, possono distribuire il carico a zone di disponibilità integre, mentre servizi come Amazon Route 53 e AWS Global Accelerator, possono instradare il carico verso regioni AWS integre.

Per carichi di lavoro su data center fisici o virtuali esistenti o cloud privati CloudEndure Disaster Recovery, disponibile tramite AWS Marketplace, consente alle organizzazioni di impostare una strategia di disaster recovery automatizzata su AWS. CloudEndure supporta inoltre il disaster recovery tra regioni e zone di disponibilità in AWS.

## Risorse

### Video

- [AWS re:Invent 2019: soluzioni di backup e ripristino e disaster recovery con AWS \(STG208\)](#)

### Documentazione

- [Cos'è AWS Backup?](#)
- [Correggere risorse AWS non conformi con le regole di AWS Config](#)
- [AWS Systems Manager Automation](#)
- AWS CloudFormation: [rilevamento delle deviazioni su un intero stack CloudFormation](#)
- [Amazon RDS: copia di backup tra regioni](#)
- [RDS: replica di una replica di lettura tra regioni](#)
- [S3: replica tra regioni](#)
- [Route 53: configurazione del failover DNS](#)
- [CloudEndure Disaster Recovery](#)
- In che modo è possibile implementare una soluzione di [gestione della configurazione dell'infrastruttura](#) in AWS?
- [CloudEndure Disaster Recovery in AWS](#)
- [AWS Marketplace: prodotti utilizzabili per disaster recovery](#)
- [Partner APN: partner che possono assistere con disaster recovery](#)

## Implementazioni di esempio per obiettivi di disponibilità

In questa sezione, esamineremo i progetti del carico di lavoro utilizzando la distribuzione di una tipica applicazione Web che consiste in un reverse proxy, contenuto statico su Amazon S3, un server dell'applicazione e un database SQL per l'archiviazione permanente dei dati. Per ogni target di disponibilità, forniremo un'implementazione di esempio. Questo carico di lavoro potrebbe invece utilizzare container o AWS Lambda per l'elaborazione e NoSQL (ad esempio Amazon DynamoDB) per il database, ma gli approcci sono simili. In ogni scenario, illustriamo come raggiungere gli obiettivi di disponibilità attraverso la progettazione del carico di lavoro per questi argomenti:

Argomento	Per ulteriori informazioni, consulta questa sezione
Monitoraggio delle risorse	<a href="#">Monitora le risorse del carico di lavoro</a>
Adattarsi alle modifiche nella domanda	<a href="#">Progetta il carico di lavoro per adattarti alle variazioni della domanda</a>
Implementazione della modifica	<a href="#">Implementa la modifica</a>
Esegui il backup dei dati	<a href="#">Esegui il backup dei dati</a>
Architettura mirata alla resilienza	<a href="#">Utilizza l'isolamento dei guasti per proteggere il tuo carico di lavoro</a> <a href="#">Progetta il carico di lavoro per resistere ai guasti dei componenti</a>
Testa la resilienza	<a href="#">Testa l'affidabilità</a>
Pianificazione per il disaster recovery (DR)	<a href="#">Pianifica il disaster recovery (DR)</a>

## Selezione delle dipendenze

Abbiamo scelto di utilizzare Amazon EC2 per le nostre applicazioni. Mostriamo come l'utilizzo di Amazon RDS e di più zone di disponibilità migliora la disponibilità delle nostre applicazioni. Useremo Amazon Route 53 come DNS. Quando utilizzeremo più zone di disponibilità, utilizzeremo Elastic Load Balancing. Amazon S3 viene utilizzato per il backup e per il contenuto statico. Dal momento che progettiamo per raggiungere un'affidabilità più elevata, dobbiamo utilizzare servizi dotati anch'essi di una disponibilità più elevata. Consulta l'[Appendice A: disponibilità prevista per i servizi AWS Select](#) per gli obiettivi di progettazione dei rispettivi servizi AWS.

## Scenari a regione singola

### Scenario a due 9 (99%)

Questi carichi di lavoro sono utili per l'azienda, ma è solo un *inconveniente* se non sono disponibili. Questo tipo di carico di lavoro può essere strumentazione interna, gestione interna delle conoscenze o monitoraggio dei progetti. Oppure può trattarsi di carichi di lavoro effettivi rivolti al cliente ma serviti da un servizio sperimentale, con un interruttore funzionale in grado di nascondere il servizio, se necessario.

Queste applicazioni possono essere distribuite con una regione e una zona di disponibilità.

#### Monitoraggio delle risorse

Avremo un monitoraggio semplice, indicando se la home page del servizio sta restituendo uno stato HTTP 200 (OK). Quando si verificano problemi, il nostro playbook indicherà che il log dall'istanza verrà utilizzato per stabilire la causa principale.

#### Adattarsi alle modifiche nella domanda

Avremo playbook per i guasti hardware comuni, gli aggiornamenti software urgenti e altre modifiche che causano interruzioni.

#### Implementazione delle modifiche

Utilizzeremo AWS CloudFormation per definire la nostra infrastruttura come codice e in particolare per accelerare la ricostruzione in caso di guasto.

Gli aggiornamenti del software verranno eseguiti manualmente utilizzando un runbook, con tempi di inattività richiesti per l'installazione e il riavvio del servizio. Se si verifica un problema durante la distribuzione, il runbook descrive come ripristinare la versione precedente.

Eventuali correzioni dell'errore vengono effettuate utilizzando l'analisi dei log da parte dei team operativi e di sviluppo e la correzione viene distribuita dopo aver completato e stabilito la priorità della soluzione.

#### Esegui il backup dei dati

Utilizzeremo un fornitore o una soluzione di backup appositamente creata per inviare dati di backup crittografati ad Amazon S3 utilizzando un runbook. Verificheremo che i backup funzionino ripristinando i dati e garantendo la possibilità di usarli su base regolare utilizzando un runbook. Configuriamo il controllo delle versioni sui nostri oggetti Amazon S3 e rimuoveremo le autorizzazioni per l'eliminazione dei backup. Usiamo una politica del ciclo di vita del bucket Amazon S3 per archiviare o eliminare definitivamente i dati in base ai nostri requisiti.

#### Architettura mirata alla resilienza

I carichi di lavoro vengono distribuiti con una regione e una zona di disponibilità. Distribuiamo l'applicazione, incluso il database, in un'unica istanza.

## Testa la resilienza

È prevista una pipeline di distribuzione del nuovo software, con alcuni test unitari, ma si tratta principalmente di test white-box/black-box del carico di lavoro assemblato.

## Pianificazione in funzione del disaster recovery (DR)

In caso di guasto, attendiamo il completamento dell'errore, con la possibilità di instradare le richieste a un sito Web statico utilizzando la modifica DNS tramite un runbook. Il tempo di ripristino sarà determinato dalla velocità con cui l'infrastruttura può essere distribuita e il database può essere ripristinato al backup più recente. Questa distribuzione può trovarsi nella stessa zona di disponibilità o in una zona di disponibilità diversa in caso di guasto della zona di disponibilità mediante un runbook.

## Obiettivo di progettazione della disponibilità

Servono 30 minuti per comprendere e decidere di eseguire il ripristino, 10 minuti per distribuire l'intero stack in AWS CloudFormation, assumendo di distribuire in una nuova zona di disponibilità e che il database possa essere ripristinato in 30 minuti. Questo implica che ci vogliono circa 70 minuti per ripristinare un guasto. Supponendo un guasto per trimestre, il nostro tempo di impatto stimato per l'anno è di 280 minuti, ovvero quattro ore e 40 minuti.

Ciò significa che il limite massimo di disponibilità è del 99,9%. La disponibilità effettiva dipenderà anche dal tasso reale di guasto, dalla durata dell'errore e dalla rapidità con cui ogni guasto viene di fatto ripristinato. Per questa architettura prevediamo che l'applicazione sia offline durante gli aggiornamenti (stima di 24 ore all'anno: quattro ore per update, sei volte all'anno), oltre agli eventi reali. Quindi, facendo riferimento alla tabella sulla disponibilità delle applicazioni più sopra nel whitepaper, vediamo che il nostro **obiettivo di progettazione della disponibilità** è del 99%.

## Riepilogo

Argomento	Implementazione
Monitoraggio delle risorse	Solo controllo dello stato del sito; nessun avviso.
Adattarsi alle modifiche nella domanda	Ridimensionamento verticale tramite ridistribuzione.
Implementazione della modifica	Runbook per distribuzione e rollback.
Esegui il backup dei dati	Runbook per backup e ripristino.
Architettura mirata alla resilienza	Ricostruzione completa; ripristino tramite backup.
Testa la resilienza	Ricostruzione completa; ripristino tramite backup.
Pianificazione per il disaster recovery (DR)	Backup crittografati, ripristino in una zona di disponibilità diversa, se necessario.

## Scenario a tre 9 (99,9%)

Il prossimo obiettivo di disponibilità è per le applicazioni per le quali è importante essere altamente disponibili, ma che possono tollerare brevi periodi di indisponibilità. Questo tipo di carico di lavoro viene in genere utilizzato per le operazioni interne che hanno un effetto sui dipendenti quando non sono attivi. Questo tipo di carico di lavoro può essere utilizzato anche per sistemi rivolti al cliente ma non costituisce un'entrata elevata per l'azienda e può tollerare tempi di ripristino o punti di ripristino più lunghi. Tali carichi di lavoro includono applicazioni amministrative per la gestione degli account o delle informazioni.

Possiamo migliorare la disponibilità per i carichi di lavoro utilizzando due zone di disponibilità per la nostra distribuzione e separando le applicazioni in livelli separati.

### Monitoraggio delle risorse

Il monitoraggio verrà ampliato per avvisare della disponibilità del sito Web soprattutto controllando lo stato HTTP 200 (OK) nella home page. Inoltre, ci saranno avvisi su ogni sostituzione di un server Web e quando si verifica un failover del database. Monitoreremo anche la disponibilità dei contenuti statici su Amazon S3 e avviseremo in caso di mancata disponibilità. I log verranno aggregati per facilitare la gestione e per aiutare nell'analisi della causa principale.

### Adattarsi alle modifiche nella domanda

La scalabilità automatica è configurata per monitorare l'utilizzo della CPU sulle istanze EC2 e aggiungere o rimuovere istanze per mantenere il target CPU al 70%, ma con non meno di un'istanza EC2 per zona di disponibilità. Se i modelli di caricamento sull'istanza RDS indicano che è necessario aumentare il dimensionamento, il tipo di istanza verrà modificato durante una finestra di manutenzione.

### Implementazione delle modifiche

Le tecnologie di distribuzione dell'infrastruttura rimangono le stesse dello scenario precedente.

La distribuzione di nuovo software ha una scadenza fissa ogni due o quattro settimane. Gli aggiornamenti del software saranno automatizzati, senza utilizzare modelli di distribuzione canary o blue/green, ma piuttosto utilizzando la sostituzione in loco. La decisione di eseguire il rollback verrà presa utilizzando il runbook.

Avremo playbook per stabilire la causa principale dei problemi. Dopo che la causa principale è stata identificata, la correzione dell'errore sarà identificata da una combinazione di team operativi e di sviluppo. La correzione verrà distribuita dopo aver sviluppato la correzione.

### Esegui il backup dei dati

Il backup e il ripristino possono essere eseguiti utilizzando Amazon RDS. Verrà eseguito regolarmente utilizzando un runbook per garantire che possiamo soddisfare i requisiti di ripristino.

## Architettura mirata alla resilienza

Possiamo migliorare la disponibilità per le applicazioni utilizzando due zone di disponibilità per la nostra distribuzione e separando le applicazioni in livelli diversi. Utilizzeremo servizi che funzionano su più zone di disponibilità, come Elastic Load Balancing, Auto Scaling e Amazon RDS ad AZ multiple con archiviazione crittografata tramite AWS Key Management Service. Ciò garantirà la tolleranza ai guasti a livello di risorsa e a livello di zona di disponibilità.

Il sistema di bilanciamento del carico indirizzerà il traffico solo verso istanze di applicazioni integre. Il controllo dello stato deve essere sul piano dati/livello dell'applicazione che indica la capacità dell'applicazione sull'istanza. Questa verifica non dovrebbe essere sul piano di controllo. Un URL di controllo dello stato per l'applicazione Web sarà presente e configurato per l'uso da parte del sistema di bilanciamento del carico e di Auto Scaling, in modo che le istanze guaste vengano rimosse e sostituite. Amazon RDS gestirà il motore di database attivo in modo che sia disponibile nella seconda zona di disponibilità se l'istanza fallisce nella zona di disponibilità primaria, quindi effettuerà il ripristino con la stessa resilienza.

Dopo aver separato i livelli, è possibile utilizzare modelli di resilienza distribuiti per aumentare l'affidabilità in modo che l'applicazione possa essere ancora disponibile anche quando il database è temporaneamente non disponibile durante un failover della zona di disponibilità.

## Testa la resilienza

I test funzionali vengono eseguiti come nello scenario precedente. Non testiamo le funzionalità di correzione automatica di ELB, scalabilità automatica o failover RDS.

Avremo playbook per problemi di database comuni, incidenti relativi alla sicurezza e distribuzioni non riuscite.

## Pianificazione in funzione del disaster recovery (DR)

Esistono runbook per il ripristino totale del carico di lavoro e report comuni. Il ripristino utilizza i backup archiviati nella stessa regione del carico di lavoro.

## Obiettivo di progettazione della disponibilità

Partiamo dal presupposto che almeno alcuni guasti richiederanno una decisione manuale per eseguire il ripristino. Tuttavia, con una maggiore automazione in questo scenario, supponiamo che solo due eventi all'anno richiedano questa decisione. Dedichiamo 30 minuti per decidere di eseguire il ripristino e supponiamo che il ripristino sia completato entro 30 minuti. Ciò implica 60 minuti per il ripristino del guasto. Supponendo due incidenti all'anno, il nostro tempo di impatto stimato per l'anno è di 120 minuti.

Ciò significa che il limite massimo di disponibilità è del 99,95%. La disponibilità effettiva dipenderà anche dal tasso reale di errore, dalla durata del guasto e dalla rapidità con cui ciascun componente viene ripristinato nella pratica. Per questa architettura è necessario che l'applicazione sia brevemente offline per gli aggiornamenti, ma questi aggiornamenti sono automatizzati. Stimiamo 150 minuti all'anno per questo: 15 minuti per update, 10 volte all'anno. Ciò aggiunge fino a 270 minuti all'anno quando il servizio non è disponibile, quindi il nostro **obiettivo di progettazione della disponibilità** è del 99,9%.

**Riepilogo**

<b>Argomento</b>	<b>Implementazione</b>
Monitoraggio delle risorse	Solo controllo dello stato del sito; avvisi inviati quando inattivo.
Adattarsi alle modifiche nella domanda	ELB per il livello di applicazione Web e di scalabilità automatica; ridimensionamento di RDS ad AZ multiple.
Implementazione della modifica	Distribuzione automatica in loco e runbook per il rollback.
Esegui il backup dei dati	Backup automatici tramite RDS per soddisfare RPO e runbook per il ripristino.
Architettura mirata alla resilienza	Scalabilità automatica per fornire un livello di applicazione e web autorigeneranti; RDS ad AZ multiple.
Testa la resilienza	ELB e applicazione autorigeneranti; RDS ad AZ multiple; nessun test esplicito.
Pianificazione per il disaster recovery (DR)	Backup crittografati tramite RDS nella stessa regione AWS.

**Scenario a quattro 9 (99,99%)**

Questo obiettivo di disponibilità per le applicazioni richiede che l'applicazione sia altamente disponibile e tollerante ai guasti dei componenti. L'applicazione deve essere in grado di assorbire i guasti senza la necessità di ottenere risorse aggiuntive. Questo obiettivo di disponibilità è per le applicazioni mission critical che sono fonte di entrate principali o significative per un'azienda, come un sito di e-commerce, un servizio web business-to-business o un sito di contenuti/media ad alto traffico.

Possiamo migliorare ulteriormente la disponibilità utilizzando un'architettura *staticamente stabile* all'interno della regione. Questo obiettivo di disponibilità non richiede una modifica del piano di controllo nel comportamento del nostro carico di lavoro per tollerare il guasto. Ad esempio, dovrebbe esserci una capacità sufficiente per resistere alla perdita di una zona di disponibilità. Non dovremmo richiedere aggiornamenti al DNS di Amazon Route 53. Non dovremmo aver bisogno di creare alcuna nuova infrastruttura, che si tratti di creare o modificare un bucket S3, creare nuove policy IAM (o modifiche alle policy) o modificare le configurazioni delle attività di Amazon ECS.

## Monitoraggio delle risorse

Il monitoraggio includerà parametri di successo e avvisi quando si verificano problemi. Inoltre, ci saranno avvisi su ogni sostituzione di un server Web, quando si verifica un failover del database e quando si verifica un errore di zona di disponibilità.

## Adattarsi alle modifiche nella domanda

Utilizzeremo Amazon Aurora come RDS, che consente la scalabilità automatica delle repliche di lettura. Per queste applicazioni, anche la progettazione per la disponibilità in lettura rispetto alla disponibilità in scrittura del contenuto primario è una decisione chiave dell'architettura. Aurora può anche aumentare automaticamente lo storage in base alle esigenze, con incrementi di 10 GB fino a 64 TB.

## Implementazione delle modifiche

Distribuiremo gli aggiornamenti utilizzando le distribuzioni canary o blue/green in ciascuna zona di isolamento singolarmente. Le distribuzioni sono completamente automatizzate, incluso un rollback se gli indicatori KPI indicano un problema.

Esistono runbook per requisiti di reportistica rigorosi e monitoraggio delle prestazioni. Se le operazioni riuscite tendono a non raggiungere gli obiettivi di prestazione o disponibilità, verrà utilizzato un playbook per stabilire cosa sta causando quella tendenza. Esistono dei manuali per le modalità di errore non scoperte e gli incidenti di sicurezza. Esistono anche playbook per stabilire la causa principale dei guasti. Ci impegneremo anche con AWS Support per l'offerta di Infrastructure Event Management.

Il team che crea e fa funzionare il sito Web identificherà la correzione per qualsiasi guasto imprevisto e darà la priorità alla correzione da distribuire dopo che è stata implementata.

## Esegui il backup dei dati

Il backup e il ripristino possono essere eseguiti utilizzando Amazon RDS. Verrà eseguito regolarmente utilizzando un runbook per garantire che possiamo soddisfare i requisiti di ripristino.

## Architettura mirata alla resilienza

Consigliamo tre zone di disponibilità per questo approccio. Utilizzando una distribuzione con tre zone di disponibilità, ciascuna zona di disponibilità ha una capacità statica del 50% di picco. Potrebbero essere utilizzate due zone di disponibilità, ma il costo della capacità staticamente stabile sarebbe maggiore perché entrambe le zone di disponibilità dovrebbero avere il 100% della capacità di picco. Aggiungeremo Amazon CloudFront per fornire la memorizzazione geografica nella cache, oltre a richiedere una riduzione sul piano dati della nostra applicazione.

Utilizzeremo Amazon Aurora come RDS e distribuiremo repliche di lettura in tutte e tre le zone.

L'applicazione verrà creata utilizzando i modelli di resilienza software/applicazione in tutti i livelli.

### **Testa la resilienza**

La pipeline di distribuzione avrà una suite di test completa, inclusi test di prestazioni, carico e inserimento degli errori.

Eserciteremo costantemente le nostre procedure di ripristino dei guasti durante le giornate di gioco, utilizzando i runbook per assicurarci di poter eseguire le attività e non deviare dalle procedure. Il team che costruisce il sito Web gestisce anche il sito Web.

### **Pianificazione in funzione del disaster recovery (DR)**

Esistono runbook per il ripristino totale del carico di lavoro e report comuni. Il ripristino utilizza i backup archiviati nella stessa regione del carico di lavoro. Le procedure di ripristino vengono regolarmente esercitate come parte delle giornate di gioco.

### **Obiettivo di progettazione della disponibilità**

Partiamo dal presupposto che almeno alcuni guasti richiederanno una decisione manuale per eseguire il ripristino, tuttavia con una maggiore automazione in questo scenario ipotizziamo che solo due eventi all'anno richiedano questa decisione e le azioni di ripristino saranno rapide. Dedichiamo 10 minuti per decidere di eseguire il ripristino e supponiamo che il ripristino sia completato entro cinque minuti. Ciò implica 15 minuti per il ripristino del guasto. Supponendo due guasti all'anno, il nostro tempo di impatto stimato per l'anno è di 30 minuti.

Ciò significa che il limite massimo di disponibilità è del 99,99%. La disponibilità effettiva dipenderà anche dal tasso reale di errore, dalla durata del guasto e dalla rapidità con cui ciascun componente viene ripristinato nella pratica. Per questa architettura ipotizziamo che l'applicazione sia costantemente online durante gli aggiornamenti. Sulla base di questo, il nostro **obiettivo di progettazione della disponibilità** è del 99,99%.

**Riepilogo**

<b>Argomento</b>	<b>Implementazione</b>
Monitoraggio delle risorse	Controlli di integrità a tutti i livelli e sugli indicatori KPI; avvisi inviati quando vengono attivati gli allarmi configurati; avvisi su tutti i guasti. Le riunioni operative sono rigorose per rilevare le tendenze e riuscire a creare obiettivi.
Adattarsi alle modifiche nella domanda	ELB per il livello applicativo di scalabilità automatica e Web; storage di scalabilità automatica e repliche di lettura in più zone per Aurora RDS.
Implementazione della modifica	Distribuzione automatizzata tramite canary o blue/green e rollback automatico quando gli indicatori KPI o avvisi indicano problemi non rilevati nell'applicazione. Le distribuzioni vengono effettuate per zona di isolamento.
Esegui il backup dei dati	Backup automatizzati tramite RDS per soddisfare RPO e ripristino automatico che viene praticato regolarmente in una giornata di gioco.
Architettura mirata alla resilienza	Implementate zone di isolamento degli errori per l'applicazione; ridimensionamento automatico per fornire livelli di applicazione e web autorigeneranti; RDS con Multi-AZ.
Testa la resilienza	La verifica dei guasti delle componenti e della zona di isolamento è nella pipeline e viene effettuata regolarmente con il personale operativo in una giornata di gioco; esistono playbook per diagnosticare problemi sconosciuti; ed esiste un processo di analisi della causa principale.
Pianificazione per il disaster recovery (DR)	Backup crittografati tramite RDS nella stessa regione AWS effettuati in una giornata di gioco.

**Scenari multi regione**

L'implementazione della nostra applicazione in più regioni AWS aumenterà i costi operativi, in parte perché isoliamo le regioni per mantenere la loro indipendenza. Seguire questo approccio dovrebbe essere una decisione molto ponderata. Detto questo, le regioni offrono un confine di isolamento forte e facciamo del nostro meglio per evitare guasti correlati tra

le regioni. L'uso di più aree ti consentirà di avere un maggiore controllo sui tempi di ripristino in caso di un grave errore di dipendenza in un servizio AWS regionale. In questa sezione, discuteremo di vari modelli di implementazione e della loro tipica disponibilità.

## **Scenario a tre 9 e ½ (99,95%) con un tempo di ripristino compreso tra 5 e 30 minuti**

Questo obiettivo di disponibilità per le applicazioni richiede tempi di inattività estremamente brevi e una perdita di dati molto ridotta per periodi specifici. Le applicazioni con questo obiettivo di disponibilità includono applicazioni nei settori: bancario, investimenti, servizi di emergenza e acquisizione dei dati. Queste applicazioni hanno tempi di ripristino e punti di ripristino molto brevi.

Possiamo migliorare ulteriormente i tempi di ripristino utilizzando un approccio *Warm Standby* in due regioni AWS. Distribuiremo l'intero carico di lavoro in entrambe le regioni, con il nostro sito passivo ridimensionato e mantenendo la coerenza finale di tutti i dati. Entrambe le distribuzioni saranno *stabili staticamente* nelle rispettive regioni. Le applicazioni devono essere create utilizzando i modelli di resilienza del sistema distribuito. Dovremo creare un componente di *instradamento* leggero che monitori lo stato del carico di lavoro e, se necessario, possa essere configurato per instradare il traffico verso la regione passiva.

### **Monitoraggio delle risorse**

Inoltre, ci saranno avvisi per ogni sostituzione di un server Web, quando si verifica un failover del database e quando si verifica un errore di regione. Monitoreremo anche la disponibilità dei contenuti statici su Amazon S3 e avviseremo in caso di mancata disponibilità. I log verranno aggregati per facilitare la gestione e per aiutare nell'analisi della causa principale in ciascuna regione.

Il componente di instradamento monitora sia lo stato della nostra applicazione sia le dipendenze hard regionali che abbiamo.

### **Adattarsi alle modifiche nella domanda**

Uguale allo scenario a quattro 9.

### **Implementazione delle modifiche**

La distribuzione di nuovo software ha una scadenza fissa ogni due o quattro settimane. Gli aggiornamenti del software saranno automatizzati, utilizzando modelli di distribuzione canary o blue/green.

Esistono runbook per quando si verifica il failover della regione, per problemi comuni dei clienti che si verificano durante tali eventi e per report comuni.

Avremo playbook per problemi di database comuni, incidenti relativi alla sicurezza, distribuzioni non riuscite, problemi imprevisti dei clienti nel failover della regione e per stabilire la causa principale dei problemi. Dopo che la causa principale è stata identificata,

la correzione dell'errore sarà identificata da una combinazione di team operativi e di sviluppo e distribuita quando la correzione è sviluppata.

Ci impegneremo anche con l'AWS Support per l'offerta di Infrastructure Event Management.

### **Esegui il backup dei dati**

Come per lo scenario a quattro 9, usiamo backup RDS automatici e la funzione versioni multiple di S3. I dati vengono replicati automaticamente e in modo asincrono dal cluster Aurora RDS nella regione attiva alle repliche di lettura tra regioni nella regione passiva. La replica tra regioni S3 viene utilizzata per spostare automaticamente e in modo asincrono i dati dalla regione attiva a quella passiva.

### **Architettura mirata alla resilienza**

Come lo scenario a quattro 9, è possibile il failover regionale. Questa operazione viene gestita manualmente. Durante il failover, indirizzeremo le richieste a un sito Web statico utilizzando il failover DNS fino al ripristino nella seconda regione.

### **Testa la resilienza**

Come per lo scenario a quattro 9 più una convalida dell'architettura attraverso le giornate di gioco utilizzando i runbook. Inoltre, la correzione RCA ha la priorità rispetto al rilascio delle funzionalità per l'implementazione e la distribuzione immediate

### **Pianificazione in funzione del disaster recovery (DR)**

Il failover regionale viene gestito manualmente. Tutti i dati vengono replicati in modo asincrono. L'infrastruttura in *warm standby* viene ridimensionata. Questo può essere automatizzato utilizzando un flusso di lavoro eseguito su AWS Step Functions. Anche AWS Systems Manager (SSM) può essere d'aiuto con questa automazione, poiché è possibile creare documenti SSM che aggiornano i gruppi Auto Scaling e ridimensionano le istanze.

### **Obiettivo di progettazione della disponibilità**

Partiamo dal presupposto che almeno alcuni guasti richiederanno una decisione manuale per eseguire il ripristino, tuttavia con una maggiore automazione in questo scenario ipotizziamo che solo due eventi all'anno richiedano questa decisione. Impieghiamo 20 minuti per decidere di eseguire il ripristino e supponiamo che questo sia completato entro 10 minuti. Questo implica che ci vogliono circa 30 minuti per eseguire il ripristino da un guasto. Supponendo due incidenti all'anno, il nostro tempo di impatto stimato per l'anno è di 60 minuti.

Ciò significa che il limite massimo di disponibilità è del 99,95%. La disponibilità effettiva dipenderà anche dal tasso reale di errore, dalla durata del guasto e dalla rapidità con cui ciascun componente viene ripristinato nella pratica. Per questa architettura ipotizziamo che l'applicazione sia costantemente online durante gli aggiornamenti. Sulla base di questo, il nostro **obiettivo di progettazione della disponibilità** è del 99,95%.

### **Riepilogo**

Argomento	Implementazione
Monitoraggio delle risorse	Controlli di integrità a tutti i livelli, incluso lo stato del DNS a livello di regione AWS, e sugli indicatori KPI; avvisi inviati quando vengono attivati gli allarmi configurati; avvisi per tutti i guasti. Le riunioni operative sono rigorose per rilevare le tendenze e riuscire a creare obiettivi.
Adattarsi alle modifiche nella domanda	ELB per il livello applicativo di scalabilità automatica e Web; storage di scalabilità automatica e repliche di lettura in più zone nelle regioni attive e passive per Aurora RDS. Dati e infrastruttura sincronizzati tra regioni AWS per una stabilità statica.
Implementazione della modifica	Distribuzione automatizzata tramite canary o blue/green e rollback automatico quando gli indicatori KPI o gli avvisi indicano problemi non rilevati nell'applicazione, distribuzioni effettuate in una zona di isolamento in una regione AWS alla volta.
Esegui il backup dei dati	Backup automatizzati in ciascuna regione AWS tramite RDS per soddisfare RPO e ripristino automatico effettuato regolarmente in una giornata di gioco. I dati di Aurora RDS e S3 vengono replicati automaticamente e in modo asincrono da una regione attiva a una regione passiva.
Architettura mirata alla resilienza	Scalabilità automatica per fornire un livello di applicazione e web autorigeneranti; RDS ad AZ multiple; il failover regionale viene gestito manualmente con il sito statico mostrato durante il failover.

Testa la resilienza	La verifica dei guasti dei componenti e della zona di isolamento è nella pipeline e viene effettuata regolarmente con il personale operativo in una giornata di gioco; esistono playbook per diagnosticare problemi sconosciuti; ed esiste un processo di analisi della causa principale, con percorsi di comunicazione per individuare il problema e come questo è stato corretto o prevenuto. La correzione RCA ha la priorità rispetto al rilascio delle funzionalità per l'implementazione e la distribuzione immediate.
Pianificazione per il disaster recovery (DR)	Warm Standby distribuito in un'altra regione. L'infrastruttura viene ricalibrata utilizzando flussi di lavoro eseguiti attraverso AWS Step Functions o documenti AWS Systems Manager. Backup crittografati tramite RDS. Repliche di lettura tra regioni tra due regioni AWS. Replica tra regioni di asset statici in S3. Il ripristino è nella regione AWS attiva corrente, viene effettuato in una giornata di gioco ed è coordinato con AWS.

## Scenario a cinque 9 (99,999%) o superiore con un tempo di ripristino inferiore a 1 minuto

Questo obiettivo di disponibilità per le applicazioni richiede tempi di inattività e una perdita di dati pressoché nulli per periodi specifici. Le applicazioni che potrebbero avere questo obiettivo di disponibilità includono, ad esempio, alcune applicazioni bancarie, di investimento, finanziarie, governative e di business cruciali che costituiscono le attività principali di un'azienda che genera entrate estremamente elevate. L'obiettivo è di avere archivi di dati fortemente coerenti e completa ridondanza a tutti i livelli. Abbiamo selezionato un archivio dati basato su SQL. Tuttavia, in alcuni scenari, troveremo difficile ottenere un RPO molto basso. Se riesci a partizionare i tuoi dati è possibile che non ci siano perdite di dati. Ciò potrebbe richiedere l'aggiunta di logica e di latenza nell'applicazione per garantire la coerenza dei dati tra le posizioni geografiche, nonché la capacità di spostare o copiare i dati tra le partizioni. L'esecuzione di questo partizionamento potrebbe essere più semplice se si utilizza un database NoSQL.

Possiamo migliorare ulteriormente la disponibilità utilizzando un approccio *attivo/attivo* o *multi-master* in più regioni AWS. Il carico di lavoro verrà distribuito in tutte le regioni desiderate che sono *staticamente stabili* tra le regioni (in modo che le regioni rimanenti possano gestire il carico con la perdita di una regione). Un livello *di instradamento* indirizza il traffico verso le posizioni geografiche integre e modifica automaticamente la destinazione quando una posizione non è integra, interrompendo anche temporaneamente i livelli di replica dei dati. Amazon Route 53 offre controlli di integrità a intervalli di 10 secondi e offre anche TTL sui set di record fino a un secondo.

### **Monitoraggio delle risorse**

Come per lo scenario a 3½ 9, ma si verificano inoltre gli avvisi quando una regione viene rilevata come non integra e il traffico viene instradato da essa.

### **Adattarsi alle modifiche nella domanda**

Uguale allo scenario a 3½ 9.

### **Implementazione delle modifiche**

La pipeline di distribuzione avrà una suite di test completa, inclusi test di prestazioni, carico e inserimento dei guasti. Distribuiremo gli aggiornamenti utilizzando le distribuzioni canary o blue/green in una zona di isolamento alla volta, in una singola regione, prima di passare a un'altra. Durante la distribuzione, le versioni precedenti continueranno a essere in esecuzione su altre istanze per facilitare un rollback più veloce. Queste sono completamente automatizzate, incluso un rollback se gli indicatori KPI indicano un problema. Il monitoraggio includerà parametri di successo e avvisi quando si verificano problemi.

Esistono runbook per requisiti di reportistica rigorosi e monitoraggio delle prestazioni. Se le operazioni riuscite tendono a non raggiungere gli obiettivi di prestazione o disponibilità, verrà utilizzato un playbook per stabilire cosa sta causando la tendenza. Esistono dei manuali per le modalità di errore non scoperte e gli incidenti di sicurezza. Esistono anche playbook per stabilire la causa principale dei guasti.

Il team che costruisce il sito Web, lo gestisce anche. Quel team identificherà la correzione per qualsiasi guasto imprevisto e darà la priorità alla correzione da distribuire dopo che è stata implementata. Ci impegneremo anche con AWS Support per Infrastructure Event Management.

### **Esegui il backup dei dati**

Uguale allo scenario a 3½ 9.

### **Architettura mirata alla resilienza**

Le applicazioni devono essere create utilizzando i modelli di resilienza software/applicazione. È possibile che siano necessari molti altri livelli di routing per implementare la disponibilità necessaria. La complessità di questa implementazione aggiuntiva non deve essere sottovalutata. L'applicazione verrà implementata nelle zone di isolamento degli errori di distribuzione e

partizionata e distribuita in modo tale che anche un evento su tutta la regione non influirà su tutti i clienti.

### Testa la resilienza

Eserciteremo costantemente l'architettura durante le giornate di gioco, utilizzando i runbook per assicurarci di poter eseguire le attività e non deviare dalle procedure.

### Pianificazione in funzione del disaster recovery (DR)

Distribuzione multi-regione *attiva-attiva* con infrastruttura completa del carico di lavoro e dati in più regioni. Utilizzando una strategia globale di lettura locale e di scrittura, una regione è il database master per tutte le scritture e i dati vengono replicati per le letture in altre regioni. Se la regione database master ha esito negativo, sarà necessario promuovere un nuovo database. La strategia di lettura locale e di scrittura prevede l'assegnazione di utenti a una regione di origine in cui vengono gestite le scritture DB. Ciò consente agli utenti di leggere o scrivere da qualsiasi regione, ma richiede una logica complessa per gestire potenziali conflitti di dati tra scritture in regioni diverse.

Quando una regione viene rilevata come non integra, il livello di instradamento instrada automaticamente il traffico verso le regioni integre rimanenti. Non è richiesto alcun intervento manuale.

Gli archivi di dati devono essere replicati tra le regioni in modo da poter risolvere potenziali conflitti. Sarà necessario creare strumenti e processi automatizzati per copiare o spostare i dati tra le partizioni per motivi di latenza e per bilanciare richieste o quantità di dati in ciascuna partizione. Anche la risoluzione dei conflitti di dati richiederà runbook operativi aggiuntivi.

### Obiettivo di progettazione della disponibilità

Partiamo dal presupposto che vengono effettuati ingenti investimenti per automatizzare tutto il ripristino e che il ripristino può essere completato in un minuto. Non prevediamo ripristini attivati manualmente, ma fino a un'azione di ripristino automatizzata al trimestre. Ciò implica quattro minuti all'anno per il ripristino. Ipotizziamo che l'applicazione sia costantemente online durante gli aggiornamenti. Sulla base di questo, il nostro **obiettivo di progettazione della disponibilità** è del 99,999%.

### Riepilogo

Argomento	Implementazione
Monitoraggio delle risorse	Controlli di integrità a tutti i livelli, incluso lo stato del DNS a livello di regione AWS, e sugli indicatori KPI; avvisi inviati quando vengono attivati gli allarmi configurati; avvisi per tutti i guasti. Le riunioni operative sono rigorose per rilevare le tendenze e riuscire a creare obiettivi.
Adattarsi alle modifiche nella domanda	ELB per il livello applicativo di scalabilità automatica e Web; storage di scalabilità automatica e repliche di lettura in più zone nelle regioni attive e passive per Aurora RDS. Dati e infrastruttura sincronizzati tra regioni AWS per una stabilità statica.
Implementazione della modifica	Distribuzione automatizzata tramite canary o blue/green e rollback automatico quando gli indicatori KPI o gli avvisi indicano problemi non rilevati nell'applicazione, distribuzioni effettuate in una zona di isolamento in una regione AWS alla volta.
Esegui il backup dei dati	Backup automatizzati in ciascuna regione AWS tramite RDS per soddisfare RPO e ripristino automatico effettuato regolarmente in una giornata di gioco. I dati di Aurora RDS e S3 vengono replicati automaticamente e in modo asincrono da una regione attiva a una regione passiva.
Architettura mirata alla resilienza	Implementate zone di isolamento degli errori per l'applicazione; Auto Scaling per fornire livelli di applicazione e web autorigeneranti; RDS ad AZ multiple, failover regionale automatizzato.

Testa la resilienza	La verifica dei guasti dei componenti e della zona di isolamento è nella pipeline e viene effettuata regolarmente con il personale operativo in una giornata di gioco; esistono playbook per diagnosticare problemi sconosciuti; ed esiste un processo di analisi della causa principale, con percorsi di comunicazione per individuare il problema e come questo è stato corretto o prevenuto. La correzione RCA ha la priorità rispetto al rilascio delle funzionalità per l'implementazione e la distribuzione immediate.
Pianificazione per il disaster recovery (DR)	Configurazione attiva-attiva distribuita in almeno due regioni. L'infrastruttura è completamente ridimensionata e staticamente stabile tra le regioni. I dati vengono partizionati e sincronizzati tra regioni. Backup crittografati tramite RDS. Il guasto di una regione viene praticato in una giornata di gioco ed è coordinato con AWS. Durante il ripristino, potrebbe essere necessario promuovere un nuovo master del database.

## Risorse

### Documentazione

- [Amazon Builders' Library](#) - Come Amazon crea e gestisce software
- [Centro architetturale AWS](#)

### Corsi

- [I corsi sull'affidabilità di AWS Well-Architected](#)

### Collegamenti esterni

- Modello di accodamento adattivo: [errori su vasta scala](#)
- [Calcolo della disponibilità totale del sistema](#)

### Libri

- Robert S. Hammer "[Modelli per software con tolleranza ai guasti](#)"
- Andrew Tanenbaum e Marten van Steen "[Sistemi distribuiti: principi e paradigmi](#)"

## Conclusioni

Che tu non abbia alcuna esperienza di argomenti quali disponibilità e affidabilità o che tu sia un esperto in cerca di approfondimenti per massimizzare la disponibilità del tuo carico di lavoro mission critical, speriamo che questo whitepaper abbia stimolato il tuo modo di pensare, ti abbia offerto nuove idee o suggerito nuove possibili domande. Ci auguriamo che ciò ti porti a comprendere in modo più approfondito il giusto livello di disponibilità in base alle esigenze della tua azienda e come progettare l'affidabilità necessaria a raggiungerla. Ti invitiamo a trarre vantaggio dai consigli di progettazione, operativi e orientati al ripristino offerti qui, nonché dalla conoscenza e dall'esperienza dei Solution Architects AWS. Ci piacerebbe avere tue notizie, in particolare sulle tue storie di successo che hanno raggiunto livelli elevati di disponibilità su AWS. Contatta il team del tuo account o usa [Contattaci tramite il nostro sito Web](#).

## Collaboratori

I collaboratori di questo documento includono:

- Seth Eliot, Solutions Architect principale per l'affidabilità, Well-Architected, Amazon Web Services
- Adrian Hornsby, evangelista tecnologico principale, Architettura, Amazon Web Services
- Philip Fitzsimons, Direttore senior Well-Architected, Amazon Web Services
- Rodney Lester, responsabile affidabilità, Well-Architected Amazon Web Services
- Kevin Miller, Direttore sviluppo software, Amazon Web Services
- Shannon Richards, Direttore senior programma tecnico, Amazon Web Services

## Approfondimenti

Per ulteriori informazioni, consulta:

- [Canone di architettura AWS](#)

## Revisioni del documento

Data	Descrizione
<b>Aprile 2020</b>	<p>Aggiornamenti sostanziali e contenuti nuovi/revisionati, tra cui:</p> <ul style="list-style-type: none"> <li>• Aggiunta della sezione best practice "Architettura del carico di lavoro"</li> <li>• Riorganizzazione delle best practice nelle sezioni Gestione delle modifiche e Gestione dei guasti</li> <li>• Aggiornamento delle risorse</li> <li>• Aggiornamento per includere le risorse e i servizi AWS più recenti, ad esempio AWS Global Accelerator, le quote dei servizi AWS e AWS Transit Gateway</li> <li>• Aggiunta/aggiornamento delle definizioni di affidabilità, disponibilità e resilienza</li> <li>• Miglior allineamento del whitepaper allo strumento AWS Well-Architected Tool (domande e best practice) utilizzato per le recensioni del canone di architettura</li> <li>• Riordinamento dei principi di progettazione, spostando <b>Ripristino automatico del guasto</b> prima di <b>Test delle procedure di ripristino</b></li> <li>• Aggiornamento di diagrammi e formati per le equazioni</li> <li>• Eliminazione delle sezioni Servizi chiave e integrazione dei riferimenti ai servizi AWS chiave nelle best practice</li> </ul>
<b>Ottobre 2019</b>	Correzione link danneggiato
<b>Aprile 2019</b>	Appendice A aggiornata
<b>Settembre 2018</b>	Aggiunta di consigli specifici sulla rete AWS Direct Connect e di obiettivi di progettazione del servizio aggiuntivi
<b>Giugno 2018</b>	Aggiunta delle sezioni Principi di progettazione e Gestione dei limiti. Aggiornamento dei link, eliminazione ambiguità sulla terminologia upstream/downstream e aggiunta di riferimenti espliciti ai restanti argomenti del principio di base di affidabilità negli scenari di disponibilità.
<b>Marzo 2018</b>	Modificata la soluzione DynamoDB su più regioni con le tabelle globali DynamoDB
<b>Dicembre 2017</b>	Piccola correzione al calcolo della disponibilità per includere la disponibilità dell'applicazione

Data	Descrizione
<b>Novembre 2017</b>	Aggiornamento per fornire indicazioni su progetti ad alta disponibilità, inclusi concetti, best practice ed esempi di implementazione.
<b>Novembre 2016</b>	Prima pubblicazione

## Appendice A: disponibilità prevista per determinati servizi AWS

Di seguito indichiamo la disponibilità che determinati servizi AWS sono stati progettati per raggiungere. Questi valori non rappresentano un Contratto sul livello di servizio o una garanzia, ma forniscono piuttosto informazioni sugli obiettivi di progettazione di ciascun servizio. In alcuni casi, differenziamo parti del servizio in cui esiste una differenza significativa nell'obiettivo di progettazione della disponibilità. Questo elenco non è completo per tutti i servizi AWS e prevediamo di aggiornarlo periodicamente con informazioni su servizi aggiuntivi. Amazon CloudFront, Amazon Route 53 e il piano di controllo di Identity and Access Management forniscono un servizio globale e l'obiettivo di disponibilità dei componenti è dichiarato di conseguenza. Altri servizi forniscono servizi all'interno di una regione AWS e l'obiettivo di disponibilità è dichiarato di conseguenza. Molti servizi forniscono l'indipendenza tra le zone di disponibilità. In questi casi, forniamo l'obiettivo di progettazione della disponibilità per una singola zona di disponibilità e quando vengono utilizzate due (o più) zone di disponibilità.

NOTA: i numeri nella tabella seguente non si riferiscono alla durabilità (conservazione a lungo termine dei dati); sono numeri di disponibilità (accesso a dati o funzioni).

Servizio	Componente	Obiettivo di progettazione della disponibilità
Amazon API Gateway	Piano di controllo	99,950%
	Piano dati	99,990%
Amazon Aurora	Piano di controllo	99,950%
	Piano dati Singola AZ	99,950%
	Piano dati Multi AZ	99,990%
AWS CloudFormation	Servizio	99,950%
Amazon CloudFront	Piano di controllo	99,900%
	Piano dati (distribuzione di contenuti)	99,990%
Amazon CloudSearch	Piano di controllo	99,950%
	Piano dati	99,950%
Amazon CloudWatch	Parametri CW (servizio)	99,990%

<b>Servizio</b>	<b>Componente</b>	<b>Obiettivo di progettazione della disponibilità</b>
	Eventi CW (servizio)	99,990%
	Log CW (servizio)	99,950%
<b>AWS Database Migration Service</b>	Piano di controllo	99,900%
	Piano dati	99,950%
<b>AWS Data Pipeline</b>	Servizio	99,990%
<b>Amazon DynamoDB</b>	Servizio (standard)	99,990%
	Servizio (tablele globali)	99,999%
<b>Amazon EC2</b>	Piano di controllo	99,950%
	Piano dati Singola AZ	99,950%
	Piano dati Multi AZ	99,990%
<b>Amazon ElastiCache</b>	Servizio	99,990%
<b>Amazon Elastic Block Store</b>	Piano di controllo	99,950%
	Piano dati (disponibilità volumi)	99,999%
<b>Amazon ElasticSearch</b>	Piano di controllo	99,950%
	Piano dati	99,950%
<b>Amazon EMR</b>	Piano di controllo	99,950%
<b>Amazon S3 Glacier</b>	Servizio	99,900%
<b>AWS Glue</b>	Servizio	99,990%
<b>Amazon Kinesis Data Streams</b>	Servizio	99,990%
<b>Amazon Kinesis Data Firehose</b>	Servizio	99,900%
<b>Amazon Kinesis Video Streams</b>	Servizio	99,900%

Servizio	Componente	Obiettivo di progettazione della disponibilità
<b>Amazon Neptune</b>	Servizio	99,900%
<b>Amazon RDS</b>	Piano di controllo	99,950%
	Piano dati Singola AZ	99,950%
	Piano dati Multi AZ	99,990%
<b>Amazon Rekognition</b>	Servizio	99,980%
<b>Amazon Redshift</b>	Piano di controllo	99,950%
	Piano dati	99,950%
<b>Amazon Route 53</b>	Piano di controllo	99,950%
	Piano dati (risoluzione query)	100.000%
<b>Amazon SageMaker</b>	Piano dati (Model Hosting)	99,990%
	Piano di controllo	99,950%
<b>Amazon S3</b>	Servizio (standard)	99,990%
<b>AWS Auto Scaling</b>	Piano di controllo	99,900%
	Piano dati	99,990%
<b>AWS Batch</b>	Piano di controllo	99,900%
	Piano dati	99,950%
<b>AWS CloudHSM</b>	Piano di controllo	99,900%
	Piano dati Singola AZ	99,900%
	Piano dati Multi AZ	99,990%
<b>AWS CloudTrail</b>	Piano di controllo (config)	99,900%
	Piano dati (eventi di dati)	99,990%
	Piano dati (eventi di gestione)	99,999%

Servizio	Componente	Obiettivo di progettazione della disponibilità
<b>AWS Config</b>	Servizio	99,950%
<b>AWS Direct Connect</b>	Piano di controllo	99,900%
	Piano dati a posizione singola	99,900%
	Piano dati a posizione multipla	99,990%
<b>Amazon Elastic File System</b>	Piano di controllo	99,950%
	Piano dati	99,990%
<b>AWS Identity and Access Management</b>	Piano di controllo	99,900%
	Piano dati (autenticazione)	99,995%
<b>AWS IoT Core</b>	Servizio	99,900%
<b>AWS IoT Device Management</b>	Servizio	99,900%
<b>AWS IoT Greengrass</b>	Servizio	99,900%
<b>AWS Lambda</b>	Invocazione della funzione	99,950%
<b>AWS Secrets Manager</b>	Servizio	99,900%
<b>AWS Shield</b>	Piano di controllo	99,500%
	Piano dati (rilevamento)	99,000%
	Piano dei dati (mitigazione)	99,900%
<b>AWS Storage Gateway</b>	Piano di controllo	99,950%
	Piano dati	99,950%
<b>AWS X-Ray</b>	Piano di controllo (console)	99,900%
	Piano dati	99,950%
<b>EC2 Container Service</b>	Piano di controllo	99,900%
	EC2 Container Registry	99,990%

<b>Servizio</b>	<b>Componente</b>	<b>Obiettivo di progettazione della disponibilità</b>
	EC2 Container Service	99,990%
<b>Elastic Load Balancing</b>	Piano di controllo	99,950%
	Piano dati	99,990%
<b>Key Management System (KMS)</b>	Piano di controllo	99,990%
	Piano dati	99,995%