AWS Well-Architected Framework

Säule der Zuverlässigkeit

Säule der Zuverlässigkeit: AWS Well-Architected Framework

Table of Contents

Säule „Zuverlässigkeit“ – AWS Well-Architected Framework

Veröffentlichungsdatum: 6. November 2024 (Dokumentversionen)

Das vorliegende Dokument befasst sich schwerpunktmäßig mit der Säule „Zuverlässigkeit“ des AWS Well-Architected Framework. Es bietet Informationen für Kunden zur Anwendung von bewährten Methoden für die Konzeption, Bereitstellung und Wartung von Amazon Web Services (AWS)-Umgebungen.

Einführung

Das AWS-Well-Architected-Framework unterstützt Sie dabei, die Vor- und Nachteile der Entscheidungen nachzuvollziehen, die Sie beim Erstellen von Workloads in AWS treffen. Das Framework hilft Ihnen, bewährte Architekturmethoden für den Entwurf und Betrieb zuverlässiger, sicherer, effizienter, kostengünstiger und nachhaltiger Workloads in der Cloud zu ermitteln. Es bietet eine Möglichkeit, Ihre Architekturen konsistent auf die Einhaltung bewährter Methoden zu prüfen und Verbesserungspotenzial zu identifizieren. Wir sind der Meinung, dass eine Well-Architected-Workload-Architektur die Wahrscheinlichkeit für den geschäftlichen Erfolg deutlich erhöht.

Das AWS Well-Architected-Framework basiert auf sechs Säulen:

- Operational Excellence
- Sicherheit
- Zuverlässigkeit
- Leistungseffizienz
- Kostenoptimierung
- Nachhaltigkeit

Dieses Dokument legt den Fokus auf die Säule „Zuverlässigkeit“ und wie Sie diese Säule auf Ihre Lösungen anwenden können. Das Erreichen von Zuverlässigkeit kann in herkömmlichen On-Premises-Umgebungen aufgrund von Single Points of Failure und einer mangelnden Automatisierung und Elastizität eine große Herausforderung darstellen. Durch die Einführung der in diesem Dokument beschriebenen Methoden können Sie Architekturen entwickeln, die sich durch eine starke Grundlage, eine hohe Ausfallsicherheit, eine konsistente Änderungsverwaltung und bewährte Wiederherstellungsprozesse nach Fehlern auszeichnen.

Dieses Dokument richtet sich an Personen in technologischen Funktionen wie Chief Technology Officers (CTOs), Architekten, Entwickler und Mitglieder des Betriebsteams. Nach der Lektüre dieses Dokuments sind Sie mit den bewährten Methoden und Strategien von AWS für die Entwicklung zuverlässiger Cloud-Architekturen vertraut. Dieses Dokument enthält allgemeine Implementierungsdetails und Architekturmodelle sowie Referenzen zu weiteren Ressourcen.

Zuverlässigkeit

Die Säule „Zuverlässigkeit“ umfasst die Fähigkeit einer Workload, die beabsichtigte Funktion erwartungsgemäß korrekt und konsistent auszuführen. Dies umfasst die Möglichkeit, die Workload während des gesamten Lebenszyklus zu betreiben und zu testen. Dieses Dokument bietet umfassende Informationen mit bewährten Methoden für die Implementierung zuverlässiger Workloads in AWS.

Themen

Modell der geteilten Verantwortung für Ausfallsicherheit

Die Ausfallsicherheit ist eine geteilte Verantwortung zwischen AWS und Ihnen. Sie sollten unbedingt wissen, wie die Notfallwiederherstellung (DR) und Verfügbarkeit als Teil der Ausfallsicherheit im Rahmen dieses gemeinsamen Modells funktionieren.

AWS-Verantwortung – Ausfallsicherheit der Cloud

AWS ist für die Ausfallsicherheit der Infrastruktur verantwortlich, über die alle in AWS Cloud angebotenen Services ausgeführt werden. Diese Infrastruktur umfasst die Hardware, Software, Netzwerke und Einrichtungen, die die AWS Cloud-Services ausführen. AWS unternimmt wirtschaftlich vertretbare Anstrengungen, um diese AWS Cloud-Services verfügbar zu halten und sicherzustellen, dass die Verfügbarkeit der Services die Service Level Agreements (SLAs) von AWS erfüllt oder übertrifft.

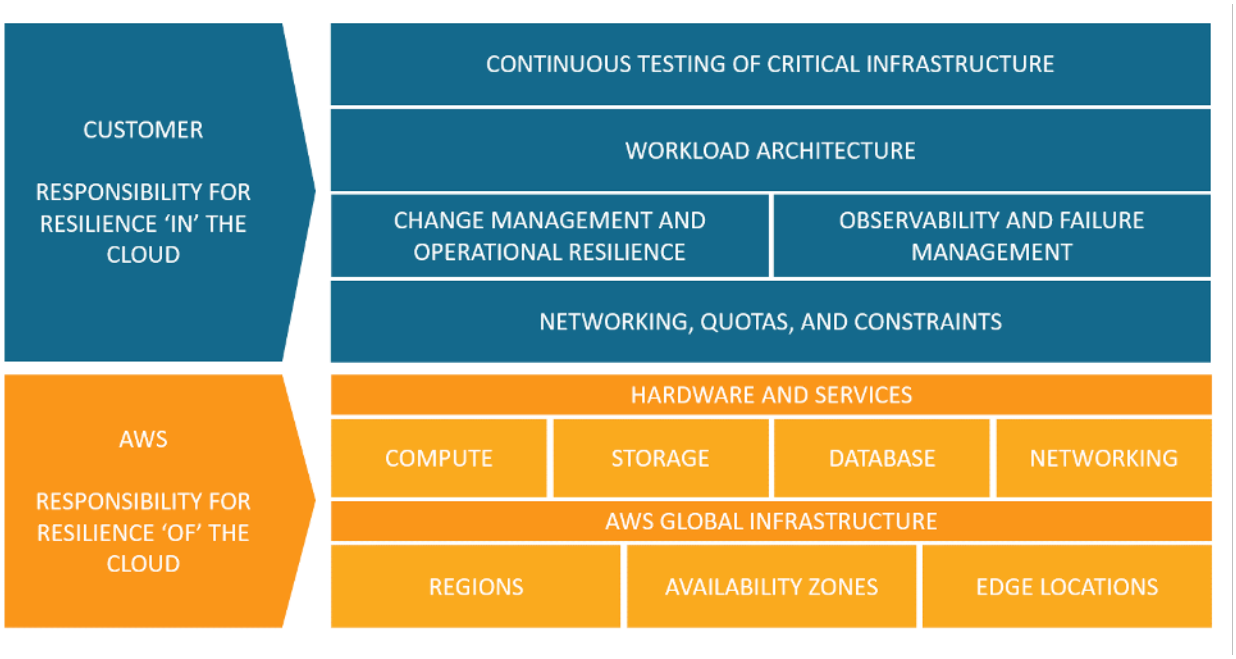
Die globale Cloud-Infrastruktur von AWS ist so konzipiert, dass Kunden hochgradig widerstandsfähige Workload-Architekturen erstellen können. Jede AWS-Region-Region ist vollständig isoliert und besteht aus mehreren Availability Zones, bei denen es sich um physisch isolierte Partitionen der Infrastruktur handelt. Availability Zones isolieren Fehler, die die Ausfallsicherheit von Workloads beeinträchtigen könnten, und verhindern, dass sie sich auf andere Zonen in der Region auswirken. Gleichzeitig sind alle Zonen in einer AWS-Region mit einem Netzwerk mit hoher Bandbreite und geringer Latenz verbunden, und zwar über vollständig redundante, dedizierte Metro-

Glasfaserverbindungen, die einen hohen Durchsatz und eine geringe Latenz zwischen den Zonen ermöglichen. Der gesamte Datenverkehr zwischen den Zonen ist verschlüsselt. Die Leistung des Netzwerks ist ausreichend, um eine synchrone Replikation zwischen den Zonen zu ermöglichen. Wenn eine Anwendung auf mehrere AZs aufgeteilt wird, sind Unternehmen besser isoliert und vor Problemen wie Stromausfällen, Blitzeinschlägen, Tornados, Wirbelstürmen und mehr geschützt.

Kundenverantwortung – Ausfallsicherheit in der Cloud

Ihre Verantwortung wird von den AWS Cloud-Services bestimmt, die Sie auswählen. Dies bestimmt den Umfang der Konfigurationsarbeit, die Sie als Teil Ihrer Verantwortung für die Ausfallsicherheit durchführen müssen. Bei einem Service wie Amazon Elastic Compute Cloud (Amazon EC2) muss der Kunde zum Beispiel alle notwendigen Aufgaben zur Konfiguration und Verwaltung der Ausfallsicherheit übernehmen. Kunden, die Amazon-EC2-Instances bereitstellen, sind für die Bereitstellung von Amazon-EC2-Instances an mehreren Standorten (wie AWS Availability Zones), die Implementierung der Selbstreparatur mit Services wie Auto Scaling sowie die Verwendung von bewährten Methoden für eine ausfallsichere Workload-Architektur für Anwendungen, die auf den Instances installiert sind, verantwortlich. Für verwaltete Services wie Amazon S3 und Amazon DynamoDB betreibt AWS die Infrastrukturebene, das Betriebssystem und die Plattformen. Kunden greifen auf die Endpunkte zu, um Daten zu speichern und abzurufen. Sie sind dafür verantwortlich, die Ausfallsicherheit Ihrer Daten zu verwalten, einschließlich Sicherungs-, Versionsverwaltungs- und Replikationsstrategien.

Das Bereitstellen Ihrer Workload in mehreren Availability Zones in einer AWS-Region ist Teil einer Hochverfügbarkeitsstrategie, die darauf abzielt, Workloads zu schützen, indem Probleme auf eine Availability Zone beschränkt werden. Die Redundanz der anderen Availability Zones wird genutzt, um Anfragen weiterhin zu bedienen. Eine Multi-AZ-Architektur ist außerdem Teil einer Notfallwiederherstellungsstrategie, die darauf abzielt, Workloads besser zu isolieren und vor Problemen wie Stromausfällen, Blitzeinschlägen, Tornados, Erdbeben und anderen Ereignissen zu schützen. Notfallwiederherstellungsstrategien können auch auf mehrere AWS-Regionen zurückgreifen. In einer Aktiv/Passiv-Konfiguration wird der Service für die Workload beispielsweise von der aktiven Region auf die Notfallwiederherstellungsregion übertragen, wenn die aktive Region die Anfragen nicht mehr bedienen kann.



Verantwortung für die Ausfallsicherheit in der Cloud und der Cloud selbst für Kunden und AWS

Sie können AWS-Services nutzen, um Ihre Ausfallsicherheitsziele zu erreichen. Als Kunde sind Sie für die Verwaltung der folgenden Aspekte Ihres Systems verantwortlich, um die Ausfallsicherheit in der Cloud zu erreichen. Weitere Informationen zu den einzelnen Services finden Sie in der AWS-Dokumentation.

Netzwerke, Kontingente und Beschränkungen

- Bewährte Methoden für diesen Bereich des Modells der geteilten Verantwortung werden unter Grundlagen detailliert beschrieben.
- Planen Sie Ihre Architektur mit ausreichendem Spielraum zum Skalieren. Informieren Sie sich über die Service Quotas und die Beschränkungen der genutzten Services unter Berücksichtigung der erwarteten Zunahme der Last, falls zutreffend.
- Entwerfen Sie Ihre Netzwerktopologie so, dass sie hochverfügbar, redundant und skalierbar ist.

Änderungsmanagement und operative Ausfallsicherheit

- Das Änderungsmanagement umfasst die Einführung und Verwaltung von Änderungen in Ihrer Umgebung. Die Implementierung von Änderungen erfordert die Erstellung und Aktualisierung von Runbooks und Bereitstellungsstrategien für Ihre Anwendung und Infrastruktur.

- Eine belastbare Strategie zur Überwachung von Workload-Ressourcen berücksichtigt alle Komponenten, einschließlich technischer und geschäftlicher Metriken, Benachrichtigungen, Automatisierung und Analysen.
- Workloads in der Cloud müssen sich an Veränderungen der Nachfrage anpassen und als Reaktion auf Beeinträchtigungen oder Schwankungen in der Nutzung skalieren.

Beobachtbarkeit und Ausfallmanagement

- Die Beobachtung von Ausfällen durch Überwachung ist erforderlich, um die Selbstreparatur zu automatisieren, damit Ihre Workloads Komponentenausfälle überstehen können.
- Ausfallmanagement erfordert Datensicherung, die Anwendung bewährter Methoden, um Ihre Workload gegen Komponentenausfälle abzusichern, und die Planung einer Notfallwiederherstellung.

Workload-Architektur

- Ihre Workload-Architektur umfasst die Art und Weise, wie Sie Services rund um geschäftliche Bereiche entwerfen, SOA und das Design verteilter Systeme anwenden, um Fehler zu vermeiden, und Funktionen wie Drosselung, Wiederholungen, Warteschlangenmanagement, Zeitüberschreitungen und Notfallfunktionen integrieren.
- Verlassen Sie sich auf bewährte AWS-Lösungen, die Amazon Builders' Library und Serverless-Muster, um sich an bewährten Methoden zu orientieren und Implementierungen anzugehen.
- Nutzen Sie kontinuierliche Verbesserungen, um Ihr System in verteilte Services aufzuteilen und so schneller zu skalieren und Innovationen voranzutreiben. Nutzen Sie Leitfäden für AWS-Microservices und verwaltete Serviceoptionen, um Ihre Möglichkeiten zur Umsetzung von Veränderungen und Innovationen zu vereinfachen und zu beschleunigen.

Kontinuierliches Testen kritischer Infrastrukturen

- Das Testen der Zuverlässigkeit bedeutet auf der Funktions-, Leistungs- und Chaos-Ebene zu testen sowie die Anwendung von Vorfallsanalysen und Gameday-Verfahren, um Fachwissen zur Lösung von Problemen aufzubauen, die nicht genau verstanden werden.
- Sowohl bei Anwendungen, die vollständig in der Cloud laufen, als auch bei hybriden Anwendungen können Sie sich schnell und zuverlässig von Ausfällen erholen, wenn Sie wissen, wie sich Ihre Anwendung bei Problemen oder Komponentenausfällen verhält.

- Erstellen und dokumentieren Sie wiederholbare Experimente, um zu verstehen, wie sich Ihr System verhält, wenn Dinge nicht wie erwartet funktionieren. Diese Tests belegen die Effektivität Ihrer allgemeinen Ausfallsicherheit und bieten eine Feedback-Schleife für Ihre operativen Verfahren, bevor Sie mit realen Fehlerszenarien konfrontiert werden.

Designprinzipien

In der Cloud gibt es zahlreiche Grundsätze, die Sie dabei unterstützen können, die Zuverlässigkeit zu erhöhen. Diese sollten Sie bei der Beschreibung der bewährten Methoden beachten:

- Automatische Wiederherstellung nach einem Fehler: Durch die Überwachung wichtiger Leistungskennzahlen (KPIs) einer Workload können Sie die Automatisierung auslösen, sobald ein Schwellenwert überschritten wurde. Diese KPIs sollten als Kennzahlen für den Geschäftswert und nicht als technische Aspekte für den Betrieb des Service betrachtet werden. Dies ermöglicht eine automatische Benachrichtigung bei und Verfolgung von Fehlern sowie die Einleitung einer automatisierten Wiederherstellung, die eine Fehlerumgehung bietet oder den Fehler behebt. Bei einer ausgefeilteren Automatisierung ist es möglich, Fehler vor ihrem eigentlichen Auftreten zu antizipieren und zu beheben.
- Testen von Wiederherstellungsverfahren: In einer On-Premises-Umgebung werden häufig Tests durchgeführt, um nachzuweisen, dass die Workload in einem bestimmten Szenario funktioniert. Mit den Tests werden in der Regel keine Wiederherstellungsstrategien validiert. In der Cloud können Sie testen, in welchen Situationen die Workload Fehler produziert, und Sie können die Wiederherstellungsverfahren validieren. Mit der Automatisierung können Sie verschiedene Fehler simulieren oder Szenarien reproduzieren, die zuvor zu Fehlern geführt haben. Dieser Ansatz stellt Fehlerpfade bereit, die Sie testen und beheben können, bevor ein echtes Fehlerszenario auftritt, und reduziert so das Risiko.
- Horizontale Skalierung zur Erhöhung der aggregierten Workload-Verfügbarkeit: Ersetzen Sie eine große Ressource durch mehrere kleine Ressourcen, um die Auswirkung eines einzigen Fehlers auf das Gesamtsystem zu reduzieren. Verteilen Sie Anfragen auf mehrere kleinere Ressourcen, damit sie keine gemeinsame Fehlerquelle haben.
- Genaue Analyse der verfügbaren Kapazität: Eine häufige Fehlerursache bei On-Premises-Workloads ist die Ressourcensättigung. Ein solches Szenario liegt vor, wenn die Anforderungen an die Workload die Kapazität dieser Workload überschreiten (dies ist häufig das Ziel von Denial-of-Service-Angriffen). In der Cloud können Sie die Nachfrage und die Workload-Auslastung überwachen und das Hinzufügen oder Entfernen von Ressourcen automatisieren, um den Bedarf ohne Über- oder Unterbereitstellung stets optimal zu erfüllen. Es gibt weiterhin Grenzen, aber

einige Kontingente können gesteuert und andere verwaltet werden (siehe Verwaltung von Service Quotas und Einschränkungen).

- Änderungsmanagement per Automatisierung: Änderungen an Ihrer Infrastruktur sollten über eine Automatisierung vorgenommen werden. Zu den Änderungen, die verwaltet werden müssen, gehören Änderungen an der Automatisierung, die anschließend nachverfolgt und überprüft werden können.

Definitionen

Dieses Whitepaper behandelt die Zuverlässigkeit in der Cloud und beschreibt bewährte Methoden für die folgenden vier Bereiche:

- Grundlagen
- Workload-Architektur
- Änderungsmanagement
- Fehlerverwaltung

Um Zuverlässigkeit zu erreichen, müssen Sie mit den Grundlagen beginnen – einer Umgebung, in der Service Quotas und Netzwerktopologie der Workload entsprechen. Die Workload-Architektur des verteilten Systems muss so ausgelegt sein, dass Ausfälle verhindert und minimiert werden. Die Workload muss Änderungen in Bezug auf den Bedarf oder die Anforderungen verarbeiten und so konzipiert sein, dass sie Fehler erkennt und sie automatisch selbst behebt.

Themen

Ausfallsicherheit und die Komponenten der Zuverlässigkeit

Die Zuverlässigkeit einer Workload in der Cloud hängt von mehreren Faktoren ab. Die Ausfallsicherheit ist hierbei der wichtigste Faktor:

- Unter Ausfallsicherheit versteht man die Fähigkeit einer Workload, sich von Infrastruktur- oder Serviceunterbrechungen zu erholen, Datenverarbeitungsressourcen dynamisch zur

Erfüllung des Bedarfs anzufordern und Unterbrechungen zu minimieren, die beispielsweise aus Fehlkonfigurationen oder vorübergehenden Netzwerkproblemen entstehen.

Weitere Faktoren, die sich auf die Zuverlässigkeit von Workloads auswirken:

- Operative Exzellenz, einschließlich Automatisierung von Änderungen, Verwendung von Playbooks zur Reaktion auf Ausfälle und Überprüfungen der betrieblichen Bereitschaft (Operational Readiness Reviews, ORRs), um zu bestätigen, dass Anwendungen für den Produktionsbetrieb bereit sind.
- Sicherheit, einschließlich der Verhinderung von Daten- oder Infrastrukturschäden durch böswillige Akteure, die sich auf die Verfügbarkeit auswirken würden. Verschlüsseln Sie beispielsweise Sicherungen, um zu gewährleisten, dass die Daten geschützt sind.
- Leistungseffizienz, einschließlich der Konzeption für maximale Anfrageraten und der Latenzminimierung für Ihre Workload.
- Kostenoptimierung, die Kompromisse einschließt, z. B. ob Sie mehr für EC2-Instances ausgeben möchten, um statische Stabilität zu erzielen, oder ob Sie sich auf die automatische Skalierung verlassen möchten, wenn mehr Kapazität benötigt wird.

In diesem Whitepaper geht es vor allem um die Ausfallsicherheit.

Die anderen vier Aspekte sind ebenfalls wichtig und werden von den jeweiligen Säulen des AWS Well-Architected Framework abgedeckt. Viele der bewährten Methoden hier kommen auch diesen Aspekten der Zuverlässigkeit zugute, der Fokus liegt aber auf der Ausfallsicherheit.

Verfügbarkeit

Verfügbarkeit (bzw. Serviceverfügbarkeit) ist sowohl eine häufig verwendete Metrik zur quantitativen Messung der Ausfallsicherheit als auch ein Ziel für die Ausfallsicherheit.

- Verfügbarkeit ist der Prozentsatz der Zeit, für den eine Workload zur Verfügung steht.

Verfügbar zur Verwendung bedeutet, dass die vereinbarte Funktion bei Bedarf erfolgreich ausgeführt wird.

Dieser Prozentsatz wird über einen definierten Zeitraum berechnet, z. B. über einen Monat, ein Jahr oder über drei aufeinander folgende Jahre. Wenn man der strengsten Interpretation folgt, reduziert sich die Verfügbarkeit immer dann, wenn die Anwendung nicht normal ausgeführt wird, einschließlich geplanter und ungeplanter Unterbrechungen. Wir definieren Verfügbarkeit wie folgt:

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

- Die Verfügbarkeit ist ein Prozentsatz der Betriebszeit (z. B. 99,9 %) über einen bestimmten Zeitraum (in der Regel ein Monat oder ein Jahr).
- Die übliche Kurzform bezieht sich nur auf die Anzahl der Neunen, so stehen „fünf Neunen“ für eine Verfügbarkeit von 99,999 %.
- Einige Kunden entscheiden sich dafür, geplante Serviceausfallzeiten (z. B. geplante Wartungsarbeiten) von der Gesamtzeit in der Formel auszuschließen. Das wird jedoch nicht empfohlen, da Ihre Benutzer Ihren Service wahrscheinlich während dieser Zeit nutzen wollen werden.

Im Folgenden finden Sie eine Tabelle mit bekannten Designzielen für die Anwendungsverfügbarkeit und der möglichen Dauer von Unterbrechungen, die innerhalb eines Jahres auftreten können, ohne sich auf das Erreichen des Ziels auszuwirken. Die Tabelle enthält Beispiele für die Anwendungstypen, die auf den jeweiligen Verfügbarkeitsstufen häufig vorkommen. In diesem Dokument beziehen wir uns immer wieder auf diese Werte.

Verfügbarkeit	Maximale Unterbrechung der Verfügbarkeit (pro Jahr)	Anwendungskategorien
99 %	3 Tage und 15 Stunden	Stapelverarbeitung, Datenextrahierung, Übertragung und Laden von Aufgaben
99,9 %	8 Stunden und 45 Minuten	Interne Tools wie Wissensmanagement, Projektverfolgung
99,95%	4 Stunden und 22 Minuten	Online-Handel, Verkaufsort
99,99 %	52 Minuten	Workloads zur Videoberitstellung und für Broadcasts

Verfügbarkeit	Maximale Unterbrechung der Verfügbarkeit (pro Jahr)	Anwendungskategorien
99,999 %	5 Minuten	Geldautomaten-Transaktionen, Telekommunikations-Workloads

Messung der Verfügbarkeit anhand von Anfragen. Es kann für Ihren Service einfacher sein, anstelle der „zur Nutzung verfügbaren Zeit“ die erfolgreichen und fehlgeschlagenen Anfragen zu messen. In diesem Fall kann die folgende Berechnung verwendet werden:

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

Dies wird oft für Zeiträume von einer oder fünf Minuten gemessen. Mit dem Durchschnitt dieser Zeiträume kann dann ein monatlicher Prozentwert für die Verfügbarkeit (zeitbasierte Verfügbarkeitsmessung) berechnet werden. Wenn in einem bestimmten Zeitraum keine Anfragen eingehen, wird für diesen Zeitraum von 100 % Verfügbarkeit ausgegangen.

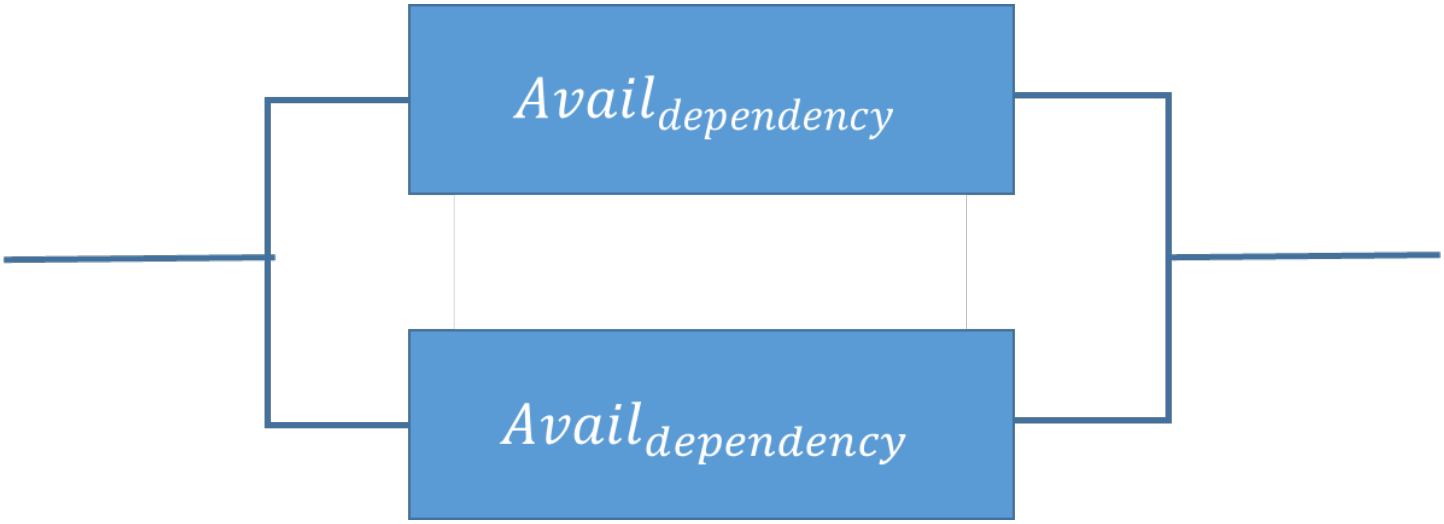
Berechnen der Verfügbarkeit mit harten Abhängigkeiten. Viele Systeme weisen harte Abhängigkeiten von anderen Systemen auf. In einem solchen Fall wirkt sich eine Unterbrechung in einem abhängigen System direkt auf eine Unterbrechung des aufrufenden Systems aus. Dies steht im Gegensatz zu einer weichen Abhängigkeit, bei der ein Fehler im abhängigen System in der Anwendung kompensiert wird. Wenn harte Abhängigkeiten auftreten, ist die Verfügbarkeit des aufrufenden Systems das Produkt der Verfügbarkeiten der abhängigen Systeme. Beispiel: Wenn Sie über ein System verfügen, das für eine Zuverlässigkeit von 99,99 % ausgelegt ist und eine harte Abhängigkeit von zwei weiteren unabhängigen Systemen aufweist, die jeweils für eine Verfügbarkeit von 99,99 % ausgelegt sind, kann die Workload theoretisch eine Verfügbarkeit von 99,97 % erreichen:

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

99,99 % × 99,99 % × 99,99 % = 99,97 %

Es ist daher wichtig, dass Sie sich bei der Berechnung Ihrer Ziele mit Ihren Abhängigkeiten und den jeweiligen Verfügbarkeitsdesignzielen vertraut machen.

Berechnen der Verfügbarkeit mit redundanten Komponenten. Wenn ein System die Verwendung unabhängiger, redundanter Komponenten beinhaltet (z. B. redundante Ressourcen in unterschiedlichen Availability Zones), wird die theoretische Verfügbarkeit mit 100 % minus dem Produkt aus den Komponentenfehlerraten berechnet. Beispiel: Wenn ein System zwei unabhängige Komponenten verwendet und jede einzelne Komponente eine Verfügbarkeit von 99,9 % aufweist, liegt die effektive Verfügbarkeit dieser Abhängigkeit bei 99,9999 %:



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

99,9999 % = 100 % − (0,1 % × 0,1 %)

Schnellberechnung: Wenn die Verfügbarkeit aller Komponenten in Ihrer Berechnung ausschließlich aus der Ziffer Neun besteht, können Sie die Anzahl der Ziffern addieren, um ein Ergebnis zu erhalten. Im obigen Beispiel ergeben zwei redundante, unabhängige Komponenten mit drei Neunen Verfügbarkeit sechs Neunen.

Berechnen der Abhängigkeitsverfügbarkeit. Für einige Abhängigkeiten stehen Informationen zur Verfügbarkeit bereit, einschließlich der Verfügbarkeitsdesignziele für viele AWS-Services. In Fällen, in denen diese Informationen jedoch nicht verfügbar sind (z. B. bei einer Komponente, bei der der Hersteller Verfügbarkeitsinformationen nicht veröffentlicht), können Sie die Verfügbarkeit über die Ermittlung der mittleren Betriebsdauer zwischen Ausfällen (MTBF) und der mittleren Reparaturzeit (MTTR)) schätzen. Eine Verfügbarkeitsschätzung kann über die folgende Formel erfolgen:

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

Wenn beispielsweise der Wert für MTBF 150 Tage ist und der Wert für MTTR mit einer Stunde angegeben ist, ergibt sich eine geschätzte Verfügbarkeit von 99,97 %.

Weitere Details finden Sie unter Availability and Beyond: Understanding and improving the resilience of distributed systems on AWS. Diese Informationen können Sie bei der Berechnung Ihrer Verfügbarkeit unterstützen.

Kosten für Verfügbarkeit. Die Entwicklung von Anwendungen für höhere Verfügbarkeiten geht in der Regel mit höheren Kosten einher. Daher ist es sinnvoll, den tatsächlichen Verfügbarkeitsbedarf zu ermitteln, bevor Sie eine Anwendung konzipieren. Höhere Verfügbarkeiten erfordern striktere Anforderungen für Tests und Validierung unter umfassenden Fehlerszenarios. Sie erfordern die Automatisierung der Wiederherstellung aus allen Fehlertypen und außerdem, dass alle Aspekte der Systemabläufe auf Basis der gleichen Standards in gleicher Weise aufgebaut und getestet werden. So müssen das Hinzufügen oder Entfernen von Kapazität, die Bereitstellung oder das Rollback von aktualisierter Software oder Konfigurationsänderungen oder die Migration von Systemdaten gemäß dem, gewünschten Verfügbarkeitsziel durchgeführt werden. Erschwerend kommt hinzu, dass neben den Kosten für die Software-Entwicklung bei einem sehr hohen Verfügbarkeitsgrad die Innovation leidet, da die Geschwindigkeit bei der Bereitstellung von Systemen sehr stark herabgesetzt werden muss. Der Leitfaden muss daher in der Anwendung der Standards und der Berücksichtigung der entsprechenden Verfügbarkeitsziele über den gesamten Lebenszyklus des Systembetriebs sehr gründlich sein.

Eine andere Möglichkeit, dass Kosten in Systemen eskalieren, die unter Designzielen für eine höhere Verfügbarkeit betrieben werden, ist die Auswahl von Abhängigkeiten. Unter diesen höheren Zielen wird sich das Angebot an Software und Services, die als Abhängigkeiten ausgewählt werden können, abhängig davon reduzieren, bei welchen dieser Services die zuvor beschriebenen hohen Investitionen zum Tragen kommen. Bei steigenden Verfügbarkeitsdesignzielen ist es üblich, dass weniger Mehrzweckservices (z. B. eine relationale Datenbank) und mehr spezielle Services angeboten werden. Der Grund dafür ist, dass spezielle Services einfacher bewertet, getestet und automatisiert werden können und ein geringeres Potenzial für überraschende Interaktionen mit eingeschlossener, jedoch nicht genutzter Funktionalität bergen.

Notfallwiederherstellungsziele

Neben Verfügbarkeitszielen sollte Ihre Strategie für Ausfallsicherheit auch Ziele für die Notfallwiederherstellung (Disaster Recovery, DR) umfassen, die auf den Strategien zum Wiederherstellen Ihrer Workload bei Auftreten eines Notfalls basieren. Die Notfallwiederherstellung konzentriert sich auf einmalige Wiederherstellungsziele als Reaktion auf Naturkatastrophen,

umfangreiche technische Fehler oder menschliche Bedrohungen wie Angriffe oder Fehler. Das unterscheidet sich von der Verfügbarkeit, bei der die mittlere Ausfallsicherheit über einen bestimmten Zeitraum bei Komponentenstörungen, Lastspitzen oder Softwarefehlern gemessen wird.

Recovery Time Objective (RTO) Wird von der Organisation festgelegt. RTO ist die maximal akzeptable Verzögerung zwischen der Unterbrechung und der Wiederherstellung des Service. Damit wird festgelegt, was als akzeptables Zeitfenster gilt, wenn der Service nicht verfügbar ist.

Recovery Point Objective (RPO) Wird von der Organisation festgelegt. RPO ist die maximal zulässige Zeitspanne seit dem letzten Wiederherstellungspunkt. Damit wird festgelegt, was als akzeptabler Datenverlust zwischen dem letzten Wiederherstellungspunkt und der Serviceunterbrechung gilt.

Die Beziehung von RPO (Recovery Point Objective), RTO (Recovery Time Objective) und dem Notfallereignis.

RTO ist ähnlich wie MTTR (mittlere Reparaturzeit), da beide die Zeit zwischen dem Start eines Ausfalls und der Wiederherstellung einer Workload messen. MTTR ist jedoch ein Mittelwert, der über mehrere Verfügbarkeitsbeeinträchtigungen für Ereignisse über einen bestimmten Zeitraum hinweg verwendet wird, während RTO ein Ziel oder Maximalwert für ein einzelnes Ereignis ist, das sich auf die Verfügbarkeit auswirkt.

Verstehen des Verfügbarkeitsbedarfs

Es ist weit verbreitet, die Verfügbarkeit einer Anwendung anfänglich als einzelnes Ziel für die Anwendung als Ganzes zu betrachten. Bei näherer Betrachtung können wir jedoch sehen, dass bestimmte Aspekte einer Anwendung oder eines Service unterschiedliche Verfügbarkeitsanforderungen aufweisen. Einige Systeme legen beispielsweise den Schwerpunkt auf die Fähigkeit, neue Daten vor dem Abrufen vorhandener Daten zu empfangen und zu speichern. Andere Systeme priorisieren Echtzeitvorgänge gegenüber Vorgängen, die sich auf die Konfiguration oder die Umgebung eines Systems auswirken können. Services können zu bestimmten Zeiten eines Tages möglicherweise mit sehr hohen Verfügbarkeitsanforderungen verknüpft sein, aber außerhalb dieser Zeiten wesentlich längere Ausfallzeiten tolerieren. Dies sind einige der Möglichkeiten, um eine Anwendung in ihre Bestandteile aufzugliedern und anschließend die Verfügbarkeitsanforderungen für diese Teile zu bewerten. Der Vorteil dieses Ansatzes besteht darin, dass der Fokus der Bemühungen (und der Kosten) gemäß den spezifischen Anforderungen auf die Verfügbarkeit gelegt werden kann, statt das gesamte System auf die strengste Anforderung hin zu konzipieren.

Empfehlung

Sie sollten die einzigartigen Aspekte Ihrer Anwendungen kritisch bewerten und die Verfügbarkeits- und Notfallwiederherstellungsdesignziele ggf. differenzieren, damit sie die Anforderungen Ihres Unternehmens widerspiegeln.

In AWS teilen wir Services in der Regel in „Datenebene“ und „Steuerebene“ auf. Die Datenebene ist zuständig für die Bereitstellung von Echtzeitservices, während die Steuerebene dazu verwendet wird, die Umgebung zu konfigurieren. So handelt es sich bei Amazon-EC2-Instances, Amazon-RDS-Datenbanken und Schreib-/Lesevorgängen in Amazon-DynamoDB-Tabellen beispielsweise ausschließlich um Datenebenenvorgänge. Im Gegensatz dazu handelt es sich beim Starten neuer EC2-Instances oder RDS-Datenbanken oder dem Hinzufügen oder Ändern von Tabellenmetadaten in DynamoDB um Vorgänge auf Steuerebene. Ein hoher Verfügbarkeitsgrad ist für all diese Funktionen wichtig. Allerdings verfolgt die Datenebene in der Regel Designziele für eine höhere Verfügbarkeit als die Steuerebene. Aus diesem Grund sollten Workloads mit hohen Verfügbarkeitsanforderungen keine Laufzeitabhängigkeiten von Abläufen der Steuerebene haben.

Viele AWS-Kunden verfolgen einen ähnlichen Ansatz, um ihre Anwendungen kritisch zu bewerten und Unterkomponenten mit abweichenden Verfügbarkeitsanforderungen zu identifizieren. Verfügbarkeitsdesignziele werden dann auf die verschiedenen Aspekte zugeschnitten und es

werden entsprechende Bemühungen unternommen, um das System weiterzuentwickeln. AWS verfügt über ausgeprägte Erfahrungen beim Engineering von Anwendungen mit verschiedenen Verfügbarkeitsdesignzielen, einschließlich Services mit einer Verfügbarkeit von 99,999 % oder mehr. AWS Solution Architects (SAs) können Sie dabei unterstützen, ein Design entsprechend Ihren Verfügbarkeitszielen zu erarbeiten. Je eher Sie AWS in Ihren Designprozess integrieren, desto besser können wir Sie bei der Erfüllung Ihrer Verfügbarkeitsziele unterstützen. Das Planen der Verfügbarkeit erfolgt nicht erst direkt vor der Einführung Ihrer Workload. Es ist ein kontinuierlicher Prozess, mit dem Sie Ihr Design mit wachsender Kenntnis, dem Kennenlernen realer Ereignisse und der Bewältigung verschiedener Fehlerarten anpassen können. Anschließend können Sie geeignete Anstrengungen unternehmen, um Ihre Implementierung zu verbessern.

Die Verfügbarkeitsanforderungen, die für eine Workload erforderlich sind, müssen den geschäftlichen Anforderungen und der Kritikalität entsprechen. Legen Sie zunächst unternehmenskritische Rahmenbedingungen mit definieren RTO-, RPO- und Verfügbarkeitswerten fest, um anschließend die einzelnen Workloads zu bewerten. Ein solcher Ansatz erfordert, dass die Mitarbeiter, die an der Implementierung der Workload beteiligt sind, über das Framework und die Auswirkungen ihrer Workload auf die Geschäftsanforderungen informiert sind.

Grundlagen

Grundlegende Anforderungen sind diejenigen, deren Umfang über eine einzelne Workload oder ein einzelnes Projekt hinausgeht. Vor dem Aufbau der Architektur eines System sollten grundlegende Anforderungen, die sich auf die Zuverlässigkeit auswirken, implementiert werden. So müssen Sie beispielsweise Ihre Rechenzentren mit einer ausreichenden Netzwerkbandbreite versorgen.

In einer On-Premises-Umgebung können diese Anforderungen aufgrund von Abhängigkeiten lange Durchlaufzeiten zur Folge haben, daher sollten sie schon in der ersten Planungsphase berücksichtigt werden. Bei AWS sind die meisten dieser grundlegenden Anforderungen bereits berücksichtigt oder können nach Bedarf adressiert werden. Die Cloud ist vom Konzept her nahezu unbegrenzt. Daher liegt es in der Verantwortung von AWS, die Anforderungen an ausreichende Netzwerk- und Datenverarbeitungskapazität zu erfüllen, sodass Sie die Größe und Zuweisungen von Ressourcen bedarfsgerecht ändern können.

In den folgenden Abschnitten werden bewährte Methoden erläutert, die sich aus Gründen der Zuverlässigkeit auf diese Überlegungen konzentrieren.

Themen

Verwaltung von Servicekontingenten und Einschränkungen

Für cloudbasierte Workload-Architekturen gibt es Servicekontingente (die auch als Servicebeschränkungen bezeichnet werden). Mit diesen Kontingenten soll verhindert werden, dass versehentlich mehr Ressourcen bereitgestellt werden, als Sie benötigen, und es sollen die Anforderungsraten bei API-Vorgängen begrenzt werden, um Services vor Missbrauch zu schützen. Es gibt auch Einschränkungen für Ressourcen, z. B. im Bezug auf die Rate, mit der Bits über ein Glasfaserkabel übertragen werden können, oder die Menge an Speicherplatz auf einer physischen Festplatte.

Bei der Nutzung von AWS-Marketplace-Anwendungen ist es wichtig, die Einschränkungen dieser Anwendungen zu kennen. Auch bei Webservices oder Software-as-a-Service-Angeboten von Drittanbietern ist es wichtig, mit den Limits vertraut zu sein.

Best Practices

REL01-BP01 Kenntnis von Service Quotas und Einschränkungen

Sie wissen über die Standardkontingente Bescheid und verwalten Anfragen zur Kontingenterhöhung für Ihre Workload-Architektur. Außerdem wissen Sie, welche Ressourceneinschränkungen, z. B. bezüglich Datenträgern oder Netzwerken, potenziell große Auswirkungen haben.

Gewünschtes Ergebnis: Kunden können eine Beeinträchtigung oder Unterbrechung ihrer Services in ihrer AWS-Konten verhindern, indem sie geeignete Richtlinien für die Überwachung von Schlüsselkennzahlen, Infrastrukturüberprüfungen und Automatisierungsschritte zur Behebung von Problemen einführen, um sicherzustellen, dass Service Quotas und Einschränkungen, die eine Beeinträchtigung oder Unterbrechung der Dienste verursachen könnten, nicht erreicht werden.

Typische Anti-Muster:

- Bereitstellung eines Workloads ohne Kenntnis der harten oder weichen Quoten und ihrer Grenzen für die verwendeten Services.
- Bereitstellung eines Ersatz-Workloads, ohne die erforderlichen Quoten zu analysieren und neu zu konfigurieren oder den Support im Voraus zu kontaktieren.
- Annehmen, dass Cloud-Services keine Grenzen haben und die Service ohne Berücksichtigung von Tarifen, Grenzen, Zählungen und Mengen genutzt werden können.
- Annehmen, dass die Quoten automatisch erhöht werden.
- Keine Kenntnis des Prozesses und der Zeitleiste von Quotenanforderungen.
- Annehmen, dass das Standardkontingent für Cloud-Services für jeden Service im regionalen Vergleich identisch ist.
- Annehmen, dass die Servicebeschränkungen überschritten werden können und die Systeme automatisch skalieren oder das Limit über die Beschränkungen der Ressource hinaus erhöhen.

- Die Anwendung nicht bei Spitzenbelastungen testen, um die Auslastung der Ressourcen zu strapazieren.
- Bereitstellung der Ressource ohne Analyse der erforderlichen Ressourcengröße.
- Überbereitstellung von Kapazitäten durch Auswahl von Ressourcentypen, die weit über den tatsächlichen Bedarf oder die erwarteten Spitzen hinausgehen.
- Keine Bewertung des Kapazitätsbedarfs für neue Datenverkehrsniveaus im Vorfeld eines neuen Kundenereignisses und keine Einführung einer neuen Technologie.

Vorteile der Nutzung dieser bewährten Methode: Durch die Überwachung und automatisierte Verwaltung von Service Quotas und Ressourcenbeschränkungen können Ausfälle proaktiv reduziert werden. Änderungen in den Datenverkehrsmustern für den Service eines Kunden können zu einer Unterbrechung oder Verschlechterung führen, wenn die bewährten Methoden nicht befolgt werden. Durch die Überwachung und Verwaltung dieser Werte in allen Regionen und auf allen Konten können die Anwendungen bei ungünstigen oder ungeplanten Ereignissen besser geschützt werden.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Hoch

Implementierungsleitfaden

Service Quotas ist ein AWS-Service, mit dem Sie Ihre Kontingente für über 250 AWS-Services von einem Standort aus verwalten können. Neben der Suche nach den Kontingentwerten können Sie auch Kontingenterhöhungen über die Service Quotas-Konsole oder über das AWS SDK anfordern und nachverfolgen. AWS Trusted Advisor bietet eine Service Quotas-Prüfung, die Ihre Nutzung und Ihre Kontingente für bestimmte Aspekte einiger Services anzeigt. Die Standardkontingente pro Service finden Sie ebenfalls in der AWS-Dokumentation für den jeweiligen Service (weitere Informationen finden Sie unter Amazon VPC-Kontingente).

Einige Servicelimits wie Ratenlimits für gedrosselte APIs werden innerhalb des Amazon API Gateway selbst festgelegt. Dazu wird ein Nutzungsplan konfiguriert. Andere Limits, die für ihre jeweiligen Services konfiguriert werden, sind bereitgestellte IOPS, zugewiesener Amazon RDS-Speicher und Amazon EBS-Volume-Zuweisungen. Amazon Elastic Compute Cloud verfügt über ein eigenes Service Limits-Dashboard, mit dem Sie Ihre Limits für Instances, Amazon Elastic Block Store und Elastic IP-Adressen verwalten können. Wenn Sie einen Anwendungsfall haben, bei dem sich Service Quotas auf die Leistung Ihrer Anwendung auswirken und eine Anpassung an Ihre Anforderungen nicht möglich ist, wenden Sie sich an den Support, um zu ermitteln, ob es Lösungen gibt.

Service Quotas können spezifisch für eine Region oder auch global sein. Ein AWS-Service, der sein Kontingent erreicht hat, verhält sich bei normaler Nutzung nicht wie erwartet und es kann zu

Unterbrechungen oder Beeinträchtigungen des Services kommen. Beispielsweise begrenzen Service Quotas die Anzahl der DL Amazon EC2, die in einer Region genutzt werden können, und dieses Limit kann während eines Ereignisses zur Skalierung des Datenverkehrs durch Auto Scaling-Gruppen (ASG) erreicht werden.

Service Quotas für die einzelnen Konten sollten regelmäßig auf ihre Nutzung hin überprüft werden, um festzustellen, welche Servicelimits für das jeweilige Konto angemessen sind. Diese Service Quotas dienen als betrieblicher Integritätsschutz, um zu verhindern, dass versehentlich mehr Ressourcen bereitgestellt werden, als Sie benötigen. Sie begrenzen auch die Anfrageraten bei API-Operationen, um Services vor Missbrauch zu schützen.

Serviceeinschränkungen und Service Quotas unterscheiden sich voneinander. Serviceeinschränkungen stellen die Limits einer bestimmten Ressource dar, wie sie durch diesen Ressourcentyp definiert sind. Dabei kann es sich um die Speicherkapazität (z. B. hat gp2 eine Größenbegrenzung von 1 GB bis 16 TB) oder den Festplattendurchsatz handeln. Es ist von entscheidender Bedeutung, dass die Beschränkung eines Ressourcentyps konstruiert und ständig auf eine Nutzung geprüft wird, durch die das Limit erreicht werden könnte. Wenn eine Beschränkung unerwartet erreicht wird, können die Anwendungen oder Services des Kontos beeinträchtigt oder unterbrochen werden.

Wenn es einen Anwendungsfall gibt, bei dem sich Service Quotas auf die Leistung Ihrer Anwendung auswirken und eine Anpassung an die Anforderungen nicht möglich ist, wenden Sie sich an den Support, um zu ermitteln, ob es Lösungen gibt. Weitere Einzelheiten zur Anpassung fester Kontingente finden Sie unter REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur.

Es gibt eine Reihe von AWS-Services und -Tools, die Sie bei der Überwachung und Verwaltung von Service Quotas unterstützen. Der Service und die Tools sollten genutzt werden, um automatische oder manuelle Überprüfungen der Kontingente zu ermöglichen.

- AWS Trusted Advisor bietet eine Service Quotas-Prüfung, die Ihre Nutzung und Ihre Kontingente für einige Aspekte einiger Services anzeigt. Es kann dabei helfen, Services zu identifizieren, die ihr Kontingent fast erreicht haben.
- AWS-Managementkonsole bietet Methoden, um Service-Quota-Werte für Services anzuzeigen, zu verwalten, neue Kontingente anzufordern, den Status von Kontingentanforderungen zu überwachen und den Verlauf von Kontingenten anzuzeigen.
- AWS CLI und CDKs bieten programmatische Methoden zur automatischen Verwaltung und Überwachung von Service Quotas und deren Nutzung.

Implementierungsschritte

Für Service Quotas:

- Überprüfen Sie AWS Service Quotas.
- Bestimmen Sie die verwendeten Services (wie IAM Access Analyzer), damit Sie Ihre bestehenden Service Quotas kennen. Es gibt etwa 250 AWS-Services, für die Service Quotas gelten. Bestimmen Sie dann den spezifischen Service-Quota-Namen, der für jedes Konto und jede Region verwendet werden kann. Pro Region gibt es etwa 3 000 Service-Quota-Namen.
- Ergänzen Sie diese Kontingentanalyse um AWS Config, um alle AWS-Ressourcen zu finden, die in Ihren AWS-Konten verwendet werden.
- Bestimmen Sie anhand von AWS CloudFormation-Daten Ihre verwendeten AWS-Ressourcen. Sehen Sie sich die Ressourcen an, die in der AWS-Managementkonsole oder über den Befehl `list-stack-resources`AWS CLIBefehl erstellt wurden. Sie können zudem Ressourcen anzeigen, die für die Bereitstellung in der Vorlage selbst konfiguriert sind.
- Ermitteln Sie alle für die Workload erforderlichen Services durch Untersuchung des Bereitstellungscodes.
- Ermitteln Sie die geltenden Service Quotas. Nutzen Sie die programmgesteuert über Trusted Advisor und Service Quotas zugänglichen Informationen. .
- Richten Sie eine automatisierte Überwachungsmethode ein (siehe REL01-BP02 Verwalten von Service Quotas für mehrere Konten und Regionen und REL01-BP04 Kontingente überwachen und verwalten), um zu warnen und zu informieren, wenn die Service Quotas fast erschöpft sind oder ihr Limit erreicht haben.
- Richten Sie eine automatische, programmatische Methode ein, um zu überprüfen, ob ein Service Quota in einer Region, aber nicht in anderen Regionen desselben Kontos geändert wurde (siehe REL01-BP02 Verwalten von Service Quotas für mehrere Konten und Regionen und REL01-BP04 Kontingente überwachen und verwalten).
- Automatisieren Sie das Scannen von Anwendungsprotokollen und Metriken, um festzustellen, ob Fehler beim Kontingent oder bei Serviceeinschränkungen vorliegen. Falls Fehler vorhanden sind, senden Sie Warnmeldungen an das Überwachungssystem.
- Führen Sie technische Verfahren zur Berechnung der erforderlichen Kontingentänderung ein (siehe REL01-BP05 Automatisieren der Kontingentverwaltung), wenn festgestellt wird, dass für bestimmte Services größere Kontingente erforderlich sind.

- Erstellen Sie einen Bereitstellungs- und Genehmigungs-Workflow, um Änderungen am Service Quota anzufordern. Dies sollte einen Ausnahme-Workflow für den Fall umfassen, dass ein Antrag abgelehnt oder nur teilweise genehmigt wird.
- Erstellen Sie eine technische Methode zur Überprüfung von Service Quotas vor der Bereitstellung und Nutzung neuer AWS-Services, und zwar vor dem Rollout in Produktionsumgebungen oder Umgebungen mit Last (z. B. Lasttestkonto).

Bei Serviceeinschränkungen:

- Führen Sie Überwachungs- und Messmethoden ein, um auf Ressourcen aufmerksam zu machen, die ihre Ressourceneinschränkungen fast erreicht haben. Nutzen Sie CloudWatch gegebenenfalls für Metriken oder Protokollüberwachung.
- Legen Sie Warnschwellenwerte für jede Ressource fest, die eine für die Anwendung oder das System bedeutsame Einschränkung hat.
- Erstellen Sie Verfahren für die Verwaltung von Workflows und Infrastrukturen, um den Ressourcentyp zu ändern, wenn die Nutzungseinschränkung fast erreicht ist. Dieser Workflow sollte Lasttests beinhalten, um zu überprüfen, ob der neue Typ der richtige Ressourcentyp mit den neuen Einschränkungen ist.
- Migrieren Sie die identifizierte Ressource unter Verwendung bestehender Verfahren und Prozesse auf den empfohlenen neuen Ressourcentyp.

Ressourcen

Zugehörige bewährte Methoden:

- REL01-BP02 Verwalten von Service Quotas für mehrere Konten und Regionen
- REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur
- REL01-BP04 Kontingente überwachen und verwalten
- REL01-BP05 Automatisieren der Kontingentverwaltung
- REL01-BP06 Sicherstellen eines ausreichenden Spielraums zwischen den aktuellen Kontingenten und der maximalen Nutzung, damit ein Failover möglich ist
- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten
- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (früher als Service Limits bezeichnet)
- Bewährte AWS Trusted Advisor-Prüfungsmethoden (siehe Abschnitt „Servicelimits“)
- AWS Limit Monitor in AWS Answers
- Amazon EC2 Service Limits
- Was ist Service Quotas?
- How to Request Quota Increase (So beantragen Sie eine Kontingenterhöhung)
- Service endpoints and quotas (Service-Endpunkte und -Quoten)
- Service Quotas — Benutzerhandbuch
- Quota Monitor für AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS für Daten
- What is Continuous Integration?
- What is Continuous Delivery?
- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Zugehörige Videos:

- AWS Live re:Inforce 2019 – Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

Zugehörige Tools:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP02 Verwalten von Service Quotas für mehrere Konten und Regionen

Wenn Sie mehrere Konten oder Regionen verwenden, fordern Sie die entsprechenden Kontingente in allen Umgebungen an, in denen die Produktions-Workloads ausgeführt werden.

Gewünschtes Ergebnis: Services und Anwendungen sollten bei Konfigurationen, die sich über Konten oder Regionen erstrecken oder die über ein Ausfallsicherheitsdesign mit Zonen-, Regions- oder Konto-Failover verfügen, nicht von der Erschöpfung des Service Quota betroffen sein.

Typische Anti-Muster:

- Es wird zugelassen, dass die Ressourcennutzung in einer Isolationsregion zunimmt, ohne dass es einen Mechanismus zur Aufrechterhaltung der Kapazität in den anderen Zonen gibt.
- Alle Kontingente werden manuell und in jeder Isolationsregion einzeln festgelegt.
- Nichtberücksichtigung der Auswirkungen von Ausfallsicherheitsarchitekturen (wie aktiv oder passiv) auf den künftigen Kontingentbedarf bei einer Verschlechterung in der nicht primären Region.
- Keine regelmäßige Bewertung der Kontingente und Durchführung der erforderlichen Änderungen in jeder Region und jedem Konto, in dem die Workload ausgeführt wird.
- Keine Nutzung von Vorlagen für Kontingentanforderungen, um Erhöhungen für mehrere Regionen und Konten zu beantragen.

- Keine Aktualisierung von Service Quotas, weil man fälschlicherweise davon ausgeht, dass eine Erhöhung der Kontingente Kosten nach sich zieht, wie z. B. Anforderungen von Rechenkapazitäten.

Vorteile der Einführung dieser bewährten Methode: Überprüfen, ob Sie Ihre aktuelle Last in sekundären Regionen oder Konten bewältigen können, falls regionale Services nicht mehr verfügbar sind. Dies kann dazu beitragen, die Anzahl von Fehlern oder Verschlechterungen zu verringern, die beim Verlust von Regionen auftreten.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Service Quotas werden pro Konto aufgezeichnet. Sofern nicht anders angegeben, gilt jedes Kontingent für eine bestimmte AWS-Region. Zusätzlich zu den Produktionsumgebungen verwalten Sie auch Kontingente in allen anwendbaren Nicht-Produktionsumgebungen, damit Tests und Entwicklung nicht behindert werden. Die Aufrechterhaltung eines hohen Maßes an Ausfallsicherheit setzt voraus, dass die Service Quotas ständig überprüft werden (entweder automatisch oder manuell).

Da durch die Implementierung von Designs mit den Ansätzen Aktiv/Aktiv, Aktiv/Passiv – Hot, Aktiv/Passiv – Cold und Aktiv/Passiv – Pilot Light immer mehr Workloads auf die Regionen verteilt werden, ist es wichtig, alle Kontingente für Regionen und Konten zu kennen. Frühere Datenverkehrsmuster sind nicht immer ein guter Indikator dafür, ob das Service Quota korrekt eingestellt ist.

Ebenso wichtig ist, dass das Namenslimit für das Service Quota nicht immer für alle Regionen gleich ist. In einer Region kann der Wert fünf sein, in einer anderen zehn. Die Verwaltung dieser Kontingente muss sich auf dieselben Services, Konten und Regionen erstrecken, um eine gleichmäßige Ausfallsicherheit unter Last zu gewährleisten.

Stimmen Sie alle Unterschiede zwischen den Service Quotas in den verschiedenen Regionen (aktive oder passive Region) ab und schaffen Sie Prozesse, um diese Unterschiede kontinuierlich abzugleichen. Die Testpläne für passive Regions-Failover sind selten auf die aktive Spitzenkapazität skaliert, was bedeutet, dass es im Ernstfall oder bei Tabletop-Übungen nicht gelingen kann, Unterschiede bei den Service Quotas zwischen den Regionen festzustellen und die korrekten Limits einzuhalten.

Service-Quota-Abweichung, d. h. der Umstand, dass die Service-Quota-Limits für ein bestimmtes benanntes Kontingent in einer Region und nicht in allen Regionen geändert werden, müssen

unbedingt verfolgt und bewertet werden. Es sollte erwogen werden, die Kontingente in Regionen mit Datenverkehr oder potenziellem Datenverkehr zu ändern.

- Wählen Sie relevante Konten und Regionen anhand von Serviceanforderungen, regulatorischen Anforderungen sowie Anforderungen für die Latenz und die Notfallwiederherstellung aus.
- Ermitteln Sie Service Quotas für alle relevanten Konten, Regionen und Availability Zones. Die Limits gelten für ein Konto und eine Region. Diese Werte sollten auf Unterschiede hin verglichen werden.

Implementierungsschritte

- Überprüfen Sie die Service Quotas-Werte, die über eine Risikostufe der Nutzung hinausgehen. AWS Trusted Advisor bietet Warnungen bei Überschreitung der Schwellenwerte von 80 % und 90 %.
- Überprüfen Sie die Werte für Service Quotas in allen passiven Regionen (in einem Aktiv/Passiv-Design). Stellen Sie sicher, dass die Last in den sekundären Regionen bei einem Ausfall in der primären Region erfolgreich ausgeführt werden kann.
- Automatisieren Sie die Bewertung, ob es zu einer Verschiebung der Service Quotas zwischen den Regionen desselben Kontos gekommen ist, und handeln Sie entsprechend, um die Limits zu ändern.
- Wenn die Organisationseinheiten (OU) des Kunden in der unterstützten Weise strukturiert sind, sollten die Vorlagen für Service Quotas aktualisiert werden, um Änderungen an Kontingenten widerzuspiegeln, die auf mehrere Regionen und Konten angewendet werden sollen.
  - Erstellen Sie eine Vorlage und weisen Sie der Kontingentänderung Regionen zu.
  - Überprüfen Sie alle bestehenden Vorlagen für Service Quotas auf erforderliche Änderungen (Region, Limits und Konten).

Ressourcen

Zugehörige bewährte Methoden:

- REL01-BP01 Kenntnis von Service Quotas und Einschränkungen
- REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur
- REL01-BP04 Kontingente überwachen und verwalten

- REL01-BP05 Automatisieren der Kontingentverwaltung
- REL01-BP06 Sicherstellen eines ausreichenden Spielraums zwischen den aktuellen Kontingenten und der maximalen Nutzung, damit ein Failover möglich ist
- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten
- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler
- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (früher als Service Limits bezeichnet)
- AWS Trusted Advisor Bewährte -Prüfungsmethoden (siehe Abschnitt „Servicelimits)
- AWS Limit Monitor in AWS Answers
- Amazon EC2 Service Limits
- Was ist Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas-Benutzerhandbuch
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS für Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Zugehörige Videos:

- AWS Live re:Inforce 2019 – Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

Zugehörige Services:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur

Achten Sie auf nicht änderbare Service Quotas, Servicebeschränkungen und physische Ressourcen-Limits. Entwerfen Sie Architekturen für Anwendungen und Services, um zu verhindern, dass sich diese Limits auf die Zuverlässigkeit auswirken.

Beispiele hierfür sind die Netzwerkbandbreite, die Datengröße beim Aufrufen von Serverless-Funktionen, die Drosselung der Burst-Rate eines API-Gateways und die gleichzeitig mit einer Datenbank verbundenen Benutzer.

Gewünschtes Ergebnis: Die Anwendung oder der Service erbringt unter normalen Bedingungen und bei hohem Datenverkehr die erwartete Leistung. Sie wurden so konzipiert, dass sie innerhalb der für diese Ressource festgelegten Beschränkungen oder Service-Kontingente arbeiten.

Typische Anti-Muster:

- Auswahl eines Designs, das eine Ressource eines Service verwendet, ohne zu wissen, dass es Design-Einschränkungen gibt, die dazu führen, dass dieses Design beim Skalieren versagt.
- Sie führen ein Benchmarking durch, das unrealistisch ist und mit dem während der Tests die festen Kontingente für den Service erreicht werden. Sie führen beispielsweise Tests mit einem Burst-Limit durch, diese aber für einen längeren Zeitraum.
- Sie wählen ein Design aus, das nicht skaliert oder geändert werden kann, wenn feste Service-Kontingente überschritten werden müssen. Ein Beispiel wäre ein SQS-Payload von 256 KB.
- Die Überwachungsfunktion wurde nicht zur Überwachung und Benachrichtigung von/für Schwellenwerte/n für Service-Kontingente entwickelt und implementiert, die bei hohem Datenverkehr gefährdet sein könnten.

Vorteile der Nutzung dieser bewährten Methode: Es wird sichergestellt, dass die Anwendung unter allen prognostizierten Last-Levels der Services ohne Unterbrechung oder Beeinträchtigung läuft.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

Im Gegensatz zu Soft-Kontingenten für Services oder Ressourcen, die durch Einheiten mit höherer Kapazität ersetzt werden können, können feste Kontingente für AWS-Services nicht geändert werden. Das bedeutet, dass alle AWS-Services dieser Art auf potenzielle harte Kapazitätsgrenzen geprüft werden müssen, wenn sie in einer Anwendung zum Einsatz kommen.

Feste Beschränkungen werden in der Service Quotas-Konsole angezeigt. Wenn in den Spalten `ADJUSTABLE = No` angezeigt wird, gibt es eine feste Beschränkung für den Service. Auch auf einigen Konfigurationsseiten für Ressourcen werden feste Beschränkungen angezeigt. Für Lambda gibt es zum Beispiel bestimmte feste Beschränkungen, die nicht angepasst werden können.

Wenn Sie beispielsweise eine Python-Anwendung entwerfen, die in einer Lambda-Funktion ausgeführt werden soll, sollte die Anwendung daraufhin geprüft werden, ob die Möglichkeit besteht, dass Lambda länger als 15 Minuten läuft. Wenn die Codeausführung länger als dieses Service-Kontingent dauert, müssen alternative Technologien oder Designs in Betracht gezogen werden. Wird diese Beschränkung nach der Bereitstellung in der Produktion erreicht, wird die Anwendung beeinträchtigt und gestört, bis sie wiederhergestellt werden kann. Im Gegensatz zu Soft-Kontingenten gibt es keine Möglichkeit, diese Beschränkungen zu ändern – selbst wenn ein Ereignis des Schweregrads 1 eintritt.

Sobald die Anwendung in einer Testumgebung bereitgestellt wurde, sollten Strategien eingesetzt werden, um herauszufinden, ob feste Beschränkungen erreicht werden könnten. Stresstests, Lasttests und Chaostests sollten Teil des Einführungstestplans sein.

Implementierungsschritte

- Sehen Sie sich die vollständige Liste der AWS-Services an. Diese können Sie in der Entwurfsphase der Anwendung verwenden.
- Sehen Sie sich die Soft-Kontingentbeschränkungen und Hard-Kontingentbeschränkungen der Services an. Nicht alle Beschränkungen werden in der Service Quotas-Konsole angezeigt. Einige Services zeigen die Beschränkungen an anderen Stellen an.
- Prüfen Sie bei der Entwicklung Ihrer Anwendung die geschäftlichen und technologischen Faktoren Ihres Workloads, wie z. B. Geschäftsergebnisse, Anwendungsfälle, abhängige Systeme, Verfügbarkeitsziele und Objekte für die Notfallwiederherstellung. Lassen Sie sich von Ihren geschäftlichen und technologischen Faktoren leiten, um das richtige verteilte System für Ihren Workload zu finden.
- Analysieren Sie die Last des Services über Regionen und Konten hinweg. Viele feste Beschränkungen für Services basieren auf Regionen. Einige Beschränkungen sind jedoch kontobasiert.
- Analysieren Sie die Architekturen zur Ausfallsicherheit der Ressourcen bei einem zonenbezogenen Fehler und einem Fehler in einer Region. Bei der Entwicklung von Multi-Regionen-Designs mit Aktiv/Aktiv-, Aktiv/Passiv-Hot-, Aktiv/Passiv-Cold- und Aktiv/Passiv-Pilot-Light-Ansätzen werden diese Fehlerfälle eine höhere Auslastung verursachen. Dies schafft einen potenziellen Anwendungsfall für feste Beschränkungen.

Ressourcen

Zugehörige bewährte Methoden:

- REL01-BP01 Kenntnis von Service Quotas und Einschränkungen
- REL01-BP02 Verwalten von Service Quotas für mehrere Konten und Regionen
- REL01-BP04 Kontingente überwachen und verwalten
- REL01-BP05 Automatisieren der Kontingentverwaltung
- REL01-BP06 Sicherstellen eines ausreichenden Spielraums zwischen den aktuellen Kontingenten und der maximalen Nutzung, damit ein Failover möglich ist
- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten

- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler
- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (früher als Service Limits bezeichnet)
- AWS Trusted Advisor Bewährte -Prüfungsmethoden (siehe Abschnitt „Servicelimits)
- AWS Limit Monitor in AWS Answers
- Amazon EC2 Service Limits
- Was ist Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas-Benutzerhandbuch
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS für Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Aktionen, Ressourcen und Bedingungsschlüssel für Service Quotas

Zugehörige Videos:

- AWS Live re:Inforce 2019 – Service Quotas

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

Zugehörige Tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP04 Kontingente überwachen und verwalten

Überprüfen Sie die potenzielle Nutzung und erhöhen Sie Ihre Kontingente entsprechend, um einen geplanten Nutzungsanstieg zu ermöglichen.

Gewünschtes Ergebnis: Es wurden aktive und automatisierte Verwaltungs- und Überwachungssysteme bereitgestellt. Diese operativen Lösungen reagieren, wenn die Schwellenwerte für die Kontingentnutzung fast erreicht werden. Sie lösen die Situation durch die proaktiven Änderungen des Kontingents.

Typische Anti-Muster:

- Keine Konfigurationsüberwachung zur Prüfung von Schwellenwerten für das Service-Kontingent.
- Keine Konfigurationsüberwachung für feste Beschränkungen, auch wenn diese Werte nicht geändert werden können.
- Sie gehen davon aus, dass eine Änderung des Soft-Kontingents direkt stattfindet oder nur wenig Zeit erfordert.

- Es werden Warnungen für den Fall konfiguriert, dass Servicekontingente erreicht werden, aber es gibt keinen Prozess für die Reaktion auf eine entsprechende Warnung.
- Nur Alarme für Dienste konfigurieren, die von AWS Service Quotas unterstützt werden, und keine Überwachung anderer AWS Dienste.
- Keine Berücksichtigung der Verwaltung von Kontingenten für die Ausfallsicherheit mehrerer Regionen, wie z. B. Aktiv/Aktiv-, Aktiv/Passiv-Hot-, Aktiv/Passiv-Cold- und Aktiv/Passiv-Pilot-Light-Ansätze.
- Keine Bewertung der Kontingentunterschiede zwischen den Regionen.
- Keine Bewertung des Bedarfs in jeder Region für eine bestimmte Kontingentserhöhung.
- Keine Nutzung von Vorlagen für die Verwaltung von Kontingenten für mehrere Regionen.

Vorteile der Einführung dieser bewährten Methode: Durch die automatische Verfolgung der AWS Service Quotas und die Überwachung Ihrer Nutzung anhand dieser Kontingente können Sie erkennen, wann Sie sich einer Kontingentbegrenzung nähern. Sie können diese Überwachungsdaten außerdem nutzen, um Verschlechterungen aufgrund einer Kontingentausschöpfung zu begrenzen.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

Bei unterstützten Services können Sie Ihre Kontingente überwachen, indem Sie verschiedene Services zur Bewertung und anschließenden Versendung von Warnungen konfigurieren. Auf diese Weise können Sie die Nutzung überwachen und werden auf sich nähernde Kontingentgrenzen aufmerksam gemacht. Diese Alarme können von Lambda-Funktionen AWS Config CloudWatch, Amazon oder von aufgerufen werden. AWS Trusted Advisor Sie können auch Metrikfilter für CloudWatch Protokolle verwenden, um Muster in Protokollen zu suchen und zu extrahieren, um festzustellen, ob sich die Nutzung den Kontingentschwellenwerten nähert.

Implementierungsschritte

Für die Überwachung:

- Erfassen Sie den aktuellen Ressourcenverbrauch (z. B. Buckets oder Instances). Verwenden Sie API Servicebetriebe wie Amazon EC2 `DescribeInstances`API, um den aktuellen Ressourcenverbrauch zu ermitteln.
- Erfassen Sie Ihre aktuellen Kontingente, die für die Services wesentlich und anwendbar sind. Nutzen Sie dazu:

  - AWS Service Quotas
  - AWS Trusted Advisor
  - AWS Dokumentation
  - AWS dienstspezifische Seiten
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- Verwenden Sie AWS Service Quotas, einen AWS Service, mit dem Sie Ihre Kontingente für über 250 AWS Dienste von einem Standort aus verwalten können.
- Verwenden Sie Trusted Advisor Servicelimits, um Ihre aktuellen Servicelimits bei verschiedenen Schwellenwerten zu überwachen.
- Prüfen Sie anhand der Historie der Servicekontingenten (Konsole oder AWS CLI), ob regionale Erhöhungen vorliegen.
- Vergleichen Sie die Änderungen der Service-Kontingente in jeder Region und jedem Konto, um bei Bedarf auszugleichen.

Für die Verwaltung:

- Automatisiert: Richten Sie eine AWS Config benutzerdefinierte Regel ein, um Servicekontingente in verschiedenen Regionen zu scannen und Unterschiede zu vergleichen.
- Automatisiert: Richten Sie eine geplante Lambda-Funktion ein, um Service-Kontingente in den Regionen zu scannen und Abweichungen zu ermitteln.
- Manuell: Scannen Sie die Service-Kontingente über AWS CLI API, oder die AWS Konsole, um Servicekontingente in verschiedenen Regionen zu scannen und auf Unterschiede zu vergleichen. Erstellen Sie einen Bericht zu den Abweichungen.
- Wenn Abweichungen in den Kontingenten zwischen den Regionen festgestellt werden, fordern Sie bei Bedarf eine Kontingentänderung an.
- Überprüfen Sie das Ergebnis aller Anforderungen.

Ressourcen

Zugehörige bewährte Methoden:

- REL01-BP01 Kenntnis von Service Quotas und Einschränkungen
- REL01-BP02 Verwalten von Service Quotas für mehrere Konten und Regionen

- REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur
- REL01-BP05 Automatisieren der Kontingentverwaltung
- REL01-BP06 Sicherstellen eines ausreichenden Spielraums zwischen den aktuellen Kontingenten und der maximalen Nutzung, damit ein Failover möglich ist
- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten
- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler
- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Die Zuverlässigkeitssäule von Well-Architected Framework: Verfügbarkeit
- AWS Service Quotas (früher als Service Limits bezeichnet)
- AWS Trusted Advisor Prüfungen bewährter Verfahren (siehe Abschnitt Service Limits)
- AWS Beschränken Sie den Monitor auf AWS Antworten
- EC2Amazon-Servicebeschränkungen
- Was ist Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas-Benutzerhandbuch
- Kontingentmonitor für AWS
- AWS Grenzen der Fehlerisolierung
- Availability with redundancy
- AWS für Daten
- What is Continuous Integration?
- What is Continuous Delivery?
- APNPartner: Partner, die beim Konfigurationsmanagement helfen können
- Verwaltung des Kontolebenszyklus in account-per-tenant SaaS-Umgebungen auf AWS
- Verwaltung und Überwachung der API Drosselung Ihrer Workloads

- Sehen Sie sich AWS Trusted Advisor Empfehlungen in großem Umfang an mit AWS Organizations
- Automatisierung von Service-Limit-Erhöhungen und Unternehmenssupport mit AWS Control Tower
- Aktionen, Ressourcen und Bedingungsschlüssel für Service Quotas

Zugehörige Videos:

- AWS Live re:inForce 2019 — Service Quotas
- Kontingente für AWS Dienste mithilfe von Service Quotas anzeigen und verwalten
- AWS IAMDemo der Kontingente
- AWS re:Invent 2018: Kreisläufe schließen und neue Denkansätze eröffnen: Wie man die Kontrolle über große und kleine Systeme übernimmt

Zugehörige Tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- DevOpsAmazon-Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP05 Automatisieren der Kontingentverwaltung

Service Quotas, in AWS-Services auch als Einschränkungen bezeichnet, sind die Höchstwerte für die Ressourcen in Ihrem AWS-Konto. Jeder AWS-Service definiert eine Reihe von Kontingenten und ihre Standardwerte. Um Ihrem Workload Zugriff auf alle benötigten Ressourcen zu gewähren, müssen Sie möglicherweise Ihre Service-Quota-Werte erhöhen.

Ein Anstieg des Verbrauchs von AWS-Ressourcen durch Workloads kann die Workload-Stabilität gefährden und sich negativ auf die Benutzererfahrung auswirken, wenn Kontingente überschritten

werden. Implementieren Sie Tools, um benachrichtigt zu werden, wenn Ihr Workload die Grenzwerte erreicht, und ziehen Sie die automatische Erstellung von Anfragen zur Kontingenterhöhung in Betracht.

Gewünschtes Ergebnis: Die Kontingente werden für die Workloads, die in den einzelnen AWS-Konto-Regionen ausgeführt werden, angemessen konfiguriert.

Typische Anti-Muster:

- Sie betrachten die Kontingente nicht und passen sie nicht entsprechend an die Workload-Anforderungen an.
- Sie verfolgen Kontingente und Nutzung mithilfe von Methoden nach, die möglicherweise veraltet sind, z. B. mit Tabellenkalkulationen.
- Sie aktualisieren die Servicegrenzen nur in regelmäßigen Abständen.
- Ihre Organisation besitzt keine operativen Prozesse für die Überprüfung vorhandener Kontingente und die Anforderung von Service-Quota-Erhöhungen, wenn notwendig.

Vorteile der Nutzung dieser bewährten Methode:

- Verbesserte Workload-Resilienz: Sie verhindern Fehler, die durch Überschreitung von AWS-Ressourcenkontingenten verursacht werden.
- Vereinfachte Notfallwiederherstellung: Sie können automatische Mechanismen zur Kontingentverwaltung, die während der DR-Einrichtung in der primären Region eingerichtet wurden, in einer anderen AWS-Region wiederverwenden.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

Zeigen Sie aktuelle Kontingente an und verfolgen Sie den laufenden Kontingentverbrauch mithilfe von Mechanismen wie der Konsole in AWS Service Quotas, AWS Command Line Interface (AWS CLI), und AWS SDKs nach. Sie können auch Ihre Datenbanken für die Konfigurationsverwaltung (Configuration Management Databases, CMDB) und Ihre Systeme für die Verwaltung von IT-Services (IT Service Management, ITSM) in die APIs von AWS Service Quotas integrieren.

Generieren Sie automatische Benachrichtigungen, wenn die Kontingentnutzung die definierten Schwellenwerte erreicht, und definieren Sie einen Prozess für das Einreichen von Anfragen zur Erhöhung des Kontingents, wenn Sie Benachrichtigungen erhalten. Wenn die zugrundeliegenden

Workloads kritisch für Ihr Unternehmen sind, können Sie die Anfragen zur Quotenerhöhung automatisieren. Testen Sie die Automatisierung jedoch sorgfältig, um zu vermeiden, dass sie außer Kontrolle gerät, indem beispielsweise eine wachsende Feedbackschleife entsteht.

Kleinere Kontingenterhöhungen werden häufig automatisch genehmigt. Größere Kontingentanfragen müssen möglicherweise manuell vom AWS-Support bearbeitet werden, was zusätzliche Zeit in Anspruch nehmen kann, um sie zu prüfen und zu verarbeiten. Planen Sie zusätzliche Zeit ein, um mehrere Anfragen oder Anfragen für größere Erhöhungen zu bearbeiten.

Implementierungsschritte

- Implementieren Sie die automatische Überwachung von Service Quotas und geben Sie Warnmeldungen aus, wenn sich die Ressourcenauslastung Ihrer Workloads den Kontingentgrenzen nähert. Quota Monitor für AWS kann beispielsweise die automatische Überwachung von Service Quotas ermöglichen. Dieses Tool ist mit AWS Organizations integriert und wird mithilfe von Cloudformation StackSets bereitgestellt, sodass neue Konten ab Erstellung automatisch überwacht werden.
- Verwenden Sie Funktionen wie Service-Quotas-Anforderungsvorlagen oder AWS Control Tower, um die Einrichtung von Service Quotas für neue Konten zu vereinfachen.
- Erstellen Sie Dashboards über Ihre aktuelle Nutzung von Service Quotas in allen AWS-Konten und AWS-Regionen und verweisen Sie auf diese, wenn notwendig, um eine Überschreitung Ihrer Kontingente zu verhindern. Das Dashboard Trusted Advisor Organizational (TAO) ist Teil von Cloud Intelligence Dashboards und ermöglicht Ihnen die schnelle Erstellung dieser Dashboards.
- Verfolgen Sie Anfragen zur Erhöhung von Servicebegrenzungen nach. Konsolidierte Einblicke aus mehreren Konten (Consolidated Insights from Multiple Accounts (CIMA)) bieten Ihnen einen Überblick über alle Ihre Anforderungen auf Organisationsebene.
- Testen Sie die Generierung von Warnmeldungen und die Automatisierung von Anforderungen zur Quotenerhöhung, indem Sie niedrigere Kontingentschwellenwerte für Konten festlegen, die nicht in der Produktion verwendet werden. Führen Sie diese Tests nicht in einem Produktionskonto durch.

Ressourcen

Zugehörige bewährte Methoden:

- OPS10-BP07 Automatisieren von Reaktionen auf Ereignisse

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- AWS Marketplace: CMDB-Produkte zur Nachverfolgung von Limits
- AWS AWS Service Quotas (früher als Service Limits bezeichnet)
- Bewährte AWS Trusted Advisor-Prüfungsmethoden (siehe Abschnitt „Servicelimits“)
- Lösung für die Kontingentüberwachung n AWS – AWS-Lösung
- Was ist Service Quotas?
- Was sind Anforderungsvorlagen für Service Quotas?

Zugehörige Videos:

- AWS Live re:Inforce 2019 – Service Quotas
- Automatisierung der Erhöhungen von Servicegrenzen und des Unternehmenssupport mit AWS Control Tower

Zugehörige Tools:

- Quota Monitor for AWS

REL01-BP06 Sicherstellen eines ausreichenden Spielraums zwischen den aktuellen Kontingenten und der maximalen Nutzung, damit ein Failover möglich ist

In diesem Artikel wird erläutert, wie Sie den Abstand zwischen dem Ressourcenkontingent und Ihrer Nutzung beibehalten können und wie Ihr Unternehmen davon profitieren kann. Wenn Sie eine Ressource nicht mehr nutzen, wird diese Ressource möglicherweise noch auf ein Kontingent angerechnet. Dies kann zu einer fehlgeschlagenen oder nicht mehr erreichbaren Ressource führen. Überprüfen Sie, ob Ihre Kontingente die Überschneidung von ausgefallenen oder nicht zugreifbaren Ressourcen und deren Ersatz abdecken. Bei der Berechnung dieser Lücke sollten Sie Anwendungsfälle wie Netzwerkfehler, Fehler in der Availability Zone oder Fehler in einer Region berücksichtigen.

Gewünschtes Ergebnis: Kleine oder große Fehler bei Ressourcen oder der Ressourcenzugänglichkeit können innerhalb der aktuellen Service-Schwellenwerte abgedeckt werden. Zonenfehler, Netzwerkfehler oder sogar regionale Fehler wurden bei der Ressourcenplanung berücksichtigt.

Typische Anti-Muster:

- Es werden Servicekontingente auf Grundlage des aktuellen Bedarfs eingerichtet, ohne dass Failover-Szenarien berücksichtigt werden.
- Keine Berücksichtigung des Prinzips der statischen Stabilität bei der Berechnung des Spitzenkontingents für einen Service.
- Keine Berücksichtigung des Potenzials nicht zugreifbarer Ressourcen bei der Berechnung des für jede Region benötigten Gesamtkontingents.
- Keine Berücksichtigung der AWS-Grenzen für die Fehlerisolierung bei einigen Services und ihrer potenziell anormalen Nutzungsmuster.

Vorteile der Nutzung dieser bewährten Methode: Wenn die Verfügbarkeit von Anwendungen durch eine Service-Störung beeinträchtigt wird, bietet Ihnen die Cloud die Möglichkeit zur Implementierung von Strategien zur Abschwächung dieser Ereignisse oder der Wiederherstellung. Zu solchen Strategien gehört oft die Erstellung zusätzlicher Ressourcen, um ausgefallene oder unzugängliche Ressourcen zu ersetzen. Ihre Kontingent-Strategie muss diese Failover-Bedingungen berücksichtigen und würde nicht zu einer zusätzlichen Verschlechterung aufgrund des Erreichens von Service-Beschränkungen führen.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

Berücksichtigen Sie bei der Bewertung der Kontingente auch Failover-Fälle, die aufgrund einer Verschlechterung auftreten können. Die folgenden Arten von Failover-Fällen sollten in Betracht gezogen werden:

- Eine VPC, die gestört oder auf die nicht zugreifbar ist.
- Ein Subnetz, auf das nicht mehr zugegriffen werden kann.
- Eine Availability Zone wurde so stark beeinträchtigt, dass die Erreichbarkeit vieler Ressourcen beeinträchtigt ist.
- Verschiedene Netzwerk-Routen oder Ingress- und Egress-Punkte sind blockiert oder verändert.
- Eine Region ist so stark gestört, dass die Erreichbarkeit vieler Ressourcen beeinträchtigt ist.
- Es gibt mehrere Ressourcen, aber nicht alle sind von einem Fehler in einer Region oder einer Availability Zone betroffen.

Die Entscheidung für einen Failover ist für jede Situation und jeden Kunden individuell, da die Auswirkungen auf den Geschäftsbetrieb sehr unterschiedlich sein können. Wenn Sie sich jedoch operativ für einen Failover von Anwendungen oder Services entscheiden, müssen Sie sich vor dem Ereignis mit der Kapazitätsplanung der Ressourcen am Failover-Standort und den entsprechenden Kontingenten befassen.

Überprüfen Sie die Service-Kontingente für jeden Service und berücksichtigen Sie dabei die möglichen Spitzenwerte. Diese Spitzen können mit Ressourcen zusammenhängen, die über Netzwerkproblemen oder Berechtigungen zwar noch aktiv, aber nicht erreichbar sind. Nicht beendete aktive Ressourcen werden weiterhin auf das Kontingent des Service angerechnet.

Implementierungsschritte

- Vergewissern Sie sich, dass zwischen Ihrem Service-Kontingent und Ihrer maximalen Nutzung genügend Spielraum besteht, um einen Failover oder den Verlust der Erreichbarkeit aufzufangen.
- Ermitteln Sie die Servicekontingente unter Berücksichtigung von Bereitstellungsmustern, der Verfügbarkeitsanforderungen und des Nutzungsanstiegs.
- Fordern Sie bei Bedarf Kontingenterhöhungen an. Planen Sie den erforderlichen Zeitraums bis zur Bewilligung von Kontingenterhöhungen.
- Bestimmen Sie Ihre Anforderungen an die Zuverlässigkeit (Anzahl der Neunen).
- Legen Sie Fehlerszenarien fest (z. B. Verlust einer Komponente, Availability Zone oder Region).
- Führen Sie eine Bereitstellungsmethode ein (z. B. Canary, Blau/Grün-Bereitstellung, Rot/Schwarz-Bereitstellung oder schrittweise).
- Berücksichtigen Sie einen angemessenen Puffer (z. B. 15 %) in aktuellen Limits.
- Berücksichtigen Sie gegebenenfalls Berechnungen zur statischen Stabilität (zonenbezogen und regional).
- Planen Sie den Nutzungsanstieg (z. B. durch Überwachen des Nutzungstrends).
- Berücksichtigen Sie die Auswirkungen der statischen Stabilität für Ihre kritischsten Workloads. Bewerten Sie Ressourcen entsprechend eines statisch stabilen Systems in allen Regionen und Availability Zones.
- Ziehen Sie den Einsatz von On-Demand-Kapazitätsreservierungen in Betracht, um vor einem Failover Kapazitäten zu reservieren. Diese Strategie kann während kritischer Geschäftszeiten sinnvoll sein, um potenzielle Risiken bei der Beschaffung der richtigen Menge und Art von Ressourcen während eines Failovers zu verringern.

Ressourcen

Zugehörige bewährte Methoden:

- REL01-BP01 Kenntnis von Service Quotas und Einschränkungen
- REL01-BP02 Verwalten von Service Quotas für mehrere Konten und Regionen
- REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur
- REL01-BP04 Kontingente überwachen und verwalten
- REL01-BP05 Automatisieren der Kontingentverwaltung
- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten
- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler
- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (früher als Service Limits bezeichnet)
- AWS Trusted Advisor Bewährte -Prüfungsmethoden (siehe Abschnitt „Servicelimits)
- AWS Limit Monitor in AWS Answers
- Amazon EC2 Service Limits
- Was ist Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas-Benutzerhandbuch
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS für Data
- What is Continuous Integration?
- What is Continuous Delivery?

- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Aktionen, Ressourcen und Bedingungsschlüssel für Service Quotas

Zugehörige Videos:

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

Zugehörige Tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

Planen der Netzwerktopologie

Workloads existieren oft in mehreren Umgebungen. Dazu gehören mehrere Cloud-Umgebungen (sowohl öffentlich zugänglich als auch privat) und möglicherweise Ihre bestehende Rechenzentrumsinfrastruktur. Die Pläne müssen Netzwerkaspekte umfassen, wie z. B. Konnektivität

innerhalb und zwischen Systemen, Verwaltung öffentlicher und privater IP-Adressen und Auflösung von Domainnamen.

Wenn Sie eine Architektur für Systeme entwickeln und dazu IP-Adressen-basierte Netzwerke verwenden, müssen Sie die Netzwerktopologie und Adresszuweisung mit Blick auf das künftige Wachstum und die Integration mit anderen Systemen und zugehörigen Netzwerken planen.

Mit Amazon Virtual Private Cloud (Amazon VPC) können Sie einen privaten, isolierten Teil der AWS-Cloud bereitstellen. Dort können Sie AWS-Ressourcen in einem virtuellen Netzwerk starten.

Best Practices

- REL02-BP01 Bereitstellen einer hochverfügbaren Netzwerkkonnektivität für öffentliche Endpunkte der Workload
- REL02-BP02 Bereitstellen redundanter Konnektivität zwischen privaten Netzwerken in der Cloud und in On-Premises-Umgebungen
- REL02-BP03 Berücksichtigen von Erweiterungen und Verfügbarkeit bei der Zuweisung von IP-Adressen für Subnetze
- REL02-BP04 Vorziehen von Hub-and-Spoke-Topologien gegenüber M-zu-N-Netzen
- REL02-BP05 Erzwingen von sich nicht überschneidenden privaten IP-Adressbereichen in allen privaten Adressbereichen, in denen eine Verbindung besteht

REL02-BP01 Bereitstellen einer hochverfügbaren Netzwerkkonnektivität für öffentliche Endpunkte der Workload

Der Aufbau einer hochverfügbaren Netzwerkkonnektivität zu öffentlichen Endpunkten Ihres Workloads kann Ihnen helfen, Ausfallzeiten aufgrund von Konnektivitätsverlusten zu reduzieren und die Verfügbarkeit und SLA Ihres Workloads zu verbessern. Verwenden Sie dazu hochverfügbares DNS, Content Delivery Networks (CDNs), API-Gateways, Load-Balancing oder Reverse-Proxies.

Gewünschtes Ergebnis: Es ist von entscheidender Bedeutung, eine hochverfügbare Netzwerkkonnektivität für Ihre öffentlichen Endpunkte zu planen, aufzubauen und in Betrieb zu nehmen. Wenn Ihr Workload aufgrund eines Konnektivitätsverlustes nicht mehr erreichbar ist, sehen Ihre Kunden Ihr System als ausgefallen an – selbst wenn Ihr Workload läuft und verfügbar ist. Durch die Kombination einer hochverfügbaren und stabilen Netzwerkkonnektivität für die öffentlichen Endpunkte Ihres Workloads mit einer stabilen Architektur für Ihren Workload selbst können Sie Ihren Kunden die bestmögliche Verfügbarkeit und das bestmögliche Serviceniveau bieten.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, AWS Lambda-Funktions-URLs, AWS AppSync-APIs und Elastic Load Balancing (ELB) bieten alle hochverfügbare öffentliche Endpunkte. Amazon Route 53 bietet einen hochverfügbaren DNS-Service für die Auflösung von Domain-Namen, um sicherzustellen, dass die Adressen Ihrer öffentlichen Endpunkte aufgelöst werden können.

Sie können außerdem AWS Marketplace-Software-Appliances für das Load-Balancing und für Proxys nutzen.

Typische Anti-Muster:

- Entwurf eines hochverfügbaren Workloads, ohne eine DNS- und Netzwerkkonnektivität mit hoher Verfügbarkeit einzuplanen.
- Verwendung öffentlicher Internetadressen auf einzelnen Instances oder Containern und Verwalten der Konnektivität zu diesen per DNS.
- Verwendung von IP-Adressen anstelle von Domain-Namen zur Lokalisierung von Services.
- Keine Tests von Szenarien, in denen die Konnektivität zu Ihren öffentlichen Endpunkten verloren geht.
- Keine Analyse des Bedarfs für den Netzwerkdurchsatz und die Verteilungsmuster im Netzwerk.
- Keine Tests und Planungen für Szenarien, in denen die Internet-Netzwerkkonnektivität zu Ihren öffentlichen Endpunkten der Workloads unterbrochen werden könnte.
- Bereitstellen von Inhalten (z. B. Webseiten, statische Komponenten oder Mediendateien) für ein großes geografisches Gebiet ohne Verwendung eines Content-Delivery-Networks.
- Keine Planung für Distributed Denial of Service (DDoS)-Angriffe. Bei DDoS-Angriffen besteht die Gefahr, dass der legitime Datenverkehr unterbrochen wird und die Verfügbarkeit für Ihre Benutzer sinkt.

Vorteile der Nutzung dieser bewährten Methode: Die Planung einer hochverfügbaren und stabilen Netzwerkkonnektivität stellt sicher, dass Ihr Workload für Ihre Benutzer zugreifbar und verfügbar ist.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Das Wichtigste beim Aufbau einer hochverfügbaren Netzwerkkonnektivität zu Ihren öffentlichen Endpunkten ist das Routing des Datenverkehrs. Um sicherzustellen, dass Ihr Datenverkehr die Endpunkte erreichen kann, muss das DNS in der Lage sein, die Domain-Namen in die

entsprechenden IP-Adressen aufzulösen. Verwenden Sie ein hochverfügbares und skalierbares Domain Name System (DNS) wie Amazon Route 53, um die DNS-Einträge Ihrer Domäne zu verwalten. Sie können außerdem die von Amazon Route 53 bereitgestellten Zustandsprüfungen verwenden. Die Zustandsprüfungen überprüfen, ob Ihre Anwendung erreichbar, verfügbar und funktionstüchtig ist. Sie können so eingerichtet werden, dass sie das Verhalten Ihres Benutzers nachahmen, z. B. das Anfordern einer Webseite oder einer bestimmten URL. Im Falle eines Fehlers reagiert Amazon Route 53 auf DNS-Auflösungsanfragen und leitet den Datenverkehr nur an Health-Endpunkte weiter. Sie können außerdem die von Amazon Route 53 angebotenen Funktionen für Geo-DNS und latenzbasiertes Routing nutzen.

Um zu überprüfen, ob Ihr Workload selbst hochverfügbar ist, verwenden Sie Elastic Load Balancing (ELB). Amazon Route 53 kann verwendet werden, um den Datenverkehr an ELB zu leiten, das den Datenverkehr an die Ziel-Computing-Instances verteilt. Sie können Amazon API Gateway außerdem zusammen mit AWS Lambda für eine Serverless-Lösung verwenden. Kunden können Workloads zudem in mehreren AWS-Regionen ausführen. Mit einem Multi-Site Aktiv/Aktiv-Muster kann der Workload den Datenverkehr aus mehreren Regionen bedienen. Bei einem Multi-Site Aktiv/Passiv-Muster bedient der Workload den Datenverkehr aus der aktiven Region, während die Daten in die sekundäre Region repliziert werden, die im Falle eines Fehlers in der primären Region aktiv wird. Mit Amazon Route 53-Zustandsprüfungen können Sie dann das DNS-Failover von einem beliebigen Endpunkt in einer primären Region zu einem Endpunkt in einer sekundären Region steuern und so sicherstellen, dass Ihr Workload erreichbar und für Ihre Benutzer verfügbar ist.

Amazon CloudFront bietet eine einfache API für die Verteilung von Inhalten mit geringer Latenz und hohen Datenübertragungsraten, indem Anfragen über ein Netzwerk von Edge-Standorten auf der ganzen Welt bedient werden. Content Delivery Networks (CDNs) dienen den Kunden, indem sie Inhalte bereitstellen, die sich in der Nähe des Benutzers befinden oder dort zwischengespeichert werden. Dies verbessert auch die Verfügbarkeit Ihrer Anwendung, da die Last der Inhalte von Ihren Servern auf die Edge-Standorte von CloudFront verlagert wird. Die Edge-Standorte und regionalen Edge-Caches halten zwischengespeicherte Kopien Ihrer Inhalte in der Nähe Ihrer Benutzer vor, was einen schnellen Abruf ermöglicht und die Erreichbarkeit und Verfügbarkeit Ihres Workloads erhöht.

Bei Workloads mit geografisch verteilten Benutzern hilft AWS Global Accelerator Ihnen, die Verfügbarkeit und Leistung der Anwendungen zu verbessern. AWS Global Accelerator bietet statische Anycast-IP-Adressen, die als fester Zugangspunkt zu Ihrer Anwendung dienen, die in einer oder mehreren AWS-Regionen gehostet wird. Dadurch kann der Datenverkehr so nah wie möglich an Ihren Benutzern in das globale AWS Netzwerk geleitet werden, was die Erreichbarkeit und Verfügbarkeit Ihres Workloads verbessert. AWS Global Accelerator überwacht außerdem den Zustand Ihrer Anwendungsendpunkte mithilfe von TCP-, HTTP- und HTTPS-Zustandsprüfungen.

Jede Änderung im Zustand oder in der Konfiguration Ihrer Endpunkte leitet den Benutzerverkehr auf funktionierende Endpunkte weiter, die Ihren Benutzern die beste Leistung und Verfügbarkeit bieten. Darüber hinaus verfügt AWS Global Accelerator über ein fehlerisolierendes Design, das zwei statische IPv4-Adressen verwendet, die von unabhängigen Netzwerkzonen bedient werden und die Verfügbarkeit Ihrer Anwendungen erhöhen.

AWS bietet AWS Shield Standard, um Kunden vor DDoS-Angriffen zu schützen. Shield Standard wird automatisch aktiviert und schützt vor gängigen Infrastrukturangriffen (Layer 3 und 4) wie SYN/UDP-Floods und Reflection-Angriffen, um die hohe Verfügbarkeit Ihrer Anwendungen auf AWS zu unterstützen. Für zusätzlichen Schutz vor ausgefeilteren und größeren Angriffen (wie UDP-Floods), State-Exhaustion-Angriffen (wie TCP-SYN-Floods) und zum Schutz Ihrer Anwendungen, die auf Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator und Amazon Route 53 ausgeführt werden, können Sie AWS Shield Advanced verwenden. Zum Schutz vor Angriffen auf der Anwendungsebene wie HTTP-POST- oder GET-Floods verwenden Sie AWS WAF. AWS WAF kann IP-Adressen, HTTP-Header, HTTP-Body, URI-Strings, SQL-Injections und Cross-Site-Scripting-Bedingungen verwenden, um zu bestimmen, ob eine Anfrage blockiert oder zugelassen werden soll.

Implementierungsschritte

1. Amazon Route 53 ist ein hochverfügbarer und skalierbarer Domain Name System (DNS)-Web-Service. Route 53 verbindet Benutzeranfragen mit Internetanwendungen, die auf AWS oder On-Premises ausgeführt werden. Weitere Informationen finden Sie unter Konfigurieren von Amazon Route 53 als DNS-Service..
2. Richten Sie Zustandsprüfungen ein: Wenn Sie Amazon Route 53 verwenden, vergewissern Sie sich, dass nur korrekt funktionierende Ziele auflösbar sind. Starten Sie mit der Erstellung von Route 53-Zustandsprüfungen und der Konfiguration des DNS-Failovers. Bei der Einrichtung von Zustandsprüfungen sind die folgenden Aspekte zu beachten:
   a. So ermittelt Amazon Route 53, ob eine Zustandsprüfung fehlerfrei ist
   b. Erstellen, Aktualisieren und Löschen von Zustandsprüfungen
   c. Den Status von Zustandsprüfungen überwachen und Benachrichtigungen erhalten
   d. Bewährte Methoden für Amazon Route 53 DNS
3. Verbinden Sie Ihren DNS-Service mit Ihren Endpunkten.
   a. Wenn Sie Elastic Load Balancing als Ziel für Ihren Datenverkehr verwenden, erstellen Sie einen Alias-Eintrag mit Amazon Route 53, der auf den regionalen Endpunkt Ihres Load-Balancers verweist. Setzen Sie bei der Erstellung des Alias-Eintrags die Option „Zielzustand evaluieren“

auf „Ja“. Setzen Sie bei der Erstellung des Alias-Eintrags die Option „Zielzustand evaluieren“ auf „Ja“.

b. Verwenden Sie bei der Nutzung von API Gateway für Serverless-Workloads oder private APIs Route 53, um den Datenverkehr zu API Gateway zu routen.

4. Entscheiden Sie sich für ein Content Delivery Netzwerk.

   a. Informieren Sie sich zunächst über die Art und Weise, wie CloudFront-Inhalte über Edge-Standorte in der Nähe des Benutzers bereitgestellt werden.

   b. Starten Sie mit einer einfachen CloudFront-Verteilung. CloudFront weiß dann, von wo aus die Inhalte ausgeliefert werden sollen, und kennt die Details zur Nachverfolgung und Verwaltung der Content-Bereitstellung. Die folgenden Aspekte sollten Sie kennen und berücksichtigen, wenn Sie die CloudFront-Verteilung einrichten:

      i. Funktionsweise der Zwischenspeicherung mit CloudFront-Edge-Standorten

      ii. Erhöhen des Anteils der Anforderungen, die direkt von den CloudFront-Caches bereitgestellt werden (Cache-Trefferrate)

      iii. Verwenden von Amazon CloudFront Origin Shield

      iv. Optimieren der Hochverfügbarkeit mit CloudFront-Ursprungs-Failover

5. Einrichten des Schutzes auf der Anwendungsebene: AWS WAF hilft Ihnen, sich gegen gängige Web-Exploits und Bots zu schützen, die die Verfügbarkeit beeinträchtigen, die Sicherheit gefährden oder übermäßig viele Ressourcen verbrauchen können. Um ein tieferes Verständnis zu erlangen, lesen Sie How AWS WAF works. Wenn Sie bereit sind, den Schutz vor HTTP-POST- und -GET-Floods auf der Anwendungsebene zu implementieren, lesen Sie Getting started with AWS WAF. Sie können außerdem AWS WAF mit CloudFront verwenden. In der Dokumentation erfahren Sie, wie AWS WAF mit Amazon CloudFront-Funktionen arbeitet.

6. Richten Sie einen zusätzlichen DDoS-Schutz ein: Standardmäßig erhalten alle Kunden von AWS mit AWS Shield Standard ohne zusätzliche Kosten einen Schutz gegen die gängigsten DDoS-Angriffe auf Netzwerk- und Transportebene, die sich gegen Ihre Website oder Anwendung richten. Für zusätzlichen Schutz von Anwendungen, die auf Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator und Amazon Route 53 ausgeführt werden, können Sie AWS Shield Advanced einsetzen und sich Beispiele für DDoS-resistente Architekturen ansehen. Um Ihren Workload und Ihre öffentlichen Endpunkte vor DDoS-Angriffen zu schützen, lesen Sie Getting started with AWS Shield Advanced.

Ressourcen

Zugehörige bewährte Methoden:

- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL11-BP04 Nutzen der Datenebene und nicht der Steuerebene während der Wiederherstellung
- REL11-BP06 Senden von Benachrichtigungen, wenn sich Ereignisse auf die Verfügbarkeit auswirken

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Marketplace für Netzwerkinfrastruktur
- Was ist AWS Global Accelerator?
- Was ist Amazon CloudFront?
- Was ist Amazon Route 53?
- Was ist Elastic Load Balancing?
- Network Connectivity capability - Establishing Your Cloud Foundations
- Was ist Amazon API Gateway?
- Was sind AWS WAF, AWS Shield und AWS Firewall Manager?
- Was ist Amazon Application Recovery Controller?
- Benutzerdefinierte Zustandsprüfungen für das DNS-Failover konfigurieren

Zugehörige Videos:

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

Zugehörige Beispiele:

- Notfallwiederherstellung mit Amazon Application Recovery Controller (ARC)

- AWS Global Accelerator Workshop

REL02-BP02 Bereitstellen redundanter Konnektivität zwischen privaten Netzwerken in der Cloud und in On-Premises-Umgebungen

Implementieren Sie Redundanz in Ihren Verbindungen zwischen privaten Netzwerken in der Cloud und On-Premises-Umgebungen, um die Stabilität der Konnektivität zu erreichen. Dies kann erreicht werden, indem zwei oder mehr Verbindungen und Datenverkehrspfade bereitgestellt werden, sodass die Konnektivität bei Netzwerkausfällen erhalten bleibt.

Typische Anti-Muster:

- Sie verlassen sich auf nur eine Netzwerkverbindung, was zu einer einzigen Fehlerquelle führt.
- Sie verwenden nur einen VPN-Tunnel oder mehrere Tunnel, die in derselben Availability Zone enden.
- Sie verlassen sich bei der VPN-Konnektivität auf einen ISP, was bei ISP-Ausfällen zu kompletten Ausfällen führen kann.
- Keine Implementierung dynamischer Routing-Protokolle wie BGP, die für die Umleitung des Datenverkehrs bei Netzwerkunterbrechungen von entscheidender Bedeutung sind.
- Sie ignorieren die Bandbreitenbeschränkungen von VPN-Tunneln und überschätzen deren Backup-Fähigkeiten.

Vorteile der Nutzung dieser bewährten Methode: Durch die Implementierung redundanter Konnektivität zwischen Ihrer Cloud-Umgebung und Ihrer Unternehmens- bzw. On-Premises-Umgebung wird die sichere Kommunikation der abhängigen Services zwischen den beiden Umgebungen gewährleistet.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Wenn Sie AWS Direct Connect verwenden, um Ihr On-Premises-Netzwerk mit AWS zu verbinden, können Sie maximale Netzwerkstabilität (SLA von 99,99 %) erreichen, indem Sie separate Verbindungen verwenden, die auf verschiedenen Geräten an mehr als einem On-Premises-Standort und mehr als einem AWS Direct Connect-Standort enden. Diese Topologie bietet Widerstandsfähigkeit gegen Geräteausfälle, Verbindungsprobleme und komplette

Standortausfälle. Alternativ können Sie eine hohe Ausfallsicherheit (SLA von 99,9 %) erreichen, indem Sie zwei einzelne Verbindungen zu mehreren Standorten verwenden (jeder On-Premises-Standort ist mit einem einzigen Direct-Connect-Standort verbunden). Dieser Ansatz schützt vor Verbindungsunterbrechungen, die durch getrennte Glasfaserkabel oder Geräteausfälle verursacht werden, und trägt dazu bei, die Auswirkungen kompletter Standortausfälle zu mindern. Das Direct Connect Resiliency Toolkit unterstützt Sie beim Entwerfen Ihrer AWS Direct Connect-Topologie.

Sie können auch erwägen, dass AWS Site-to-Site VPN auf einem AWS Transit Gateway endet, um ein kostengünstiges Backup Ihrer primären Verbindung mit AWS Direct Connect zu erhalten. Dieses Setup ermöglicht Equal-Cost-Multipath (ECMP)-Routing über mehrere VPN-Tunnel und ermöglicht einen Durchsatz von bis zu 50 Gbit/s, obwohl jeder VPN-Tunnel auf 1,25 Gbit/s begrenzt ist. Es ist jedoch wichtig zu beachten, dass AWS Direct Connect immer noch die effektivste Wahl ist, um Netzwerkunterbrechungen zu minimieren und eine stabile Konnektivität bereitzustellen.

Wenn Sie VPNs über das Internet verwenden, um Ihre Cloud-Umgebung mit Ihrem On-Premises-Rechenzentrum zu verbinden, konfigurieren Sie zwei VPN-Tunnel als Teil einer einzigen Site-to-Site-VPN-Verbindung. Jeder Tunnel sollte aus Gründen der Hochverfügbarkeit in einer anderen Availability Zone enden und redundante Hardware verwenden, um Ausfälle von On-Premises-Geräten zu verhindern. Erwägen Sie außerdem mehrere Internetverbindungen von verschiedenen Internetdienstanbietern (ISPs) an Ihrem On-Premises-Standort, um eine vollständige Unterbrechung der VPN-Konnektivität durch den Ausfall eines einzigen ISP zu vermeiden. Die Auswahl von ISPs mit unterschiedlichem Routing und Infrastruktur, insbesondere solchen mit separaten physischen Pfaden zu AWS-Endpunkten, sorgt für eine hohe Konnektivitätsverfügbarkeit.

Neben der physischen Redundanz mit mehreren AWS Direct Connect-Verbindungen und VPN-Tunneln (oder einer Kombination aus beiden) ist auch die Implementierung des dynamischen Routings des Border Gateway Protocol (BGP) von entscheidender Bedeutung. Dynamisches BGP ermöglicht die automatische Umleitung des Datenverkehrs von einem Pfad zum nächsten, basierend auf Netzwerkbedingungen in Echtzeit und konfigurierten Richtlinien. Dieses dynamische Verhalten ist besonders vorteilhaft zur Aufrechterhaltung der Netzwerkverfügbarkeit und Servicekontinuität bei Verbindungs- oder Netzwerkausfällen. Es wählt schnell alternative Pfade aus und verbessert so die Ausfallsicherheit und Zuverlässigkeit des Netzwerks.

Implementierungsschritte

- Erwerben Sie hochverfügbare Konnektivität zwischen AWS und Ihrer On-Premises-Umgebung.
  - Verwenden Sie mehrere AWS Direct Connect-Verbindungen oder VPN-Tunnel zwischen separat bereitgestellten privaten Netzwerken.

  - Verwenden Sie für eine hohe Verfügbarkeit mehrere Direct Connect-Standorte.
  - Wenn Sie mehrere AWS-Regionen verwenden, sorgen Sie in mindestens zwei davon für Redundanz.
- Verwenden Sie AWS Transit Gateway, wenn möglichy, um Ihre VPN-Verbindung zu beenden.
- Beurteilen Sie AWS Marketplace-Appliances, um VPNs zu beenden oder erweitern Sie Ihr SD-WAN auf AWS. Stellen Sie bei Verwendung von AWS Marketplace-Appliances redundante Instances bereit, um eine hohe Verfügbarkeit in verschiedenen Availability Zones zu gewährleisten.
- Stellen Sie auf Ihrer On-Premises-Umgebung eine redundante Verbindung her.
  - Möglicherweise benötigen Sie redundante Verbindungen zu mehreren AWS-Regionen, um die erforderliche Verfügbarkeit zu gewährleisten.
  - Verwenden Sie das Direct Connect Resiliency Toolkit, um loszulegen.

Ressourcen

Zugehörige Dokumente:

- AWS Direct Connect Resiliency Recommendations
- Using Redundant Site-to-Site VPN Connections to Provide Failover
- Routing policies and BGP communities
- Active/Active and Active/Passive Configurations in AWS Direct Connect
- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Marketplace für Netzwerkinfrastruktur
- Whitepaper Amazon Virtual Private Cloud Connectivity Options
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- Using redundant Site-to-Site VPN connections to provide failover
- Using the Direct Connect Resiliency Toolkit to get started
- VPC Endpoints and VPC Endpoint Services (AWS PrivateLink)
- Was ist Amazon VPC?
- What is a transit gateway?
- Was ist AWS Site-to-Site VPN?
- Working with Direct Connect gateways

Zugehörige Videos:

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

REL02-BP03 Berücksichtigen von Erweiterungen und Verfügbarkeit bei der Zuweisung von IP-Adressen für Subnetze

Die IP-Adressbereiche für Amazon VPC müssen ausreichend groß sein, um die Anforderungen eines Workloads zu erfüllen. Dabei sind zukünftige Erweiterungen und Zuweisungen von IP-Adressen zu Subnetzen in verschiedenen Availability Zones zu berücksichtigen. Dies betrifft Load Balancer, EC2-Instances sowie containerbasierte Anwendungen.

Wenn Sie Ihre Netzwerktopologie planen, besteht der erste Schritt in der Definition des IP-Adressbereichs. Private IP-Adressbereiche (gemäß RFC 1918-Richtlinien) sollten jeder VPC zugewiesen werden. Berücksichtigen Sie im Rahmen dieses Prozesses die folgenden Anforderungen:

- Ermöglichen Sie einen IP-Adressbereich für mehr als eine VPC pro Region.
- Planen Sie innerhalb einer VPC Platz für mehrere Subnetze ein, damit Sie mehrere Availability Zones abdecken können.
- Lassen Sie für eine zukünftige Erweiterung stets Raum für nicht verwendete CIDR-Blöcke innerhalb einer VPC.
- Stellen Sie sicher, dass ein IP-Adressbereich vorhanden ist, um die Anforderungen von temporären Amazon EC2-Instances zu erfüllen, die Sie möglicherweise verwenden, z. B. Spot-Flotten für Machine Learning, Amazon EMR-Cluster oder Amazon Redshift-Cluster. Ähnliche Überlegungen sollten für Kubernetes-Cluster wie Amazon Elastic Kubernetes Service (Amazon EKS) getroffen werden, da jedem Kubernetes-Pod standardmäßig eine routbare Adresse aus dem VPC-CIDR-Block zugewiesen wird.
- Beachten Sie, dass die ersten vier IP-Adressen und die letzte IP-Adresse in jedem Subnetz-CIDR-Block reserviert und nicht für Sie verfügbar sind.
- Beachten Sie, dass der VPC CIDR-Block, der anfänglich Ihrer VPC zugewiesen war, nicht geändert oder gelöscht werden kann. Sie können der VPC jedoch zusätzliche, nicht überlappende CIDR-Blöcke hinzufügen. IPv4-CIDRs für Subnetze können nicht geändert werden, IPv6 CIDRs jedoch schon.
- Der größte mögliche VPC-CIDR-Block entspricht /16 und der kleinste /28.

- Berücksichtigen Sie andere verbundene Netzwerke (VPC, On-Premises oder sonstige Cloud-Anbieter) und stellen Sie sicher, dass sich der IP-Adressraum nicht überschneidet. Weitere Informationen finden Sie unter Erzwingen von sich nicht überschneidenden privaten IP-Adressbereichen in allen privaten Adressbereichen, in denen eine Verbindung besteht.

Gewünschtes Ergebnis: Ein skalierbares IP-Subnetz kann Ihnen helfen, zukünftiges Wachstum zu bewältigen und unnötige Verschwendung zu vermeiden.

Typische Anti-Muster:

- Wenn zukünftiges Wachstum nicht berücksichtigt wird, sind die CIDR-Blöcke zu klein und müssen neu konfiguriert werden, was zu Ausfallzeiten führen kann.
- Es wird falsch eingeschätzt, wie viele IP-Adressen ein Elastic Load Balancer verwenden kann.
- Es werden viele Load Balancer mit hohem Datenverkehr in denselben Subnetzen bereitgestellt.
- Es werden automatische Skalierungsmechanismen verwendet, während der Verbrauch von IP-Adressen nicht überwacht wird.
- Die Definition übermäßig großer CIDR-Bereiche liegt weit über den zukünftigen Wachstumserwartungen, was zu Schwierigkeiten beim Peering mit anderen Netzwerken mit überlappenden Adressbereichen führen kann.

Vorteile der Nutzung dieser bewährten Methode: So wird sichergestellt, dass Sie das Wachstum Ihrer Workloads bewältigen können und beim Hochskalieren weiterhin die entsprechende Verfügbarkeit bereitstellen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Berücksichtigen Sie bei der Planung Ihres Netzwerks Ihr zukünftiges Wachstum, die Einhaltung gesetzlicher Vorschriften sowie die Kompatibilität mit anderen Netzwerken. Das Wachstum kann unterschätzt werden, gesetzliche Vorschriften können sich ändern, und bei Unternehmensübernahmen oder privaten Netzwerkverbindungen kann die Implementierung ohne fundierte Planung zur Herausforderung werden.

- Wählen Sie relevante AWS-Konten und Regionen anhand von Serviceanforderungen, regulatorischen Anforderungen sowie Anforderungen für die Latenz und die Notfallwiederherstellung aus.

- Identifizieren Sie Ihre Anforderungen bezüglich regionaler VPC-Bereitstellungen.
- Ermitteln Sie die erforderliche Größe der VPCs.
  - Ermitteln Sie, ob Multi-VPC-Konnektivität bereitgestellt werden soll.
    - Was ist ein Transit-Gateway?
    - Multi-VPC-Konnektivität in einer Region
  - Ermitteln Sie, ob aufgrund von Compliance-Anforderungen getrennte Netzwerke erforderlich sind.
  - Erstellen Sie VPCs mit CIDR-Blöcken in geeigneter Größe, um Ihren aktuellen und zukünftigen Anforderungen gerecht zu werden.
    - Wenn Sie unbekannte Wachstumsprognosen haben, sollten Sie sich für größere CIDR-Blöcke entscheiden, um das Potenzial einer zukünftigen Neukonfiguration zu verringern.
  - Erwägen Sie die Verwendung von IPv6-Adressierung für Subnetze als Teil einer Dual-Stack-VPC. IPv6 eignet sich gut für den Einsatz in privaten Subnetzen, die Flotten kurzlebiger Instances oder Container enthalten, für die andernfalls eine große Anzahl von IPv4-Adressen erforderlich wäre.

Ressourcen

Zugehörige bewährte Methoden für Well-Architected:

- REL02-BP05 Erzwingen von sich nicht überschneidenden privaten IP-Adressbereichen in allen privaten Adressbereichen, in denen eine Verbindung besteht

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Marketplace für Netzwerkinfrastruktur
- Whitepaper Amazon Virtual Private Cloud Connectivity Options
- Multiple data center HA network connectivity
- Multi-VPC-Konnektivität in einer Region
- Was ist Amazon VPC?
- IPv6 auf AWS
- IPv6 in Referenzarchitekturen

- Amazon Elastic Kubernetes Service launches IPv6 support
- Empfehlungen für Ihre VPC – Classic Load Balancer
- Availability-Zone-Subnetze – Application Load Balancer
- Availability Zones – Network Load Balancer

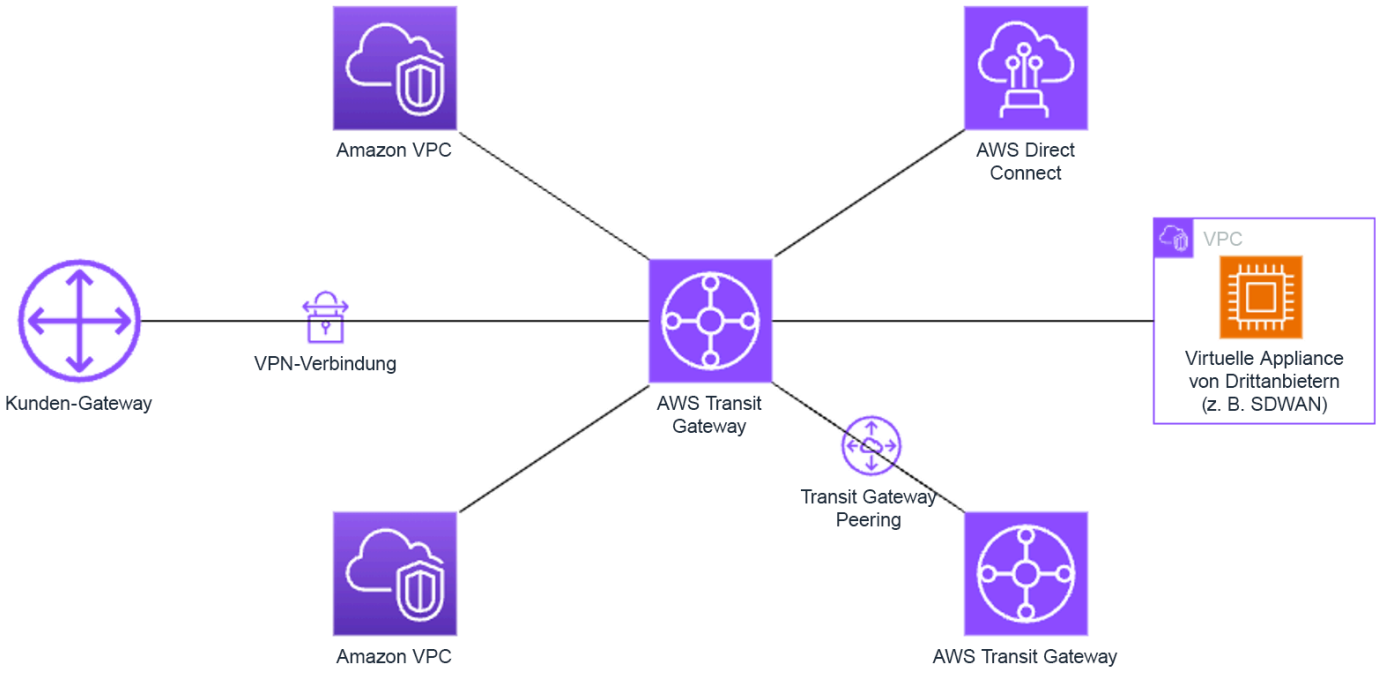
Zugehörige Videos:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)
- AWS re:Invent 2023: AWS Ready for what's next? Designing networks for growth and flexibility (NET310)

REL02-BP04 Vorziehen von Hub-and-Spoke-Topologien gegenüber M-zu-N-Netzen

Wenn Sie mehrere private Netzwerke wie Virtual Private Clouds (VPCs) und On-Premises-Netzwerke verbinden, sollten Sie sich für eine Hub-and-Spoke-Topologie statt für eine verflochtene Topologie entscheiden. Im Gegensatz zu verflochtenen Topologien, bei denen jedes Netzwerk direkt mit dem anderen verbunden ist, was die Komplexität und den Verwaltungsaufwand erhöht, zentralisiert die Hub-and-Spoke-Architektur Verbindungen über einen einzigen Hub. Diese Zentralisierung vereinfacht die Netzwerkstruktur und verbessert deren Bedienbarkeit, Skalierbarkeit und Kontrolle.

AWS Transit Gateway ist ein verwalteter, skalierbarer und hochverfügbarer Service, der für den Aufbau von Hub-and-Spoke-Netzwerken auf AWS entwickelt wurde. Er dient als zentraler Knotenpunkt Ihres Netzwerks, der Netzwerksegmentierung, zentralisiertes Routing und die vereinfachte Verbindung zu Cloud- und On-Premises-Umgebungen ermöglicht. Die folgende Abbildung zeigt, wie Sie Ihre Hub-and-Spoke-Topologie mit AWS Transit Gateway entwickeln können.



Gewünschtes Ergebnis: Sie haben Ihre Virtual Private Clouds (VPCs) und On-Premises-Netzwerke über einen zentralen Hub verbunden. Sie konfigurieren Ihre Peering-Verbindungen über den Hub, der als hoch skalierbarer Cloud-Router fungiert. Das Routing ist vereinfacht, da Sie nicht mit komplexen Peering-Beziehungen arbeiten müssen. Der Datenverkehr zwischen Netzwerken ist verschlüsselt und Sie haben die Möglichkeit, Netzwerke zu isolieren.

Typische Anti-Muster:

- Sie erstellen komplexe Netzwerk-Peering-Regeln.
- Sie stellen Routen zwischen Netzwerken bereit, die nicht miteinander kommunizieren sollten (z. B. separate Workloads, die keine gegenseitigen Abhängigkeiten haben).
- Die Verwaltung der Hub-Instance ist ineffektiv.

Vorteile der Nutzung dieser bewährten Methode: Wenn die Anzahl der verbundenen Netzwerke zunimmt, wird die Verwaltung und Erweiterung der Mesh-Konnektivität immer schwieriger.
Eine Mesh-Architektur bringt zusätzliche Herausforderungen mit sich, z. B. zusätzliche Infrastrukturkomponenten, Konfigurationsanforderungen und Überlegungen zur Bereitstellung. Das Mesh bringt außerdem zusätzlichen Aufwand für die Verwaltung und Überwachung der Komponenten der Datenebene und der Steuerebene mit sich. Sie müssen darüber nachdenken, wie eine hohe

Verfügbarkeit der Mesh-Architektur gewährleistet werden kann, wie der Zustand und die Leistung des Mesh überwacht werden können und wie mit Upgrades der Mesh-Komponenten umzugehen ist.

Ein Hub-and-Spoke-Modell hingegen ermöglicht die zentrale Weiterleitung des Datenverkehrs über mehrere Netzwerke hinweg. Es bietet einen einfacheren Ansatz für die Verwaltung und Überwachung der Komponenten der Datenebene und der Steuerebene.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Erstellen Sie ein Network-Services-Konto, wenn keins vorhanden ist. Platzieren Sie den Hub im Network-Services-Konto der Organisation. Dies ermöglicht die zentrale Verwaltung des Hubs durch Netzwerktechniker.

Im Hub-and-Spoke-Modell dient der Hub als virtueller Router für den Datenverkehr zwischen Ihren Virtual Private Clouds (VPCs) und On-Premises-Netzwerken. Dies reduziert die Netzwerkkomplexität und vereinfacht die Behebung von Netzwerkproblemen.

Berücksichtigen Sie Ihr Netzwerkdesign, einschließlich der VPCs, AWS Direct Connect und die Site-to-Site-VPN-Verbindungen, die Sie herstellen möchten.

Verwenden Sie für jede angefügte Transit-Gateway-VPC ein separates Subnetz. Verwenden Sie für jedes Subnetz einen kleinen CIDR-Wert (z. B. /28), damit Sie mehr Adressraum für Computing-Ressourcen haben. Erstellen Sie eine Netzwerk-Zugriffskontrollliste (ACL) und weisen Sie diese allen Subnetzen zu, die mit dem Hub verbunden sind. Halten Sie die Netzwerk-ACL sowohl in der Richtung für eingehenden als auch in der Richtung für ausgehender Datenverkehr geöffnet.

Entwerfen und implementieren Sie Ihre Routing-Tabellen so, dass Routen nur zwischen Netzwerken bereitgestellt werden, die miteinander kommunizieren sollen. Lassen Sie Routen zwischen Netzwerken aus, die nicht miteinander kommunizieren sollen (z. B. zwischen separaten Workloads, die keine gegenseitigen Abhängigkeiten haben).

Implementierungsschritte

1. Planen Sie Ihr Netzwerk. Ermitteln Sie, welche Netzwerke Sie verbinden möchten, und stellen Sie sicher, dass diese keine sich überschneidenden CIDR-Bereiche teilen.
2. Erstellen Sie ein AWS Transit Gateway und hänfügen Sie Ihre VPCs an.
3. Erstellen Sie VPN-Verbindungen oder Direct-Connect-Gateways und verknüpfen Sie diese mit dem Transit-Gateway. wenn notwendig.

4. Definieren Sie durch die Konfiguration Ihrer Routing-Tabelle für den Transit-Gateway, wie der Datenverkehr zwischen den verbundenen VPCs und anderen Verbindungen geleitet wird.
5. Verwenden Sie Amazon CloudWatch, um Konfigurationen zu überwachen und bei Bedarf zur Leistungs- und Kostenoptimierung anzupassen.

Ressourcen

Zugehörige bewährte Methoden:

- REL02-BP03 Berücksichtigen von Erweiterung und Verfügbarkeit bei der Zuweisung von IP-Adressen für Subnetze
- REL02-BP05 Durchsetzen von sich nicht überschneidenden privaten IP-Adressbereichen in allen privaten Adressbereichen, in denen eine Verbindung besteht

Zugehörige Dokumente:

- Was ist ein Transit-Gateway?
- Bewährte Methoden für das Transit-Gateway-Design
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- Building a global network using AWS Transit Gateway Inter-Region peering
- Amazon Virtual Private Cloud Connectivity Options
- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Marketplace für Netzwerkinfrastruktur

Zugehörige Videos:

- AWS re:Invent 2.023 - AWS networking foundations
- AWS re:Invent 2.023 - Advanced VPC designs and new capabilities

Zugehörige Workshops:

- AWS Transit Gateway-Workshop

REL02-BP05 Erzwingen von sich nicht überschneidenden privaten IP-Adressbereichen in allen privaten Adressbereichen, in denen eine Verbindung besteht

Die IP-Adressbereiche Ihrer VPCs dürfen sich nicht überschneiden, wenn sie per Peering oder über Transit Gateway oder VPN verbunden sind. Vermeiden Sie IP-Adresskonflikte zwischen einer VPC und On-Premises-Umgebungen oder anderen verwendeten Cloud-Anbietern. Sie müssen bei Bedarf auch die Möglichkeit haben, private IP-Adressbereiche zuzuweisen. Ein IP-Adressenverwaltungssystem (IPAM) kann bei der Automatisierung helfen.

Gewünschtes Ergebnis:

- Keine Konflikte mit IP-Adressbereichen zwischen VPCs, On-Premises-Umgebungen oder anderen Cloud-Anbietern.
- Eine angemessene IP-Adressverwaltung ermöglicht eine einfachere Skalierung der Netzwerkinfrastruktur, um wachsenden und sich wandelnden Netzwerkanforderungen gerecht zu werden.

Typische Anti-Muster:

- Verwendung desselben IP-Bereichs in Ihrer VPC wie On-Premises, in Ihrem Unternehmensnetzwerk oder bei anderen Cloud-Anbietern.
- Keine Verfolgung von IP-Bereichen von VPCs, die zur Bereitstellung der Workloads verwendet werden.
- Alleinige Nutzung manueller IP-Adressverwaltungsprozesse wie Tabellenkalkulationen.
- Über- oder Unterdimensionierung von CIDR-Blöcken, was zu einer Verschwendung von IP-Adressen oder zu wenig Adressbereichen für Ihren Workload führt.

Vorteile der Nutzung dieser bewährten Methode: Mit der aktiven Planung des Netzwerks stellten Sie sicher, dass dieselbe IP-Adresse in miteinander verbunden Netzwerken nicht mehrmals vorkommt. So wird verhindert, dass Routing-Probleme in Teilen der Workload auftreten, die die verschiedenen Anwendungen verwenden.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Verwenden Sie ein IPAM, z. B. den Amazon VPC IP Address Manager, um Ihre CIDR-Nutzung zu überwachen und zu verwalten. Im AWS Marketplace stehen auch mehrere IPAMs zur Verfügung. Bewerten Sie die potenzielle Nutzung in AWS, fügen Sie vorhandenen VPCs CIDR-Bereiche hinzu und erstellen Sie neue VPCs, um das geplante Wachstum abzudecken.

Implementierungsschritte

- Ermitteln Sie den aktuellen CIDR-Umfang (z. B. VPCs und Subnetze).
  - Erfassen Sie über die Service API den aktuellen CIDR-Umfang.
  - Verwenden Sie den Amazon VPC IP Address Manager, um Ressourcen zu entdecken.
- Erfassen Sie die aktuelle Subnetzauslastung.
  - Ermitteln Sie über die Service-API die in jeder Region pro VPC vorhandenen Subnetze.
  - Verwenden Sie den Amazon VPC IP Address Manager, um Ressourcen zu entdecken.
- Zeichnen Sie die aktuelle Auslastung auf.
- Prüfen Sie, ob sich IP-Bereiche überschneiden.
- Berechnen Sie die freie Kapazität.
- Identifizieren Sie sich überschneidende IP-Bereiche. Sie können wahlweise zu einem neuen Adressbereich migrieren oder Techniken wie privates NAT-Gateway oder AWS PrivateLink verwenden, wenn Sie die sich überschneidenden Bereiche verbinden müssen.

Ressourcen

Zugehörige bewährte Methoden:

- Schutz von Netzwerken

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Marketplace für Netzwerkinfrastruktur
- Whitepaper Amazon Virtual Private Cloud Connectivity Options
- Multiple data center HA network connectivity
- Connecting Networks with Overlapping IP Ranges

- Was ist Amazon VPC?
- Was ist IPAM?

Zugehörige Videos:

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

Workload-Architektur

Ausgangspunkt für eine zuverlässige Workload sind vorab getroffene Designentscheidungen für Software und Infrastruktur. Ihre Auswahl in puncto Architektur wirkt sich in allen sechs Well-Architected-Säulen auf das Verhalten der Workload aus. Zur Gewährleistung von Zuverlässigkeit sind bestimmte Muster zu befolgen.

In den folgenden Abschnitten werden bewährte Methoden erläutert, die für diese Muster eingehalten werden sollten, damit eine hohe Zuverlässigkeit erzielt wird.

Themen

Entwerfen Ihrer Workload-Servicearchitektur

Erstellen Sie hoch skalierbare und zuverlässige Workloads mithilfe einer serviceorientierten Architektur (SOA) oder einer Microservices-Architektur. Eine serviceorientierte Architektur (SOA) hat zum Ziel, Softwarekomponenten über Service-Schnittstellen wiederverwendbar zu machen. Die Microservices-Architektur geht noch weiter, um Komponenten kleiner und einfacher zu machen.

Schnittstellen für serviceorientierte Architektur (SOA) verwenden gängige Kommunikationsstandards, sodass sie schnell in neue Workloads integriert werden können. SOA hat die Erstellung monolithischer Architekturen ersetzt, die aus voneinander abhängigen, unteilbaren Einheiten bestehen.

AWS hat schon immer SOA verwendet, aber jetzt setzen wir bei der Entwicklung unserer Systeme auf Microservices. Microservices bieten eine Vielzahl attraktiver Qualitäten. Der größte Nutzen für die Verfügbarkeit liegt jedoch darin, dass Microservices kleiner und einfacher sind. Mit Microservices können Sie die Verfügbarkeit verschiedener Services differenzieren und damit den Fokus von Investitionen auf die Microservices mit dem größten Verfügbarkeitsbedarf legen. Beispiel: Um Seiten mit Produktinformationen auf Amazon.com („Detailseiten“) bereitzustellen, werden Hunderte von Microservices aufgerufen, um einzelne Teile der Seite zu erstellen. Es gibt einige Services, die zur Verfügung stehen müssen, um Preis- und Produktdetails bereitzustellen, die große Mehrheit

der Inhalte auf der Seite können jedoch einfach ausgeschlossen werden, wenn der Service nicht verfügbar ist. Selbst Elemente wie Fotos und Rezensionen sind im Prinzip nicht erforderlich, um eine Umgebung zu schaffen, in der ein Kunde ein Produkt kaufen kann.

Bewährte Methoden

- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten
- REL03-BP02 Entwickeln von Services, die sich auf bestimmte Geschäftsdomänen und Funktionen konzentrieren
- REL03-BP03 Stellen Sie Serviceverträge bereit per API

REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten

Die Workload-Segmentierung ist wichtig, wenn es um die Festlegung der Resilienzanforderungen Ihrer Anwendung geht. Eine monolithische Architektur sollte vermieden werden, wann immer möglich. Stattdessen sollten Sie sorgfältig überlegen, welche Anwendungskomponenten in Microservices aufgeteilt werden können. Je nach Ihren Anwendungsanforderungen kann es sich dabei, wenn möglich, um eine Kombination aus einer serviceorientierten Architektur (SOA) und Microservices handeln. Workloads, die zustandslos sein können, können eher als Microservices bereitgestellt werden.

Gewünschtes Ergebnis: Workloads sollten unterstützbar, skalierbar und so lose miteinander verbunden sein wie möglich.

Wiegen Sie bei Entscheidungen zur Segmentierung von Workloads die Vorteile und die Komplexitäten miteinander ab. Was für ein neues Produkt richtig ist, das gerade auf dem Markt eingeführt wird, unterscheidet sich von den Anforderungen eines Workloads, der von Anfang an skalierbar sein muss. Bei einem Faktorwechsel für einen vorhandenen Monolith müssen Sie berücksichtigen, wie gut dieser aufgeteilt und in zustandslose Anwendungen transformiert werden kann. Die Aufteilung von Services in kleinere Teile ermöglicht kleinen, klar definierten Teams, diese weiterzuentwickeln und zu verwalten. Kleinere Services können jedoch Komplexitäten wie eine möglicherweise erhöhte Latenz, ein komplexeres Debugging und einen erhöhten operativen Aufwand einführen.

Typische Anti-Muster:

- Der Microservice Death Star ist eine Situation, in der die einzelnen Komponenten so stark voneinander abhängig werden, dass der Ausfall einer einzigen Komponente einen wesentlich

größeren Ausfall bewirkt. Das bedeutet, dass die Komponenten so starr und anfällig wie ein Monolith sind.

Vorteile der Nutzung dieser bewährten Methode:

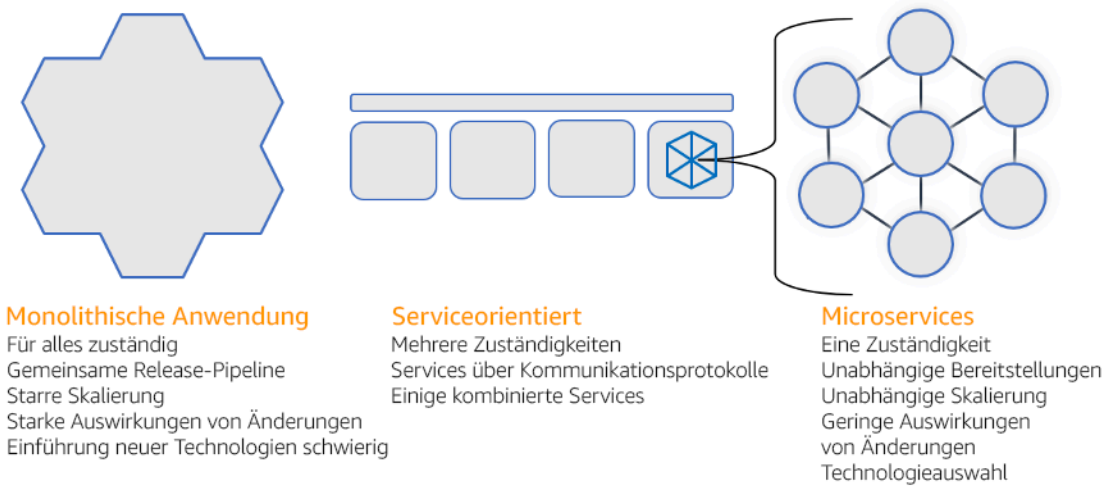
- Spezifischere Segmente führen zu einer größeren Agilität, zu organisatorischer Flexibilität und zu Skalierbarkeit.
- Die Auswirkungen von Service-Unterbrechungen werden reduziert.
- Die einzelnen Komponenten einer Anwendung besitzen möglicherweise unterschiedliche Anforderungen an die Verfügbarkeit, die von einer stärkeren Segmentierung besser unterstützt werden können.
- Die Verantwortlichkeiten der Teams, die den Workload unterstützen, sind klar definiert.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Wählen Sie Ihren Architekturtyp basierend auf der Segmentierung Ihres Workloads aus. Wählen Sie eine SOA oder eine Microservices-Architektur (oder in einigen seltenen Fällen eine monolithische Architektur). Selbst wenn Sie sich dafür entscheiden, mit einer Monolith-Architektur zu beginnen, müssen Sie sicherstellen, dass diese modular ist und sich letztendlich zu SOA Microservices weiterentwickeln kann, wenn Ihr Produkt mit der Benutzerakzeptanz skaliert. SOAund Microservices bieten jeweils eine geringere Segmentierung, was als moderne, skalierbare und zuverlässige Architektur bevorzugt wird. Allerdings müssen auch Kompromisse berücksichtigt werden, insbesondere bei der Implementierung einer Microservice-Architektur.

Aufgrund ihrer verteilten Computing-Architektur kann es schwieriger sein, die Latenzanforderungen von Benutzern zu erfüllen. Außerdem sind das Debugging und die Nachverfolgung von Benutzerinteraktionen komplexer. Zur Lösung dieses Problems können Sie AWS X-Ray verwenden. Ein weiterer Effekt ist die erhöhte operative Komplexität, da die Anzahl der von Ihnen verwalteten Anwendungen zunimmt. In der Folge müssen Sie eine größere Zahl voneinander unabhängiger Komponenten bereitstellen.



Monolithische, serviceorientierte und Microservice-Architekturen

Implementierungsschritte

- Ermitteln Sie die richtige Architektur für den Faktorwechsel oder die Entwicklung Ihrer Anwendung. SOAund Microservices bieten jeweils eine geringere Segmentierung, was als moderne, skalierbare und zuverlässige Architektur bevorzugt wird. SOAkann ein guter Kompromiss sein, um eine kleinere Segmentierung zu erreichen und gleichzeitig einige der Komplexitäten von Microservices zu vermeiden. Weitere Informationen finden Sie unter Microservice Trade-Offs.
- Wenn Ihre Workload für sie zugänglich ist und Ihre Organisation sie unterstützen kann, sollten Sie eine Microservices-Architektur verwenden, um die beste Agilität und Zuverlässigkeit zu erzielen. Weitere Informationen finden Sie unter Implementierung von Microservices auf. AWS
- Sie sollten das Muster mit der Bezeichnung Strangler Fig („Würgefeige“) verwenden, um einen Faktorwechsel für einen Monolithen durchzuführen, bei dem Sie diesen in kleinere Komponenten aufteilen. Dies umfasst die schrittweise Ersetzung spezifischer Anwendungskomponenten durch neue Anwendungen und Services. AWS Migration Hub Refactor Spaces dient als Ausgangspunkt für den inkrementellen Faktorwechsel. Weitere Informationen finden Sie unter Seamlessly migrate on-premises legacy workloads using a strangler pattern.
- Für die Implementierung von Microservices ist möglicherweise ein Service Discovery-Mechanismus erforderlich, der es diesen verteilten Diensten ermöglicht, miteinander zu kommunizieren. AWS App Meshkann mit serviceorientierten Architekturen verwendet werden, um eine zuverlässige Erkennung und den zuverlässigen Zugriff auf Dienste zu ermöglichen. AWS Cloud Mapkann auch für die dynamische, DNS basierte Serviceerkennung verwendet werden.

- Wenn Sie von einem Monolith zu Amazon MQ migrierenSOA, kann Amazon MQ Ihnen helfen, die Lücke als Service Bus bei der Neugestaltung älterer Anwendungen in der Cloud zu schließen.
- Im Fall vorhandener Monolithen mit einer einzigen, geteilten Datenbank müssen Sie entscheiden, wie Sie die Daten neu in kleineren Segmenten organisieren. Dabei kann es sich um Geschäftsbereiche, Zugriffsmuster oder Datenstrukturen handeln. An diesem Punkt des Refactoring-Prozesses sollten Sie sich dafür entscheiden, mit einem relationalen oder einem nicht-relationalen Datenbanktyp (Nein) fortzufahren. SQL Weitere Informationen finden Sie unter Von bis Nein. SQL SQL

Aufwand für den Implementierungsplan: Hoch

Ressourcen

Zugehörige bewährte Methoden:

- REL03-BP02 Entwickeln von Services, die sich auf bestimmte Geschäftsdomänen und Funktionen konzentrieren

Zugehörige Dokumente:

- Amazon API Gateway: Konfiguration eines REST API mit Open API
- Was ist eine serviceorientierte Architektur?
- Bounded Context (a central pattern in Domain-Driven Design)
- Implementierung von Microservices auf AWS
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- Microservices auf AWS
- Was ist AWS App Mesh?

Zugehörige Beispiele:

- Workshop für die iterative App-Modernisierung

Zugehörige Videos:

- Exzellenz mit Microservices erreichen AWS

REL03-BP02 Entwickeln von Services, die sich auf bestimmte Geschäftsdomänen und Funktionen konzentrieren

Eine serviceorientierte Architektur (SOA) definiert Services mit genau abgegrenzten Funktionen, die von Geschäftsanforderungen definiert werden. Microservices verwenden Domain-Modelle und begrenzten Kontext, um Servicegrenzen entlang der Grenzen des Geschäftskontextes zu ziehen. Die Konzentration auf Geschäftsdomänen und Funktionen hilft Teams dabei, unabhängige Zuverlässigkeitsanforderungen für ihre Services zu definieren. Begrenzte Kontexte isolieren und kapseln die Geschäftslogik, sodass Teams besser überlegen können, wie mit Fehlern umzugehen ist.

Gewünschtes Ergebnis: Ingenieure und geschäftliche Interessenvertreter definieren gemeinsam begrenzte Kontexte und verwenden sie, um Systeme als Services zu entwerfen, die bestimmte Geschäftsfunktionen erfüllen. Diese Teams verwenden etablierte Praktiken wie Event Storming, um Anforderungen zu definieren. Neue Anwendungen sind als Services mit klar definierten Grenzen und losen Verkopplungen definiert. Bestehende Monolithen werden in begrenzte Kontexte zerlegt und Systemdesigns bewegen sich in Richtung SOA- oder Microservice-Architekturen Bei der Refaktorisierung von Monolithen kommen etablierte Ansätze wie Bubble-Kontexte und Monolith-Zerlegung zur Anwendung.

Domain-orientierte Services werden als ein oder mehrere Prozesse ausgeführt, die keinen gemeinsamen Zustand haben. Sie reagieren selbstständig auf Nachfrageschwankungen und behandeln Störszenarien anhand Domain-spezifischer Anforderungen.

Typische Anti-Muster:

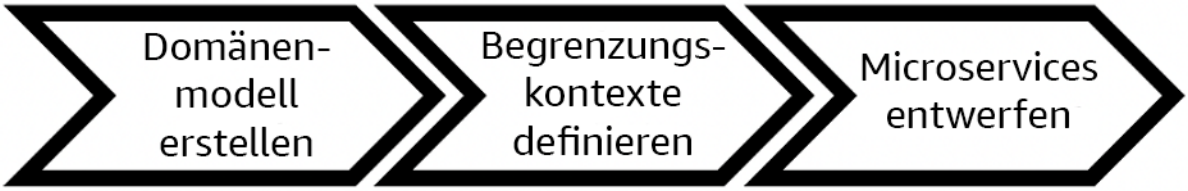
- Teams werden für bestimmte technische Bereiche wie UI und UX, Middleware oder Datenbank gebildet, anstatt für bestimmte Geschäftsdomänen.
- Anwendungen erstrecken sich über die Zuständigkeiten der einzelnen Domains. Services, die sich über begrenzte Kontexte erstrecken, können schwieriger zu verwalten sein, erfordern einen größeren Testaufwand und erfordern die Teilnahme mehrerer Domain-Teams an Softwareupdates.
- Domänenabhängigkeiten wie Domain-Entity-Bibliotheken werden von allen Services gemeinsam genutzt, sodass Änderungen für eine Servicedomäne Änderungen an anderen Service-Domänen erfordern.
- Serviceverträge und Geschäftslogik formulieren Entitäten nicht in einer gemeinsamen und konsistenten Domain-Sprache, was zu Übersetzungsebenen führt, die Systeme komplizieren und den Debugging-Aufwand erhöhen.

Vorteile der Nutzung dieser bewährten Methode: Anwendungen sind als unabhängige Services konzipiert, die durch Geschäftsdomänen begrenzt sind und eine gemeinsame Geschäftssprache verwenden. Services sind unabhängig voneinander testbar und einsetzbar. Services erfüllen die Domain-spezifischen Resilienzanforderungen für die implementierte Domain.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Domain-driven Design (DDD, domänengesteuertes Design) ist der grundlegende Ansatz für das Entwerfen und Entwickeln von Software rund um Geschäftsdomänen. Bei der Entwicklung von Services, die sich auf Geschäftsdomänen konzentrieren, ist es hilfreich, mit einem vorhandenen Framework zu arbeiten. Wenn Sie mit bestehenden monolithischen Anwendungen arbeiten, können Sie die Vorteile von Zerlegungsmustern nutzen, die etablierte Techniken zur Modernisierung von Anwendungen in Services bereitstellen.



Domain-gesteuertes Design

Implementierungsschritte

- Teams können Event-Storming-Workshops veranstalten, um rasch Ereignisse, Befehle, Mengen und Domänen in einem unkomplizierten Notizformat zu sammeln.
- Sobald Domain-Entitäten und -Funktionen in einem Domain-Kontext gebildet wurden, können Sie Ihre Domain mithilfe eines begrenzten Kontexts, weiter in kleinere Modelle unterteilt, wobei Entitäten mit ähnlichen Funktionen und Attributen in Gruppen sortiert werden. Wenn das Modell in Kontexte unterteilt ist, entsteht eine Vorlage für die Begrenzung von Microservices.
  - Für die Website Amazon.com können Entitäten beispielsweise Pakete, Zustellung, Zeitplan, Preise, Rabatte und Währung enthalten.
  - Paket, Zustellung und Zeitplan werden dem Versandkontext zugeordnet, während Preis, Rabatt und Währung dem Preiskontext zugeordnet sind.

- Unter Decomposing monoliths into microservices wird das Muster für das Refactoring von Microservices skizziert. Die Verwendung von Mustern für die Unterteilung nach Geschäftsfähigkeit, Subdomäne oder Transaktion passt gut zu Domain-gesteuerten Ansätzen.
- Taktische Techniken wie der Bubble-Kontext ermöglichen es Ihnen, DDD in bestehenden oder älteren Anwendungen einzuführen, ohne dass Sie im Voraus Änderungen vornehmen und sich voll und ganz auf DDD verlassen müssen. Bei einem Bubble-Kontext-Ansatz wird mithilfe von Service-Mapping und -koordination ein kleiner begrenzter Kontext oder eine Ebene zur Korruptionsbekämpfung erstellt, die das neu definierte Domain-Modell vor äußeren Einflüssen schützt.

Nachdem die Teams eine Domain-Analyse durchgeführt und Entitäten und Serviceverträge definiert haben, können sie AWS-Services nutzen, um ihr Domain-gesteuertes Design als Cloud-basierte Services zu implementieren.

- Beginnen Sie Ihre Entwicklung, indem Sie Tests definieren, die die Geschäftsregeln Ihrer Domain anwenden. Test-driven Development (TDD, Testgetriebene Entwicklung) und Behavior-driven Development (BDD, verhaltensgetriebene Entwicklung) helfen Teams dabei, die Services auf die Lösung von Geschäftsproblemen zu konzentrieren.
- Wählen Sie AWS-Services die den Anforderungen Ihrer Geschäfts-Domain und Ihrer Microservice-Architektur am besten entsprechen:
  - AWS Serverless ermöglicht es Ihrem Team, sich auf eine bestimmte Domain-Logik zu konzentrieren, anstatt Server und Infrastruktur zu verwalten.
  - Container in AWS vereinfachen die Verwaltung Ihrer Infrastruktur, sodass Sie sich auf Ihre Domain-Anforderungen konzentrieren können.
  - Speziell entwickelte Datenbanken helfen Ihnen dabei, Ihre Domain-Anforderungen dem am besten geeigneten Datenbanktyp zuzuordnen.
- Hexagonale Architekturen auf AWS skizzieren ein Framework zur Integration von Geschäftslogik in Services. Dabei wird rückwärts von der Geschäfts-Domain aus gearbeitet, um funktionale Anforderungen zu erfüllen und dann Integrationsadapter zu implementieren. Muster, die Schnittstellendetails von der Geschäftslogik mit AWS-Services trennen, helfen Teams, sich auf die Funktionalität der Domain zu konzentrieren und die Softwarequalität zu verbessern.

Ressourcen

Zugehörige bewährte Methoden:

- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten
- REL03-BP03 Stellen Sie Serviceverträge bereit per API

Zugehörige Dokumente:

- AWS Microservices
- Implementieren von Microservices in AWS
- How to break a Monolith into Microservices
- Getting Started with DDD when Surrounded by Legacy Systems
- Domain-Driven Design: Tackling Complexity in the Heart of Software
- Building hexagonal architectures on AWS
- Decomposing monoliths into microservices
- Event Storming
- Messages Between Bounded Contexts
- Microservices
- Testgetriebene Entwicklung
- Verhaltensgetriebene Entwicklung

Zugehörige Beispiele:

- Designing Cloud Native Microservices on AWS (from DDD/EventStormingWorkshop)

Zugehörige Tools:

- AWS Cloud-Datenbanken
- Serverless in AWS
- Container in AWS

REL03-BP03 Stellen Sie Serviceverträge bereit per API

Dienstleistungsverträge sind dokumentierte Vereinbarungen zwischen API Herstellern und Verbrauchern, die in einer maschinenlesbaren API Definition definiert sind. Eine Strategie zur Versionierung von Verträgen ermöglicht es den Verbrauchern, die vorhandenen weiterhin zu

verwenden API und ihre Anwendungen auf eine neuere zu migrieren, API sobald sie bereit sind. Die Bereitstellung durch den Produzenten kann jederzeit erfolgen, solange der Vertrag eingehalten wird. Serviceteams können den Technologie-Stack ihrer Wahl verwenden, um den API Vertrag zu erfüllen.

Gewünschtes Ergebnis: Anwendungen, die mit serviceorientierten Architekturen oder Microservice-Architekturen erstellt wurden, können unabhängig voneinander betrieben werden und verfügen gleichzeitig über eine integrierte Laufzeitabhängigkeit. Änderungen, die an einem API Verbraucher oder Hersteller vorgenommen werden, beeinträchtigen nicht die Stabilität des Gesamtsystems, wenn beide Seiten einen gemeinsamen Vertrag einhalten. API Komponenten, die über den Service kommunizieren, APIs können unabhängige funktionale Releases durchführen, Upgrades auf Laufzeitabhängigkeiten durchführen oder ein Failover zu einem Disaster Recovery (DR) -Standort durchführen, ohne sich gegenseitig zu beeinträchtigen. Darüber hinaus können spezialisierte Services unabhängig voneinander skaliert werden und können dabei den Ressourcenbedarf absorbieren, ohne dass andere Services ebenfalls skaliert werden müssen.

Typische Anti-Muster:

- Ein Dienst wird APIs ohne stark typisierte Schemas erstellt. Dies führt dazuAPIs, dass sie nicht zum Generieren von API Bindungen und Nutzlasten verwendet werden können, die nicht programmgesteuert validiert werden können.
- Es wird keine Versionierungsstrategie angewendet, die API Verbraucher dazu zwingt, Updates und Releases vorzunehmen, andernfalls scheitern sie, wenn sich die Serviceverträge weiterentwickeln.
- Fehlermeldungen, die Details der zugrundeliegenden Service-Implementierung preisgeben, anstatt Integrationsfehler im Kontext und in der Sprache der Domain zu beschreiben.
- Keine Nutzung von API Verträgen zur Entwicklung von Testfällen und API Scheinimplementierungen, um unabhängige Tests von Servicekomponenten zu ermöglichen.

Vorteile der Einführung dieser bewährten Methode: Verteilte Systeme, die aus Komponenten bestehen, die über API Serviceverträge miteinander kommunizieren, können die Zuverlässigkeit verbessern. Entwickler können potenzielle Probleme frühzeitig im Entwicklungsprozess catch, indem sie während der Kompilierung eine Typprüfung durchführen, um sicherzustellen, dass Anfragen und Antworten dem API Vertrag entsprechen und die erforderlichen Felder vorhanden sind. APIVerträge bieten eine klare, sich selbst dokumentierende Schnittstelle für eine bessere Interoperabilität zwischen verschiedenen Systemen APIs und Programmiersprachen und sorgen für eine bessere Interoperabilität.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

Sobald Sie Geschäftsbereiche identifiziert und Ihre Workload-Segmentierung festgelegt haben, können Sie Ihren Service weiterentwickeln. APIs Definieren Sie zunächst maschinenlesbare Serviceverträge für APIs und implementieren Sie anschließend eine API Versionierungsstrategie. Wenn Sie bereit sindREST, Dienste über gängige Protokolle wie GraphQL oder asynchrone Ereignisse zu integrieren, können Sie AWS Dienste in Ihre Architektur integrieren, um Ihre Komponenten mit stark API typisierten Verträgen zu integrieren.

AWS Dienste für Serviceverträge API

Integrieren Sie AWS Services wie Amazon API Gateway und Amazon EventBridge in Ihre Architektur AWS AppSync, um API Serviceverträge in Ihrer Anwendung zu verwenden. Amazon API Gateway unterstützt Sie bei der direkten Integration mit nativen AWS Diensten und anderen Webdiensten. APIGateway unterstützt die APIOpen-Spezifikation und Versionierung. AWS AppSync ist ein verwalteter GraphQL-Endpunkt, den Sie konfigurieren, indem Sie ein GraphQL-Schema definieren, um eine Serviceschnittstelle für Abfragen, Mutationen und Abonnements zu definieren. Amazon EventBridge verwendet Ereignisschemas, um Ereignisse zu definieren und Codebindungen für Ihre Ereignisse zu generieren.

Implementierungsschritte

- Definieren Sie zunächst einen Vertrag für Ihren. API Ein Vertrag drückt die Fähigkeiten eines aus und API definiert stark typisierte Datenobjekte und Felder für die API Eingabe und Ausgabe.
- Bei der Konfiguration APIs in API Gateway können Sie Open API Specifications für Ihre Endgeräte importieren und exportieren.
  - Der Import einer offenen API Definition vereinfacht die Erstellung Ihrer Definition API und kann als Codetools wie AWS Serverless Application Modelund AWS Cloud Development Kit (AWS CDK)in die AWS Infrastruktur integriert werden.
  - Das Exportieren einer API Definition vereinfacht die Integration mit API Testtools und bietet dem Servicenutzer eine Integrationsspezifikation.
- Sie können APIs GraphQL definieren und verwalten, AWS AppSync indem Sie eine GraphQL-Schemadatei definieren, um Ihre Vertragsschnittstelle zu generieren und die Interaktion mit komplexen REST Modellen, mehreren Datenbanktabellen oder älteren Diensten zu vereinfachen.
- AWS AmplifyProjekte, die integriert sind, AWS AppSync generieren stark typisierte JavaScript Abfragedateien zur Verwendung in Ihrer Anwendung sowie eine AWS AppSync GraphQL-Clientbibliothek für Amazon DynamoDB-Tabellen.

- Wenn Sie Serviceereignisse von Amazon nutzen EventBridge, entsprechen Ereignisse Schemas, die bereits in der Schemaregistrierung vorhanden sind oder die Sie mit der Open API Spec definieren. Mit einem in der Registrierung definierten Schema können Sie auch Client-Bindungen aus dem Schemavertrag generieren, um Ihren Code in Ereignisse zu integrieren.
- Erweiterung oder Versionierung Ihrer. API Das Erweitern von API ist eine einfachere Option, wenn Felder hinzugefügt werden, die mit optionalen Feldern oder Standardwerten für Pflichtfelder konfiguriert werden können.
  - JSONbasierte Verträge für Protokolle wie REST GraphQL können sich gut für eine Vertragsverlängerung eignen.
  - XMLVerträge, die auf Protokollen basieren, SOAP sollten mit den Nutzern der Dienste getestet werden, um festzustellen, ob eine Vertragsverlängerung durchführbar ist.
- Bei der Versionierung einer sollten Sie die Implementierung einer Proxy-Versionierung in Betracht ziehenAPI, bei der eine Fassade zur Unterstützung von Versionen verwendet wird, sodass die Logik in einer einzigen Codebasis verwaltet werden kann.
  - Mit API Gateway können Sie Anfragen- und Antwortzuordnungen verwenden, um die Übernahme von Vertragsänderungen zu vereinfachen, indem Sie eine Fassade einrichten, um Standardwerte für neue Felder bereitzustellen oder entfernte Felder aus einer Anfrage oder Antwort zu entfernen. Mit diesem Ansatz kann der zugrunde liegende Service mit einer einzelnen Codebasis betrieben werden.

Ressourcen

Zugehörige bewährte Methoden:

- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten
- REL03-BP02 Entwickeln von Services, die sich auf bestimmte Geschäftsdomänen und Funktionen konzentrieren
- REL04-BP02 Lose gekoppelte Abhängigkeiten implementieren
- REL05-BP03 Steuerung und Begrenzung von Wiederholungsaufrufen
- REL05-BP05 Client-Timeouts festlegen

Zugehörige Dokumente:

- Was ist eine API (Anwendungsprogrammierschnittstelle)?
- Implementierung von Microservices auf AWS

- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- Microservices auf AWS
- Ich arbeite mit API Gateway-Erweiterungen für Open API
- APIOpen-Spezifikation
- GraphQL: Schemata und Typen
- EventBridge Amazon-Codebindungen

Zugehörige Beispiele:

- Amazon API Gateway: Konfiguration eines REST API mit Open API
- CRUDAnwendung Amazon API Gateway to Amazon DynamoDB mithilfe von Open API
- Moderne Anwendungsintegrationsmuster in einem serverlosen Zeitalter: API Gateway Service Integration
- Implementierung einer Header-basierten API Gateway-Versionierung mit Amazon CloudFront
- AWS AppSync: Building a client application

Zugehörige Videos:

- Verwenden von Open API in AWS SAM zur Verwaltung von Gateway API

Zugehörige Tools:

- APIAmazon-Gateway
- AWS AppSync
- Amazon EventBridge

Entwerfen von Interaktionen in einem verteilten System, um Ausfälle zu vermeiden

Verteilte Systeme sind auf Kommunikationsnetzwerke angewiesen, um Komponenten wie Server oder Services miteinander zu verbinden. Ihre Workload muss trotz Datenverlust oder Latenz in diesen Netzwerken zuverlässig funktionieren. Die Komponenten des verteilten Systems müssen so

funktionieren, dass sie sich nicht negativ auf andere Komponenten oder die Workload auswirken. Diese bewährten Methoden verhindern Ausfälle und verbessern die mittlere Betriebsdauer zwischen Ausfällen (MTBF).

Bewährte Methoden

- REL04-BP01 Bestimmen Sie, von welcher Art von verteilten Systemen Sie abhängig sind
- REL04-BP02 Lose gekoppelte Abhängigkeiten implementieren
- REL04-BP03 Arbeite ständig
- REL04-BP04 Festlegen von Mutationsoperationen als idempotent

REL04-BP01 Bestimmen Sie, von welcher Art von verteilten Systemen Sie abhängig sind

Verteilte Systeme verwenden die synchrone, asynchrone oder Stapelverarbeitung. Synchrone Systeme müssen Anfragen so schnell wie möglich verarbeiten und miteinander kommunizieren, indem sie synchrone Anfrage- und Antwortaufrufe mithilfe von HTTP/S-, REST- oder RPC-Protokollen (Remote Procedure Call) durchführen. Asynchrone Systeme kommunizieren miteinander, indem sie Daten asynchron über einen Zwischenservice austauschen, ohne einzelne Systeme zu koppeln. Systeme mit Stapelverarbeitung empfangen eine große Menge an Eingabedaten, führen automatisierte Datenprozesse ohne menschliches Eingreifen aus und generieren Ausgabedaten.

Gewünschtes Ergebnis: Entwerfen Sie einen Workload, der effektiv mit synchronen, asynchronen und Batch-Abhängigkeiten interagiert.

Typische Anti-Muster:

- Der Workload wartet auf unbestimmte Zeit auf eine Antwort von seinen Abhängigkeiten, was dazu führen kann, dass Workload-Clients das Zeitlimit überschreiten und nicht wissen, ob ihre Anfrage eingegangen ist.
- Der Workload verwendet eine Kette von abhängigen Systemen, die sich gegenseitig synchron aufrufen. Der Erfolg der gesamten Kette hängt davon ab, dass jedes System verfügbar ist und Anfragen erfolgreich verarbeitet, was zu instabilem Verhalten und eingeschränkter Gesamtverfügbarkeit führen kann.
- Der Workload kommuniziert asynchron mit seinen Abhängigkeiten und stützt sich auf das Konzept der garantierten einmaligen Zustellung von Nachrichten, obwohl es oft immer noch möglich ist, doppelte Nachrichten zu empfangen.

- Der Workload verwendet keine geeigneten Tools zur Batchplanung und ermöglicht die gleichzeitige Ausführung desselben Batchjobs.

Vorteile der Nutzung dieser bewährten Methode: Es ist üblich, dass ein bestimmter Workload einen oder mehrere Kommunikationsstile der synchronen, asynchronen und Stapelverarbeitung implementiert. Diese bewährte Methode hilft Ihnen dabei, die verschiedenen Kompromisse zu identifizieren, die mit den einzelnen Kommunikationsstilen verbunden sind, damit Ihr Workload Störungen in allen Abhängigkeiten tolerieren kann.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Die folgenden Abschnitte enthalten sowohl allgemeine als auch spezifische Implementierungshinweise für jede Art von Abhängigkeit.

General guidance (Allgemeine Anleitung

- Stellen Sie sicher, dass die Leistungs- und Zuverlässigkeits-Servicelevel-Ziele (SLOs), die Ihre Abhängigkeiten bieten, den Leistungs- und Zuverlässigkeitsanforderungen Ihres Workloads entsprechen.
- Verwenden Sie AWS Observability Services, um Reaktionszeiten und Fehlerraten zu überwachen, und so sicherzustellen, dass Ihre Abhängigkeit den von Ihrem Workload benötigten Service bietet.
- Identifizieren Sie die potenziellen Herausforderungen, mit denen Ihr Workload bei der Kommunikation mit seinen Abhängigkeiten konfrontiert sein könnte. Verteilte Systeme sind mit einer Vielzahl von Herausforderungen verbunden, die die architektonische Komplexität, den Betriebsaufwand und die Kosten erhöhen können. Zu den häufigsten Herausforderungen gehören Latenz, Netzwerkunterbrechungen, Datenverlust, Skalierung und Verzögerungen bei der Datenreplikation.
- Implementieren Sie eine robuste Fehlerbehandlung und -protokollierung, um Probleme zu beheben, wenn es in Ihrer Abhängigkeit zu Problemen kommt.

Synchrone Abhängigkeit

Bei synchroner Kommunikation sendet Ihr Workload eine Anfrage an seine Abhängigkeit und blockiert den Vorgang, der auf eine Antwort wartet. Wenn ihre Abhängigkeit die Anfrage erhält, versucht sie, sie so schnell wie möglich zu bearbeiten, und sendet eine Antwort zurück an den Workload. Eine große Herausforderung bei synchroner Kommunikation besteht darin, dass sie

zu einer zeitlichen Kopplung führt, was erfordert, dass der Workload und dessen Abhängigkeiten gleichzeitig verfügbar sind. Beachten Sie die folgenden Hinweise, wenn der Workload synchron mit seinen Abhängigkeiten kommunizieren muss:

- Der Workload sollte sich nicht auf mehrere synchrone Abhängigkeiten stützen, um eine einzelne Funktion auszuführen. Diese Kette von Abhängigkeiten erhöht die allgemeine Instabilität, da alle Abhängigkeiten im Pfad verfügbar sein müssen, damit die Anfrage erfolgreich abgeschlossen werden kann.
- Wenn eine Abhängigkeit fehlerhaft oder nicht verfügbar ist, bestimmen Sie Ihre Strategie zur Fehlerbehandlung und versuchen Sie es erneut. Vermeiden Sie bimodales Verhalten. Bimodales Verhalten liegt vor, wenn sich der Workload im Normalmodus und im Fehlermodus unterschiedlich verhält. Weitere Informationen zu bimodalem Verhalten finden Sie unter REL11-BP05 Verhindern von bimodalem Verhalten mithilfe statischer Stabilität.
- Denken Sie daran, dass es besser ist, schnell zu scheitern, als Ihren Workload warten zu lassen. Im AWS Lambda Entwicklerleitfaden wird beispielsweise beschrieben, wie Wiederholungen und Fehlschläge beim Aufrufen von Lambda-Funktionen behandelt werden.
- Legen Sie Timeouts fest, wenn Ihr Workload seine Abhängigkeit aufruft. Dadurch wird vermieden, zu lange oder unbegrenzt auf eine Antwort zu warten. Eine hilfreiche Diskussion zu diesem Problem finden Sie unter Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications.
- Reduzieren Sie die Anzahl der Aufrufe von Ihrem Workload an seine Abhängigkeit, um eine einzelne Anfrage zu erfüllen. Durch zahlreiche Aufrufe erhöht sich die Kopplung und Latenz.

Asynchrone Abhängigkeit

Um den Workload zeitlich von dessen Abhängigkeiten zu entkoppeln, sollte die Kommunikation asynchron erfolgen. Bei einem asynchronen Ansatz kann der Workload mit jeder anderen Verarbeitung fortfahren, ohne auf eine Antwort der Abhängigkeit oder Kette von Abhängigkeiten warten zu müssen.

Beachten Sie die folgenden Hinweise, wenn der Workload asynchron mit seiner Abhängigkeit kommunizieren muss:

- Entscheiden Sie je nach Anwendungsfall und Anforderungen, ob Sie Messaging oder Ereignis-Streaming verwenden möchten. Beim Messaging kann der Workload mit seiner Abhängigkeit kommunizieren, indem er Nachrichten über einen Message Broker sendet und empfängt. Beim Ereignis-Streaming können der Workload und seine Abhängigkeiten einen Streaming-

Dienst verwenden, um Ereignisse zu veröffentlichen und zu abonnieren, die als kontinuierliche Datenströme bereitgestellt werden und so schnell wie möglich verarbeitet werden müssen.

- Messaging und Ereignis-Streaming behandeln Nachrichten unterschiedlich, sodass Sie basierend auf den folgenden Faktoren Abwägungsentscheidungen treffen müssen:
  - Nachrichtenpriorität: Message Broker können Nachrichten mit hoher Priorität vor normalen Nachrichten verarbeiten. Beim Ereignis-Streaming haben alle Nachrichten dieselbe Priorität.
  - Nachrichtenverbrauch: Message Broker stellen sicher, dass die Verbraucher die Nachricht erhalten. Ereignis-Streaming-Verbraucher müssen den Überblick über die zuletzt gelesene Nachricht behalten.
  - Nachrichtenreihenfolge: Beim Messaging ist der Empfang von Nachrichten in der genauen Reihenfolge, in der sie gesendet wurden, nicht garantiert, es sei denn, Sie verwenden einen FIFO-Ansatz (First-in-First-Out). Beim Ereignis-Streaming wird immer die Reihenfolge beibehalten, in der die Daten erzeugt wurden.
  - Löschen von Nachrichten: Beim Messaging muss der Verbraucher die Nachricht nach der Verarbeitung löschen. Der Ereignis-Streaming-Dienst hängt die Nachricht an einen Stream an und verbleibt dort, bis die Aufbewahrungsfrist der Nachricht abläuft. Diese Löschrichtlinie macht das Ereignis-Streaming für die Wiedergabe von Nachrichten geeignet.
- Definieren Sie, wie Ihr Workload weiß, wann seine Abhängigkeit seine Arbeit beendet hat. Wenn der Workload beispielsweise asynchron eine Lambda-Funktion aufruft, stellt Lambda das Ereignis in eine Warteschlange und gibt eine Erfolgsantwort ohne zusätzliche Informationen zurück. Nach Abschluss der Verarbeitung kann die Lambda-Funktion das Ergebnis an ein Ziel senden, das je nach Erfolg oder Misserfolg konfiguriert werden kann.
- Entwickeln Sie Ihren Workload so, dass er doppelte Nachrichten verarbeiten kann, indem Sie Idempotenz nutzen. Idempotenz bedeutet, dass sich die Ergebnisse des Workloads nicht ändern, auch wenn der Workload mehrmals für dieselbe Nachricht generiert wird. Es ist wichtig, darauf hinzuweisen, dass Messaging- oder Streaming-Dienste eine Nachricht erneut übermitteln, wenn ein Netzwerkausfall auftritt oder wenn keine Bestätigung eingegangen ist.
- Wenn Ihr Workload keine Antwort von seiner Abhängigkeit erhält, muss er die Anfrage erneut einreichen. Erwägen Sie, die Anzahl der Wiederholungsversuche zu begrenzen, um die CPU-, Arbeitsspeicher- und Netzwerkressourcen des Workloads für die Bearbeitung anderer Anfragen zu schonen. In der AWS Lambda-Dokumentation wird gezeigt, wie Fehler beim asynchronen Aufruf behandelt werden.
- Nutzen Sie geeignete Beobachtbarkeits-, Debugging- und Tracing-Tools, um die asynchrone Kommunikation des Workloads mit seinen Abhängigkeiten zu verwalten und zu betreiben. Sie

können Amazon CloudWatch verwenden, um Nachrichten- und Ereignis-Streaming-Dienste zu überwachen. Sie können auch Ihren Workload mit AWS X-Ray instrumentieren, um schnell Erkenntnisse zur Problembehebung zu gewinnen.

Batch-Abhängigkeit

Batch-Systeme nehmen Eingabedaten auf, initiieren eine Reihe von Aufgaben, um sie zu verarbeiten, und erzeugen einige Ausgabedaten, ohne dass manuelles Eingreifen erforderlich ist. Je nach Datengröße können Aufgaben in wenigen Minuten oder in einigen Fällen sogar in mehreren Tagen ausgeführt werden. Beachten Sie die folgenden Hinweise, wenn der Workload mit seiner Batch-Abhängigkeit kommuniziert:

- Definieren Sie das Zeitfenster, in dem der Workload den Batchjob ausführen soll. Der Workload kann ein Wiederholungsmuster einrichten, um ein Batchsystem aufzurufen, beispielsweise jede Stunde oder am Ende eines jeden Monats.
- Ermitteln Sie den Ort der Dateneingabe und der verarbeiteten Datenausgabe. Wählen Sie einen Speicherservice wie Amazon Simple Storage Services (Amazon S3), Amazon Elastic File System (Amazon EFS) und Amazon FSx for Lustre, der es Ihrem Workload ermöglicht, Dateien in großem Umfang zu lesen und zu schreiben.
- Wenn Ihr Workload mehrere Batchjobs aufrufen muss, können Sie AWS Step Functions nutzen, um die Orchestrierung von Batchjobs, die in AWS oder On-Premises ausgeführt werden, zu vereinfachen. in diesem Beispielprojekt wird die Orchestrierung von Batchjobs mithilfe von Step Functions, AWS Batch und Lambda demonstriert.
- Überwachen Sie Batchjobs, um nach Auffälligkeiten zu suchen, z. B. wenn die Ausführung eines Jobs länger dauert, als sie sollte. Sie könnten Tools wie CloudWatch Container Insights verwenden, um AWS Batch-Umgebungen und Jobs zu überwachen. In diesem Fall würde Ihr Workload den Beginn des nächsten Jobs unterbrechen und die zuständigen Mitarbeiter über die Ausnahme informieren.

Ressourcen

Zugehörige Dokumente:

- AWS Cloud-Operationen: Überwachung und Beobachtbarkeit
- Die Amazon Builders' Library: Herausforderungen für verteilte Systeme
- REL11-BP05 Verhindern von bimodalem Verhalten mithilfe statischer Stabilität

- AWS Lambda-Entwicklerhandbuch: Fehlerbehandlung und automatische Wiederholungsversuche in AWS Lambda
- Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications
- AWS-Messaging
- Was sind Streaming-Daten?
- AWS Lambda-Entwicklerhandbuch: Asynchroner Aufruf
- Amazon Simple Queue Service FAQ: FIFO-Warteschlangen
- Amazon Kinesis Data Streams-Entwicklerhandbuch: Umgang mit doppelten Datensätzen
- Amazon Simple Queue Service-Entwicklerhandbuch: Verfügbare CloudWatch-Metriken für Amazon SQS
- Amazon Kinesis Data Streams-Entwicklerhandbuch: Überwachung des Amazon Kinesis Data Streams-Service mit Amazon CloudWatch
- AWS X-Ray-Entwicklerhandbuch: AWS X-Ray-Konzepte
- AWS Beispiele auf GitHub: AWS Step functions Complex Orchestrator App
- AWS Batch-Benutzerhandbuch: AWS Batch CloudWatch Container Insights

Zugehörige Videos:

- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS (COP310)

Zugehörige Tools:

- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS X-Ray
- Amazon Simple Storage Service (Amazon S3)
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx für Lustre
- AWS Step Functions
- AWS Batch

REL04-BP02 Lose gekoppelte Abhängigkeiten implementieren

Abhängigkeiten etwa zwischen Warteschlangensystemen, Streaming-Systemen, Workflows und Load Balancern sind lose gekoppelt. Eine lose Verkoppelung hilft, das Verhalten einer Komponente von anderen Komponenten zu isolieren, die von ihr abhängig sind. Dies verbessert Resilienz und Agilität.

Die Entkopplung von Abhängigkeiten wie Warteschlangensystemen, Streaming-Systemen und Workflows trägt dazu bei, die Auswirkungen von Änderungen oder Ausfällen auf ein System zu minimieren. Durch diese Trennung wird das Verhalten einer Komponente von Auswirkungen auf andere, die von ihr abhängig sind, isoliert und so die Widerstandsfähigkeit und Agilität verbessert.

In eng gekoppelten Systemen können Änderungen an einer Komponente Änderungen an anderen Komponenten erforderlich machen, die von ihr abhängen, was die Leistung aller Komponenten beeinträchtigt. Die lose Verkopplung unterbricht diese Abhängigkeit, sodass abhängige Komponenten nur die versionierte und veröffentlichte Schnittstelle kennen müssen. Die Implementierung einer losen Verkopplung zwischen Abhängigkeiten isoliert einen Ausfall. So wird verhindert, dass er sich auf andere Komponenten auswirkt.

Die lose Verkoppelung ermöglicht Ihnen, einer Komponente zusätzlichen Code oder Features hinzuzufügen und gleichzeitig das Risiko für Komponenten zu minimieren, die von ihr abhängig sind. Sie ermöglicht auch eine granulare Ausfallsicherheit auf Komponentenebene, bei der Sie die zugrunde liegende Implementierung der Abhängigkeit aufskalieren oder sogar ändern können.

Um die Ausfallsicherheit durch lose Verkopplung weiter zu verbessern, legen Sie Komponenten-Interaktionen nach Möglichkeit als asynchron fest. Dieses Modell eignet sich für jede Interaktion, bei der keine sofortige Antwort benötigt wird, sondern die Bestätigung ausreicht, dass eine Anfrage registriert wurde. Es umfasst eine Komponente, die Ereignisse generiert, und eine andere Komponente, die sie konsumiert. Die beiden Komponenten werden nicht durch direkte point-to-point Interaktion integriert, sondern normalerweise über eine dauerhafte Zwischenspeicherschicht, z. B. eine SQS Amazon-Warteschlange, eine Streaming-Datenplattform wie Amazon Kinesis oder AWS Step Functions.

Abbildung 4: Abhängigkeiten etwa zwischen Warteschlangensystemen und Load Balancer sind lose gekoppelt

SQSAmazon-Warteschlangen und AWS Step Functions sind nur zwei Möglichkeiten, eine Zwischenschicht für lose Kopplung hinzuzufügen. Eventgesteuerte Architekturen können auch AWS Cloud mithilfe von Amazon erstellt werden EventBridge, wodurch Kunden (Event-Produzenten) von

den Diensten, auf die sie angewiesen sind (Event-Konsumenten), abstrahieren können. Amazon Simple Notification Service (AmazonSNS) ist eine effektive Lösung, wenn Sie Push-basiertes Messaging mit hohem Durchsatz benötigen. many-to-many Mithilfe von SNS Amazon-Themen können Ihre Publisher-Systeme Nachrichten zur parallel Verarbeitung an eine große Anzahl von Abonnenten-Endpunkten fächern.

Warteschlangen bieten zwar mehrere Vorteile, doch Anfragen, die älter als ein Schwellenwert sind (oft Sekunden), sollten in den meisten harten Echtzeitsystemen als veraltet betrachtet (der Client hat aufgegeben und wartet nicht mehr auf eine Antwort) und nicht verarbeitet werden. Auf diese Weise können stattdessen neuere (und wahrscheinlich noch gültige Anfragen) verarbeitet werden.

Gewünschtes Ergebnis: Wenn Sie lose gekoppelte Abhängigkeiten implementieren, können Sie die Fehlerfläche auf Komponentenebene minimieren, was die Diagnose und Lösung von Problemen erleichtert. Außerdem vereinfacht es die Entwicklungszyklen, da die Teams Änderungen auf modularer Ebene implementieren können, ohne die Leistung anderer Komponenten, die davon abhängen, zu beeinträchtigen. Dieser Ansatz ermöglicht eine Aufskalierung auf Komponentenebene auf Grundlage des Ressourcenbedarfs sowie der Auslastung einer Komponente und trägt so zur Kosteneffizienz bei.

Typische Anti-Muster:

- Bereitstellen eines monolithischen Workloads.
- Direkter Aufruf APIs zwischen Workload-Stufen, ohne dass ein Failover oder eine asynchrone Verarbeitung der Anfrage möglich ist.
- Enge Verknüpfung mithilfe gemeinsam genutzter Daten. Lose gekoppelte Systeme sollten die gemeinsame Nutzung von Daten durch gemeinsam genutzte Datenbanken oder andere Formen der eng gekoppelten Datenspeicherung vermeiden, da dies wieder zu einer engen Verknüpfung führen und die Skalierbarkeit behindern kann.
- Gegendruck wird ignoriert. Ihr Workload sollte in der Lage sein, die eingehenden Daten zu verlangsamen oder zu stoppen, wenn eine Komponente sie nicht mit der gleichen Geschwindigkeit verarbeiten kann.

Vorteile der Nutzung dieser bewährten Methode: Eine lose Verkoppelung hilft dabei, das Verhalten einer Komponente von anderen Komponenten zu isolieren, die von ihr abhängen, wodurch die Resilienz und Agilität erhöht werden. Fehler in einer Komponente sind von anderen isoliert.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Implementieren lose gekoppelter Abhängigkeiten. Es gibt verschiedene Lösungen, mit denen Sie lose gekoppelte Anwendungen erstellen können. Dazu gehören Dienste für die Implementierung vollständig verwalteter Warteschlangen, automatisierter Workflows, die Reaktion auf Ereignisse und APIs vieles mehr, die dazu beitragen können, das Verhalten von Komponenten von anderen Komponenten zu isolieren und so die Widerstandsfähigkeit und Agilität zu erhöhen.

- Erstellen Sie ereignisgesteuerte Architekturen: Amazon EventBridge hilft Ihnen beim Aufbau lose gekoppelter und verteilter ereignisgesteuerter Architekturen.
- Implementieren Sie Warteschlangen in verteilten Systemen: Sie können Amazon Simple Queue Service (AmazonSQS) verwenden, um verteilte Systeme zu integrieren und zu entkoppeln.
- Komponenten als Microservices containerisieren: Microservices ermöglichen es Teams, Anwendungen zu erstellen, die aus kleinen unabhängigen Komponenten bestehen, die über klar definierte Kanäle kommunizieren. APIs Amazon Elastic Container Service (AmazonECS) und Amazon Elastic Kubernetes Service (AmazonEKS) können Ihnen helfen, schneller mit Containern zu beginnen.
- Verwalten Sie Workflows mit Step Functions: Mit Step Functions können Sie mehrere AWS Services zu flexiblen Workflows koordinieren.
- Nutzen Sie die Messaging-Architekturen Publish-Subscribe (Pub/Sub): Amazon Simple Notification Service (AmazonSNS) ermöglicht die Nachrichtenzustellung von Verlagen an Abonnenten (auch bekannt als Produzenten und Verbraucher).

Implementierungsschritte

- Komponenten in einer ereignisgesteuerten Architektur werden durch Ereignisse ausgelöst. Ereignisse sind Aktionen, die in einem System stattfinden, z. B. wenn ein Benutzer einen Artikel in den Warenkorb legt. Wenn eine Aktion erfolgreich ist, wird ein Ereignis erzeugt, das die nächste Komponente des Systems auslöst.
  - Entwicklung ereignisgesteuerter Anwendungen mit Amazon EventBridge
  - AWS re:Invent 2022 — Entwicklung ereignisgesteuerter Integrationen mit Amazon EventBridge
- Verteilte Nachrichtensysteme haben drei Hauptbestandteile, die für eine warteschlangenbasierte Architektur implementiert werden müssen. Sie umfassen Komponenten des verteilten Systems, die Warteschlange, die für die Entkopplung verwendet wird (verteilt auf SQS Amazon-Servern), und die Nachrichten in der Warteschlange. Ein typisches System hat einen Produzenten, der die Nachricht in die Warteschlange einstellt, und einen Verbraucher, der die Nachricht aus der Warteschlange

empfängt. Die Warteschlange speichert Nachrichten aus Redundanzgründen auf mehreren SQS Amazon-Servern.

  - Grundlegende SQS Amazon-Architektur
  - Send Messages Between Distributed Applications with Amazon Simple Queue Service
- Wenn Microservices gut genutzt werden, verbessern sie die Wartbarkeit und die Skalierbarkeit, da lose gekoppelte Komponenten von unabhängigen Teams verwaltet werden. Sie ermöglichen zudem die Isolierung von Verhaltensweisen auf eine einzelne Komponente im Falle von Änderungen.
  - Implementierung von Microservices auf AWS
  - Let’s Architect! Architektur von Microservices mit Containern
- Mit können AWS Step Functions Sie unter anderem verteilte Anwendungen erstellen, Prozesse automatisieren und Microservices orchestrieren. Die Orchestrierung mehrerer Komponenten in einem automatisierten Workflow ermöglicht es Ihnen, Abhängigkeiten in Ihrer Anwendung zu entkoppeln.
  - Erstellen Sie einen serverlosen Workflow mit und AWS Step FunctionsAWS Lambda
  - Erste Schritte mit AWS Step Functions

Ressourcen

Zugehörige Dokumente:

- AmazonEC2: Idempotenz sicherstellen
- Die Amazon Builders' Library: Herausforderungen für verteilte Systeme
- Die Amazon Builders' Library: Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee
- Was ist Amazon EventBridge?
- Was ist Amazon Simple Queue Service?
- Break up with your monolith
- Orchestrieren Sie warteschlangenbasierte Microservices mit und Amazon AWS Step Functions SQS
- Grundlegende SQS Amazon-Architektur
- Warteschlangenbasierte Architektur

Zugehörige Videos:

- AWS New York Summit 2019: Einführung in ereignisgesteuerte Architekturen und Amazon (05) EventBridge MAD2
- AWS re:Invent 2018: Closed Loops and Opening Minds: Wie man die Kontrolle über große und kleine Systeme übernimmt ARC337 (beinhaltet lockere Kopplung, konstante Arbeit, statische Stabilität)
- AWS re:Invent 2019: Umstellung auf ereignisgesteuerte Architekturen (08) SVS3
- AWS re:Invent 2019: Skalierbare, serverlose, ereignisgesteuerte Anwendungen mit Amazon und Lambda SQS
- AWS re:Invent 2022 — Entwicklung ereignisgesteuerter Integrationen mit Amazon EventBridge
- AWS re:Invent 2017: Elastic Load Balancing im Detail und Best Practices

REL04-BP03 Arbeite ständig

Bei größeren, schnellen Lastveränderungen können Systeme ausfallen. Wenn Ihr Workload beispielsweise eine Zustandsprüfung ausführt, die den Zustand vieler tausend Server überwacht, sollte er jedes Mal die gleiche Nutzlast senden (einen vollständigen Snapshot des aktuellen Status). Unabhängig davon, ob keine Server oder alle Server ausfallen, führt das System für die Zustandsprüfung die Aufgaben stetig und ohne große, schnelle Änderungen aus.

Wenn das Zustandsprüfungssystem beispielsweise 100 000 Server überwacht, ist die Last darauf angesichts der normalerweise geringen Serverausfallrate nominal. Wenn jedoch ein großes Ereignis die Hälfte dieser Server fehlerhaft macht, wäre das Zustandsprüfungssystem überfordert, wenn es versucht, Benachrichtigungssysteme zu aktualisieren und den Status an seine Clients zu kommunizieren. Stattdessen sollte das Zustandsprüfungssystem jedes Mal den vollständigen Snapshot des aktuellen Status senden. 100 000 Server-Zustände, die jeweils durch ein Bit dargestellt werden, entsprächen nur eine Nutzlast von 12,5 KB. Unabhängig davon, ob keine oder alle Server ausfallen – das System für die Zustandsprüfung erledigt seine Arbeit konstant und große, schnelle Änderungen stellen keine Bedrohung für die Systemstabilität dar. Auf diese Weise führt Amazon Route 53 Zustandsprüfungen für Endpunkte (wie z. B. IP-Adressen) durch, um zu ermitteln, wie Endbenutzer an diese weitergeleitet werden.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Niedrig

Implementierungsleitfaden

- Führen Sie Aufgaben konstant aus, sodass auch bei großen, schnellen Lastveränderungen keine Fehler auf Systemen auftreten.

- Implementieren Sie lose gekoppelte Abhängigkeiten. Abhängigkeiten etwa zwischen Warteschlangensystemen, Streaming-Systemen, Workflows und Load Balancern sind lose gekoppelt. Eine lose Verkoppelung hilft, das Verhalten einer Komponente von anderen Komponenten zu isolieren, die von ihr abhängig sind. Dies verbessert Resilienz und Agilität.
  - Die Amazon Builders' Library: Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee
  - AWS re:Invent 2018: Regelkreise schließen und neue Denkansätze eröffnen: Wie man die Kontrolle über große und kleine Systeme übernimmt ARC337 (beinhaltet ständige Arbeit)
    - Beispiel: Zustandsprüfungssystem, das 100.000 Server überwacht: Entwickeln Sie die Workloads so, dass die Nutzlastgrößen unabhängig von der Anzahl der Erfolge oder Ausfälle konstant bleiben.

Ressourcen

Zugehörige Dokumente:

- AmazonEC2: Idempotenz sicherstellen
- Die Amazon Builders' Library: Herausforderungen für verteilte Systeme
- Die Amazon Builders' Library: Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee

Zugehörige Videos:

- AWS New York Summit 2019: Einführung in ereignisgesteuerte Architekturen und Amazon (05) EventBridge MAD2
- AWS re:Invent 2018: Close-Loops und offene Denkansätze: Wie man die Kontrolle über große und kleine Systeme übernimmt (beinhaltet ständige Arbeit) ARC337
- AWS re:Invent 2018: Close Loops und Opening Minds: So übernehmen Sie die Kontrolle über große und kleine Systeme ARC337 (beinhaltet lockere Kopplung, konstante Arbeit, statische Stabilität)
- AWS re:Invent 2019: Umstellung auf ereignisgesteuerte Architekturen (08) SVS3

REL04-BP04 Festlegen von Mutationsoperationen als idempotent

Ein idempotenter Service garantiert, dass jede Anfrage genau einmal verarbeitet wird. Das bedeutet, dass das Senden mehrerer identischer Anfragen den gleichen Effekt hat wie das Senden einer einzelnen Anfrage. Dies vereinfacht es für einen Client, Wiederholungen zu implementieren. So muss

nicht befürchtet werden, dass eine Anfrage fälschlicherweise mehrfach verarbeitet wird. Zu diesem Zweck können Clients API-Anfragen mit einem Idempotenz-Token ausgeben. Das gleiche Token wird verwendet, wenn die Anfrage wiederholt wird. Eine idempotente Service-API gibt mithilfe des Tokens eine Antwort zurück, die identisch mit der Antwort ist, die beim ersten Abschluss der Anfrage zurückgegeben wurde, selbst wenn sich der zugrunde liegende Zustand des Systems geändert hat.

In einem verteilten System ist es relativ einfach, eine Aktion höchstens einmal (der Client führt nur eine Anforderung aus) oder mindestens einmal (Anforderung wird ausgeführt, bis der Client eine Erfolgsbenachrichtigung erhält) durchzuführen. Es ist jedoch schwieriger, zu gewährleisten, dass eine Aktion genau einmal ausgeführt wird, so dass das Erstellen mehrerer identischer Anfragen den gleichen Effekt hat wie das Erstellen einer einzelnen Anfrage. Durch die Verwendung von idempotenten Tokens in APIs können Services einmal oder mehrmals eine mutierende Anforderung erhalten, ohne dass doppelte Datensätze erstellt werden müssen oder andere Probleme entstehen.

Gewünschtes Ergebnis: Sie verfügen über einen konsistenten, gut dokumentierten und häufig verwendeten Ansatz zur Sicherstellung der Idempotenz für alle Komponenten und Services.

Typische Anti-Muster:

- Sie wenden Idempotenz unterschiedslos an, auch wenn sie nicht benötigt wird.
- Sie führen eine zu komplexe Logik für die Implementierung von Idempotenz ein.
- Sie verwenden Zeitstempel als Schlüssel für Idempotenz. Dies kann zu Ungenauigkeiten führen, die auf eine Verschiebung der Uhrzeit oder darauf zurückzuführen sind, dass mehrere Clients dieselben Zeitstempel verwenden, um Änderungen vorzunehmen.
- Sie speichern vollständige Nutzdaten für Idempotenz. Bei diesem Ansatz speichern Sie vollständige Nutzdaten für jede Anfrage und überschreiben sie bei jeder neuen Anfrage. Dies kann Leistung und Skalierbarkeit beeinträchtigen.
- Sie generieren inkonsistente Service-Schlüssel. Ohne konsistente Schlüssel erkennen Services duplizierte Anfragen möglicherweise nicht, was zu unbeabsichtigten Ergebnissen führt.

Vorteile der Nutzung dieser bewährten Methode:

- Höhere Skalierbarkeit: Das System kann Wiederholungsversuche und doppelte Anfragen verarbeiten, ohne eine zusätzliche Logik oder eine komplexe Statusverwaltung anwenden zu müssen.
- Verbesserte Zuverlässigkeit: Mit Idempotenz können Services mehrere identische Anfragen konsistent verarbeiten, wodurch das Risiko unbeabsichtigter Nebenwirkungen oder duplizierter

Datensätze reduziert wird. Dies ist besonders wichtig in verteilten Systemen, in denen Netzwerkausfälle und Wiederholungsversuche häufig sind.

- Verbesserte Datenkonsistenz: Da dieselbe Anfrage zu derselben Antwort führt, trägt Idempotenz dazu bei, die Datenkonsistenz in verteilten Systemen aufrechtzuerhalten. Dies ist wichtig, um die Integrität von Transaktionen und Vorgängen aufrechtzuerhalten.
- Fehlerbehandlung: Idempotenz-Tokens vereinfachen die Fehlerbehandlung. Wenn ein Client aufgrund eines Problems keine Antwort erhält, kann er die Anfrage problemlos erneut mit demselben Idempotenz-Token senden.
- Betriebliche Transparenz: Idempotenz ermöglicht eine bessere Überwachung und Protokollierung. Services können Anfragen mit ihren Idempotenz-Tokens protokollieren, was das Nachverfolgen und Debuggen von Problemen erleichtert.
- Vereinfachter API-Vertrag: Dies kann den Vertrag zwischen den client- und serverseitigen Systemen vereinfachen und Sorgen hinsichtlich einer fehlerhaften Datenverarbeitung verringern.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

In einem verteilten System ist es einfach, eine Aktion höchstens einmal (der Client führt nur eine Anforderung aus) oder mindestens einmal (Anforderung wird so lange ausgeführt, bis der Erfolg bestätigt wird) durchzuführen. Es ist jedoch schwierig, die Aktion genau einmal ausführen zu lassen. Um dies zu erreichen, sollten Ihre Kunden für jede Anfrage ein Idempotenz-Token generieren und bereitstellen.

Durch die Verwendung von Idempotenz-Tokens kann ein Service zwischen neuen und wiederholten Anfragen unterscheiden. Wenn ein Service eine Anfrage mit einem Idempotenz-Token empfängt, prüft er, ob das Token bereits verwendet wurde. Wenn das Token bereits verwendet wurde, ruft der Service die gespeicherte Antwort ab und gibt sie zurück. Wenn das Token neu ist, verarbeitet der Service die Anfrage, speichert die Antwort zusammen mit dem Token und gibt dann die Antwort zurück. Dieser Mechanismus macht alle Antworten idempotent, was die Zuverlässigkeit und Konsistenz des verteilten Systems erhöht.

Idempotenz ist auch ein wichtiges Verhalten ereignisgesteuerter Architekturen. Diese Architekturen werden in der Regel durch eine Nachrichtenwarteschlange wie Amazon SQS, Amazon MQ, Amazon Kinesis Streams oder Amazon Managed Streaming for Apache Kafka (MSK) unterstützt. Unter bestimmten Umständen kann eine Nachricht, die nur einmal veröffentlicht wurde, versehentlich mehr als einmal zugestellt werden. Wenn ein Herausgeber Idempotenz-Token generiert und in Nachrichten

einfügt, wird damit angefordert, dass die Verarbeitung doppelt empfangener Nachrichten nicht zu einer wiederholten Aktion für dieselbe Nachricht führt. Verbraucher sollten jedes erhaltene Token nachverfolgen und Nachrichten ignorieren, die duplizierte Token enthalten.

Services und Verbraucher sollten das empfangene Idempotenz-Token auch an alle nachgelagerten Services weitergeben, die sie aufrufen. Jeder nachgelagerte Service in der Verarbeitungskette ist ebenfalls dafür verantwortlich, sicherzustellen, dass Idempotenz implementiert wird, um den Nebeneffekt einer mehrfachen Verarbeitung einer Nachricht zu vermeiden.

Implementierungsschritte

1. Identifizieren idempotenter Operationen

   Ermitteln Sie, für welche Operationen Idempotenz erforderlich ist. Dazu gehören in der Regel die HTTP-Methoden POST, PUT und DELETE sowie Operationen zum Einfügen, Aktualisieren oder Löschen von Datenbanken. Operationen, bei denen der Status nicht verändert wird, z. B. schreibgeschützte Abfragen, erfordern normalerweise keine Idempotenz, es sei denn, es gibt Nebeneffekte.
2. Verwenden eindeutiger Bezeichner

   Fügen Sie jeder idempotenten Operationsanforderung, die vom Absender gesendet wird, ein eindeutiges Token hinzu, entweder direkt in der Anfrage oder als Teil ihrer Metadaten (z. B. einen HTTP-Header). Auf diese Weise kann der Empfänger duplizierte Anforderungen oder Operationen erkennen und verarbeiten. Zu den üblicherweise für Token verwendeten Bezeichnern gehören Universally Unique Identifiers (UUIDs) und K-Sortable Unique Identifiers (KSUIDs).
3. Verfolgen und Verwalten des Status

   Behalten Sie den Status jeder Operation oder Anforderung in Ihrem Workload bei. Dies kann erreicht werden, indem das Idempotenz-Token und der entsprechende Status (z. B. ausstehend, abgeschlossen oder fehlgeschlagen) in einer Datenbank, einem Cache oder einem anderen persistenten Speicher gespeichert werden. Diese Statusinformationen ermöglichen dem Workload die Identifizierung und Verarbeitung duplizierter Anforderungen oder Operationen.

   Sorgen Sie für Konsistenz und Atomizität, indem Sie bei Bedarf geeignete Mechanismen zur Kontrolle der Gleichzeitigkeit verwenden, z. B. Sperren, Transaktionen oder optimistische Gleichzeitigkeitskontrollen. Dazu gehören das Aufzeichnen des idempotenten Tokens und das Ausführen aller Mutationsoperationen, die mit der Bearbeitung der Anfrage verbunden sind. Auf diese Weise wird verhindert, dass es zu Race-Bedingungen kommt, und sichergestellt, dass idempotente Operationen korrekt ausgeführt werden.

Entfernen Sie regelmäßig alte Idempotenz-Token aus dem Datenspeicher, um Speicherplatz und Leistung zu verwalten. Wenn Ihr Speichersystem es unterstützt, sollten Sie die Verwendung von Ablaufzeitstempeln für Daten in Betracht ziehen (häufig auch als Time to Live- oder TTL-Werte bezeichnet). Die Wahrscheinlichkeit einer Wiederverwendung von Idempotenz-Token nimmt mit der Zeit ab.

Zu den gängigen AWS-Speicheroptionen, die typischerweise zum Speichern von Idempotenz-Token und des zugehörigen Status verwendet werden, gehören:

- Amazon DynamoDB: DynamoDB ist ein NoSQL-Datenbankservice, der eine Leistung mit niedriger Latenz und eine hohe Verfügbarkeit bietet und sich daher gut für die Speicherung von Daten im Zusammenhang mit Idempotenz eignet. Das Schlüssel-Wert- und Dokumentdatenmodell von DynamoDB ermöglicht das effiziente Speichern und Abrufen von Idempotenz-Token und der zugehörigen Statusinformationen. DynamoDB kann Idempotenz-Token auch automatisch ablaufen lassen, wenn Ihre Anwendung beim Einfügen einen TTL-Wert festlegt.
- Amazon ElastiCache: ElastiCache kann Idempotenz-Token mit hohem Durchsatz, geringer Latenz und niedrigen Kosten speichern. Sowohl ElastiCache (Redis) als auch ElastiCache (Memcached) können Idempotenz-Token auch automatisch ablaufen lassen, wenn Ihre Anwendung beim Einfügen einen TTL-Wert festlegt.
- Amazon Relational Database Service (RDS): Sie können Amazon RDS verwenden, um Idempotenz-Token und zugehörige Statusinformationen zu speichern, insbesondere, wenn Ihre Anwendung bereits eine relationale Datenbank für andere Zwecke verwendet.
- Amazon Simple Storage Service (S3): Amazon S3 ist ein hoch skalierbarer und langlebiger Objektspeicherservice, der zum Speichern von Idempotenz-Token und der zugehörigen Metadaten verwendet werden kann. Die Versionsverwaltungsfunktionen von S3 können besonders nützlich sein, um den Status idempotenter Operationen aufrechtzuerhalten. Die Wahl des Speicherservices ist in der Regel von Faktoren wie der Menge der Daten im Zusammenhang mit Idempotenz, den erforderlichen Leistungsmerkmalen, der benötigten Speicherdauer und Verfügbarkeit sowie der Art und Weise abhängig, wie der Idempotenzmechanismus in die gesamte Workload-Architektur integriert ist.

4. Implementieren von idempotenten Operationen

   Entwerfen Sie Ihre API- und Workload-Komponenten für Idempotenz. Integrieren Sie Idempotenzprüfungen in Ihre Workload-Komponenten. Überprüfen Sie vor der Verarbeitung einer Anforderung oder Ausführung einer Operation, ob der eindeutige Bezeichner bereits verarbeitet

wurde. Wenn dies der Fall ist, geben Sie das vorherige Ergebnis zurück, anstatt die Operation erneut auszuführen. Wenn ein Client beispielsweise eine Anforderung zur Erstellung eines Benutzers sendet, überprüfen Sie, ob bereits ein Benutzer mit demselben eindeutigen Bezeichner vorhanden ist. Wenn der Benutzer vorhanden ist, sollten die vorhandenen Benutzerinformationen zurückgegeben werden, statt einen neuen Benutzer zu erstellen. Ebenso sollte ein Verbraucher Nachrichten ignorieren, die duplizierte Idempotenz-Token enthalten.

Erstellen Sie umfassende Testsuiten zur Validierung der Idempotenz von Anforderungen. Diese sollten ein breites Spektrum von Szenarien abdecken, z. B. erfolgreiche, fehlgeschlagene und doppelte Anforderungen.

Wenn Ihr Workload AWS Lambda-Funktionen nutzt, sollten Sie Powertools für AWS Lambda in Betracht ziehen. Powertools für AWS Lambda ist ein Entwickler-Toolkit zur Implementierung bewährter Serverless-Methoden und zur Steigerung der Entwicklungsgeschwindigkeit bei der Arbeit mit AWS Lambda-Funktionen. Insbesondere enthält der Kit ein Hilfsprogramm, mit dem Sie Ihre Lambda-Funktionen in idempotente Operationen konvertieren können, die sicher wiederholt werden können.

5. Klare Kommunikation der Idempotenz

   Dokumentieren Sie Ihre API- und Workload-Komponenten, um die idempotente Natur der Operationen deutlich zu machen. Dies hilft Clients, das erwartete Verhalten zu verstehen und zu wissen, wie sie zuverlässig mit Ihrem Workload interagieren können.

6. Überwachen und Prüfen

   Implementieren Sie Überwachungs- und Prüfmechanismen, um Probleme im Zusammenhang mit der Idempotenz von Antworten aufzudecken, z. B. unerwartete Antwortvarianten oder eine übermäßige Verarbeitung duplizierter Anforderungen. Dies kann Ihnen helfen, Probleme oder unerwartete Verhaltensweisen in Ihren Workloads zu erkennen und zu untersuchen.

Ressourcen

Zugehörige bewährte Methoden:

- REL05-BP03 Steuern und Einschränken von Wiederholungsaufrufen
- REL06-BP01 Überwachen aller Komponenten des Workloads (Generierung)
- REL06-BP03 Senden von Benachrichtigungen (Verarbeitung und Benachrichtigung in Echtzeit)
- REL08-BP02 Integrieren von Funktionstests in die Bereitstellung

Zugehörige Dokumente:

- Amazon Builders' Library: Sichere Wiederholungen mit idempotenten APIs
- Amazon Builders' Library: Herausforderungen bei verteilten Systemen
- Amazon Builders' Library: Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee
- Amazon Elastic Container Service: Sicherstellen der Idempotenz
- Wie mache ich meine Lambda-Funktion idempotent?
- Sicherstellen der Idempotenz bei Amazon-EC2-API-Anforderungen

Zugehörige Videos:

- Entwickeln verteilter Anwendungen mit ereignisgesteuerter Architektur – AWS Online Tech Talks
- AWS re:Invent 2023 – Building next-generation applications with event-driven architecture
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2023 – Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2018 – Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019 – Moving to event-driven architectures (SVS308)

Zugehörige Tools:

- Idempotenz mit AWS Lambda Powertools (Java)
- Idempotenz mit AWS Lambda Powertools (Python)
- AWS Lambda Powertools – GitHub-Seite

Entwerfen von Interaktionen in einem verteilten System, um Ausfälle abzumildern oder zu verkraften

Verteilte Systeme nutzen Kommunikationsnetzwerke, um Komponenten (wie Server oder Services) miteinander zu verbinden. Ihre Workload muss trotz Datenverlust oder höherer Latenz in diesen Netzwerken zuverlässig ausgeführt werden. Die Komponenten des verteilten Systems müssen so funktionieren, dass sie sich nicht negativ auf andere Komponenten oder die Workload auswirken. Diese bewährten Methoden sorgen dafür, dass Workloads Belastungen oder Fehlern standhalten,

sich schneller davon erholen und die Auswirkungen solcher Beeinträchtigungen abgeschwächt werden. Das Ergebnis ist eine verbesserte mittlere Reparaturzeit (MTTR).

Diese bewährten Methoden verhindern Ausfälle und verbessern die mittlere Betriebsdauer zwischen Ausfällen (MTBF).

Bewährte Methoden

- REL05-BP01 Implementieren einer ordnungsgemäßen Funktionsminderung, um harte Abhängigkeiten in weiche zu ändern
- REL05-BP02 Drosselung von Anfragen
- REL05-BP03 Steuerung und Begrenzung von Wiederholungsaufrufen
- REL05-BP04 Schnelles Scheitern und Begrenzen von Warteschlangen
- REL05-BP05 Client-Timeouts festlegen
- REL05-BP06 Systeme soweit möglich zustandslos machen
- REL05-BP07 Nothebel einbauen

REL05-BP01 Implementieren einer ordnungsgemäßen Funktionsminderung, um harte Abhängigkeiten in weiche zu ändern

Anwendungskomponenten sollten weiterhin ihre Kernfunktion erfüllen, auch wenn Abhängigkeiten nicht mehr verfügbar sind. Sie liefern möglicherweise leicht veraltete Daten, alternative Daten oder sogar keine Daten. Dadurch wird sichergestellt, dass die Gesamtsystemfunktion nur minimal durch lokale Ausfälle beeinträchtigt wird, während gleichzeitig der zentrale Geschäftswert gewährleistet ist.

Gewünschtes Ergebnis: Wenn die Abhängigkeiten einer Komponente fehlerhaft sind, kann die Komponente selbst weiterhin funktionieren, wenn auch in eingeschränkter Weise. Komponentenausfälle sollten als normaler Geschäftsbetrieb betrachtet werden. Arbeitsabläufe sollten so konzipiert sein, dass solche Ausfälle nicht zu einem vollständigen Ausfall oder zumindest zu vorhersehbaren und wiederherstellbaren Zuständen führen.

Typische Anti-Muster:

- Die erforderlichen Kerngeschäftsfunktionen wurden nicht identifiziert. Es wird nicht getestet, ob die Komponenten auch bei Abhängigkeitsfehlern funktionsfähig sind.
- Es werden keine Daten zu Fehlern bereitgestellt oder wenn nur eine von mehreren Abhängigkeiten nicht verfügbar ist und Teilergebnisse dennoch zurückgegeben werden können.

- Es entsteht ein inkonsistenter Zustand, wenn eine Transaktion teilweise fehlschlägt.
- Es gibt keine alternative Möglichkeit, auf einen zentralen Parameterspeicher zuzugreifen.
- Lokale Zustände werden aufgrund einer fehlgeschlagenen Aktualisierung ungültig oder geleert, ohne die Konsequenzen zu berücksichtigen.

Vorteile der Nutzung dieser bewährten Methode: Eine schrittweise Degradation verbessert die Verfügbarkeit des gesamten Systems und gewährleistet die Funktionsfähigkeit der wichtigsten Funktionen auch bei Ausfällen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Die Implementierung einer schrittweisen Degradation trägt dazu bei, die Auswirkungen von Abhängigkeitsfehlern auf die Komponentenfunktion zu minimieren. Im Idealfall erkennt eine Komponente Abhängigkeitsfehler und umgeht sie so, dass sich dies nur minimal auf andere Komponenten oder Kunden auswirkt.

Eine Architektur, die auf eine schrittweise Degradation ausgerichtet ist, bedeutet, potenzielle Ausfallmodi beim Entwurf von Abhängigkeiten zu berücksichtigen. Sorgen Sie für jeden Ausfallmodus für eine Möglichkeit, aufrufenden Komponenten oder Kunden die meisten oder zumindest die wichtigsten Funktionen der Komponente bereitzustellen. Diese Überlegungen können zu zusätzlichen Anforderungen werden, die getestet und verifiziert werden können. Im Idealfall ist eine Komponente in der Lage, ihre Kernfunktion auf akzeptable Weise auszuführen, selbst wenn eine oder mehrere Abhängigkeiten ausfallen.

Dies ist sowohl eine geschäftliche als auch eine technische Diskussion. Alle Geschäftsanforderungen sind wichtig und sollten nach Möglichkeit erfüllt werden. Es ist jedoch immer noch sinnvoll, sich zu fragen, was passieren soll, wenn nicht alle erfüllt werden können. Ein System kann so konzipiert werden, dass es verfügbar und konsistent ist. Doch was davon ist wichtiger, wenn auf eines davon verzichtet werden muss? Bei der Zahlungsabwicklung könnte dies die Konsistenz sein. Bei einer Echtzeitanwendung ist es eher die Verfügbarkeit. Bei einer kundenseitigen Website kann die Antwort von den Kundenerwartungen abhängen.

Was das bedeutet, hängt von den Anforderungen der Komponente ab und davon, was als ihre Kernfunktion angesehen werden sollte. Beispiel:

- Eine E-Commerce-Website kann Daten aus verschiedenen Systemen wie personalisierte Empfehlungen, bestbewertete Produkte und den Status von Kundenbestellungen auf der

Startseite anzeigen. Wenn ein Upstream-System ausfällt, ist es immer noch sinnvoll, alles andere anzuzeigen, anstatt einem Kunden eine Fehlerseite anzuzeigen.

- Eine Komponente, die Batch-Schreibvorgänge durchführt, kann einen Stapel trotzdem weiterverarbeiten, wenn eine der einzelnen Operationen fehlschlägt. Es sollte einfach sein, einen Wiederholungsmechanismus zu implementieren. Geben Sie dazu Informationen dazu zurück, welche Operationen erfolgreich, welche fehlgeschlagen und warum sie fehlgeschlagen sind. Oder stellen Sie fehlgeschlagene Anfragen in eine Warteschlange für unzustellbare Nachrichten, um asynchrone Wiederholungsversuche zu implementieren. Informationen über fehlgeschlagene Operationen sollten ebenfalls protokolliert werden.
- Ein System, das Transaktionen verarbeitet, muss überprüfen, ob entweder alle oder keine einzelnen Aktualisierungen ausgeführt werden. Bei verteilten Transaktionen kann das Saga-Muster verwendet werden, um vorherige Operationen rückgängig zu machen, falls ein späterer Vorgang derselben Transaktion fehlschlägt. Hier besteht die Kernfunktion darin, die Konsistenz aufrechtzuerhalten.
- Zeitkritische Systeme sollten in der Lage sein, mit Abhängigkeiten umzugehen, die nicht rechtzeitig reagieren. In diesen Fällen kann das Unterbrechermuster verwendet werden. Wenn bei Antworten aus einer Abhängigkeit eine Zeitüberschreitung auftritt, kann das System in einen geschlossenen Zustand wechseln, in dem keine weiteren Aufrufe getätigt werden.
- Eine Anwendung kann Parameter aus einem Parameterspeicher lesen. Es kann nützlich sein, Container-Images mit einem Satz von Standardparametern zu erstellen und diese zu verwenden, falls der Parameterspeicher nicht verfügbar ist.

Beachten Sie, dass die im Falle eines Komponentenausfalls eingeschlagenen Pfade getestet werden müssen und deutlich einfacher sein sollten als der primäre Pfad. Allgemein sollten Fallback-Strategien vermieden werden.

Implementierungsschritte

Identifizieren Sie externe und interne Abhängigkeiten. Überlegen Sie, welche Arten von Fehlern bei ihnen auftreten können. Überlegen Sie, wie Sie die negativen Auswirkungen dieser Ausfälle auf vor- und nachgeschaltete Systeme und Kunden minimieren können.

Im Folgenden finden Sie eine Liste von Abhängigkeiten und wie Sie sie schrittweise degradieren können, wenn sie ausfallen:

1. Teilweiser Ausfall von Abhängigkeiten: Eine Komponente kann mehrere Anfragen an nachgelagerte Systeme stellen, entweder in Form mehrerer Anfragen an ein System oder

in Form einer Anfrage an jeweils mehrere Systeme. Je nach Unternehmenskontext können unterschiedliche Vorgehensweisen angemessen sein (weitere Einzelheiten finden Sie in den vorherigen Beispielen in den Implementierungsleitfäden).

2. Ein nachgelagertes System kann Anfragen aufgrund der hohen Auslastung nicht verarbeiten: Wenn Anfragen an ein nachgelagertes System immer wieder fehlschlagen, ist es nicht sinnvoll, es erneut zu versuchen. Dies kann ein bereits überlastetes System zusätzlich belasten und die Wiederherstellung erschweren. Hier kann das Unterbrechermuster verwendet werden, das fehlgeschlagene Aufrufe an ein nachgelagertes System überwacht. Wenn eine große Anzahl von Aufrufen fehlschlägt, werden keine weiteren Anfragen mehr an das nachgelagerte System gesendet und nur gelegentlich Aufrufe durchgelassen, um zu testen, ob das nachgelagerte System wieder verfügbar ist.
3. Ein Parameterspeicher ist nicht verfügbar: Um einen Parameterspeicher umzuwandeln, können Soft Dependency Caching oder vernünftige Standardwerte verwendet werden, die in Container-Images oder Machine Images enthalten sind. Beachten Sie, dass diese Standardwerte auf dem neuesten Stand gehalten und in die Testsuiten aufgenommen werden müssen.
4. Ein Überwachungsservice oder eine andere nicht funktionale Abhängigkeit ist nicht verfügbar: Wenn eine Komponente zeitweise nicht in der Lage ist, Protokolle, Metriken oder Spuren an einen zentralen Überwachungsservice zu senden, ist es oft am besten, Geschäftsfunktionen weiterhin wie gewohnt auszuführen. Es ist oft nicht akzeptabel, Metriken über einen längeren Zeitraum stillschweigend nicht zu protokollieren oder weiterzuleiten. In einigen Anwendungsfällen können auch vollständige Auditeinträge erforderlich sein, um die Compliance-Anforderungen zu erfüllen.
5. Eine primäre Instances einer relationalen Datenbank ist möglicherweise nicht verfügbar: Amazon Relational Database Service kann, wie fast alle relationalen Datenbanken, nur eine primäre Writer-Instance haben. Dies führt zu einem einzigen Fehlerpunkt für Schreib-Workloads und erschwert die Skalierung. Dies kann teilweise gemildert werden, indem eine Multi-AZ-Konfiguration für hohe Verfügbarkeit oder Amazon Aurora Serverless für eine bessere Skalierung verwendet wird. Bei sehr hohen Verfügbarkeitsanforderungen kann es sinnvoll sein, sich überhaupt nicht auf den primären Writer zu verlassen. Für Abfragen, die nur lesen, können Lesereplikate verwendet werden, die Redundanz und die Möglichkeit bieten, nicht nur hoch-, sondern auch aufzuskalieren. Schreibvorgänge können gepuffert werden, zum Beispiel in einer Amazon Simple Queue Service-Warteschlange, sodass Schreibanfragen von Kunden auch dann akzeptiert werden können, wenn das primäre Gerät vorübergehend nicht verfügbar ist.

Ressourcen

Zugehörige Dokumente:

- Amazon API Gateway: Throttle API Requests for Better Throughput
- CircuitBreaker (Zusammenfassung des Unterbrechers aus dem Buch „Release It!“)
- Error Retries and Exponential Backoff in AWS
- Michael Nygard “Release It! Design and Deploy Production-Ready Software”
- Die Amazon Builders' Library: Vermeiden von Fallback in verteilten Systemen
- Die Amazon Builders' Library: Vermeiden von nicht mehr aufholbaren Warteschlangen-Rückständen
- Die Amazon Builders' Library: Herausforderungen und Strategien für das Caching
- Die Amazon Builders' Library: Timeouts, Wiederholungen und Backoff mit Jitter

Zugehörige Videos:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders’ Library (DOP328)

REL05-BP02 Drosselung von Anfragen

Drosseln Sie Anfragen, um eine Ressourcenüberlastung aufgrund eines unerwarteten Nachfrageanstiegs zu verringern. Anfragen, die unter der Drosselungsrate liegen, werden verarbeitet, während Anfragen, die über dem definierten Limit liegen, abgelehnt werden. Es wird eine Meldung zurückgegeben, die besagt, dass die Anfrage gedrosselt wurde.

Gewünschtes Ergebnis: Stark ansteigendes Volumen, das entweder durch plötzliche Anstiege des Kundendatenverkehrs, Flooding-Angriffe oder Wiederholungsstürme verursacht wird, wird durch Anfragedrosselung abgeschwächt, sodass Workloads die normale Verarbeitung des unterstützten Anforderungsvolumens fortsetzen können.

Typische Anti-Muster:

- API-Endpunktdrosselungen sind nicht implementiert oder werden auf Standardwerten belassen, ohne die erwarteten Volumina zu berücksichtigen.
- API-Endpunkte werden nicht ausgelastet oder die Drosselungsgrenzwerte werden nicht getestet.
- Anforderungsraten werden ohne Berücksichtigung der Größe oder Komplexität der Anfrage gedrosselt.
- Es werden sowohl die maximalen Anforderungsraten als auch die maximale Anforderungsgröße getestet, aber nicht beides zusammen.

- Ressourcen werden nicht mit denselben Limits bereitgestellt, die beim Testen festgelegt wurden.
- Es wurden keine Nutzungspläne konfiguriert oder für A2A-API-Verbraucher in Betracht gezogen.
- Für Warteschlangenverbraucher, die horizontal skalieren, sind keine Einstellungen für maximale Parallelität konfiguriert.
- Eine Ratenbegrenzung pro IP-Adresse wurde nicht implementiert.

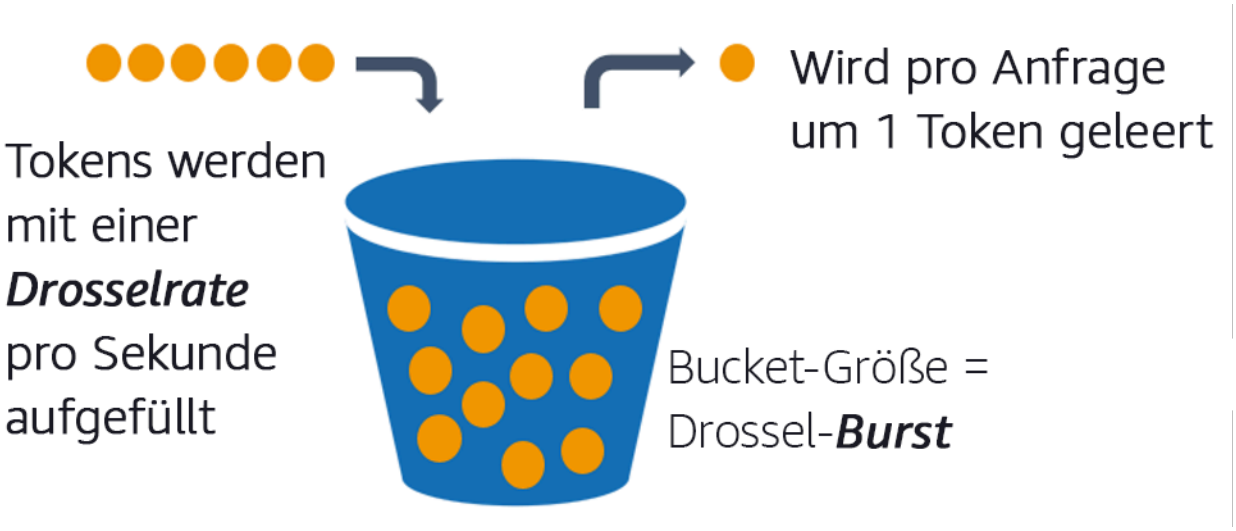
Vorteile der Nutzung dieser bewährten Methode: Workloads, die Drosselgrenzwerte festlegen, können normal arbeiten und akzeptierte Anfragen auch bei unerwarteten Volumenspitzen erfolgreich verarbeiten. Plötzliche oder anhaltende Spitzen von Anfragen an APIs und Warteschlangen werden gedrosselt und verbrauchen keine Ressourcen für die Anforderungsverarbeitung. Ratenbegrenzungen drosseln einzelne Anforderer, sodass ein hohes Datenverkehrsvolumen von einer einzelnen IP-Adresse oder einem API-Verbraucher keine Ressourcen verbraucht, die sich auf andere Verbraucher auswirken.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Services sollten so konzipiert sein, dass sie eine bekannte Kapazität von Anfragen verarbeiten. Diese Kapazität kann durch Auslastungstests ermittelt werden. Wenn die Anzahl der Anfragen die Grenzwerte überschreitet, signalisiert die entsprechende Antwort, dass eine Anfrage gedrosselt wurde. Dies ermöglicht es dem Verbraucher, den Fehler zu beheben und es später erneut zu versuchen.

Wenn für Ihren Service eine Drosselungsimplementierung erforderlich ist, sollten Sie die Implementierung des Token-Bucket-Algorithmus in Betracht ziehen, bei dem ein Token für eine Anfrage zählt. Tokens werden mit einer Drosselrate pro Sekunde aufgefüllt und asynchron um ein Token pro Anfrage geleert.



Der Token-Bucket-Algorithmus

Amazon API Gateway implementiert den Token-Bucket-Algorithmus entsprechend den Konto- und Regionslimits und kann pro Client mit Nutzungsplänen konfiguriert werden. Darüber hinaus können Amazon Simple Queue Service (Amazon SQS) und Amazon Kinesis Anfragen zwischenspeichern, um die Anforderungsrate auszugleichen, und höhere Drosselungsraten für Anfragen ermöglichen, die bearbeitet werden können. Schließlich können Sie die Ratenbegrenzung mit AWS WAF implementieren, um bestimmte API-Verbraucher zu drosseln, die ungewöhnlich hohe Lasten erzeugen.

Implementierungsschritte

Sie können API Gateway mit Drosselungslimits für Ihre APIs konfigurieren und `429 Too Many Requests`-Fehler zurückgeben, wenn Grenzwerte überschritten werden. Sie können AWS WAF zusammen mit Ihren AWS AppSync- und API Gateway-Endpunkten verwenden, um die Ratenbegrenzung pro IP-Adresse zu aktivieren. Wenn Ihr System asynchrone Verarbeitung toleriert, können Sie außerdem Nachrichten in eine Warteschlange oder einen Stream stellen, um die Antworten an Service-Clients zu beschleunigen und so höhere Drosselungsraten zu erreichen.

Wenn Sie Amazon SQS als Ereignisquelle für AWS Lambda konfiguriert haben, können Sie mit asynchroner Verarbeitung maximale Gleichzeitigkeit konfigurieren, um zu verhindern, dass hohe Ereignisraten die für andere Services in Ihrem Workload oder Konto benötigten Kontingente für gleichzeitige Ausführungen auf Kontoebene verbrauchen.

API Gateway bietet zwar eine verwaltete Implementierung des Token-Buckets, aber in Fällen, in denen Sie API Gateway nicht verwenden können, können Sie sprachspezifische Open-Source-

Implementierungen (siehe entsprechende Beispiele unter Ressourcen) des Token-Buckets für Ihre Services nutzen.

- Verstehen und konfigurieren Sie API Gateway-Drosselungslimits auf Kontoebene pro Region, API pro Phase und API-Schlüssel pro Nutzungsplanebene.
- Wenden Sie die AWS WAF-Regeln zur Ratenbegrenzung auf API Gateway- und AWS AppSync-Endpunkte an, um sich vor Flooding zu schützen und schädliche IPs zu sperren Regeln zur Ratenbegrenzung können auch für AWS AppSync-API-Schlüssel für A2A-Verbraucher konfiguriert werden.
- Überlegen Sie, ob Sie für AWS AppSync-APIs mehr Drosselungskontrolle als Ratenbegrenzung benötigen, und konfigurieren Sie in diesem Fall ein API Gateway vor Ihrem AWS AppSync-Endpunkt.
- Wenn Amazon SQS-Warteschlangen als Auslöser für Lambda-Warteschlangenverbraucher eingerichtet werden, legen Sie die maximale Gleichzeitigkeit auf einen Wert fest, mit dem genug verarbeitet wird, um Ihre Service-Level-Ziele zu erreichen, aber keine Gleichzeitigkeitsbeschränkungen ausnutzt werden, die sich auf andere Lambda-Funktionen auswirken. Erwägen Sie, die reservierte Gleichzeitigkeit für andere Lambda-Funktionen in demselben Konto und derselben Region festzulegen, wenn Sie Warteschlangen mit Lambda verbrauchen.
- Verwenden Sie API Gateway mit nativen Serviceintegrationen in Amazon SQS oder Kinesis, um Anfragen zwischenzuspeichern.
- Wenn Sie API Gateway nicht verwenden können, nutzen Sie sprachspezifische Bibliotheken, um den Token-Bucket-Algorithmus für Ihren Workload zu implementieren. Sehen Sie sich den Abschnitt mit den Beispielen an und recherchieren Sie selbst, um eine geeignete Bibliothek zu finden.
- Testen Sie Grenzwerte, die Sie festlegen oder deren Erhöhung Sie zulassen möchten, und dokumentieren Sie die getesteten Grenzwerte.
- Erhöhen Sie die Grenzwerte nicht über das hinaus, was Sie beim Testen festgelegt haben. Wenn Sie einen Grenzwert erhöhen, stellen Sie sicher, dass die bereitgestellten Ressourcen bereits denen in Testszenarien entsprechen oder diese übertreffen, bevor Sie die Erhöhung anwenden.

Ressourcen

Zugehörige bewährte Methoden:

- REL04-BP03 Arbeite ständig

- REL05-BP03 Steuerung und Begrenzung von Wiederholungsaufrufen

Zugehörige Dokumente:

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

Zugehörige Beispiele:

- The three most important AWS WAF rate-based rules
- Java Bucket4j
- Python-Token-Bucket
- Node-Token-Bucket
- .NET System Threading Rate Limiting

Zugehörige Videos:

- Implementing GraphQL API security best practices with AWS AppSync

Zugehörige Tools:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF
- Virtueller Warteraum auf AWS

REL05-BP03 Steuerung und Begrenzung von Wiederholungsaufrufen

Verwenden Sie das exponentielle Backoff, um Anfragen in zunehmend längeren Intervallen zwischen den einzelnen Wiederholungsversuchen zu wiederholen. Führen Sie Jitter zwischen den

Wiederholungen ein, um die Wiederholungsintervalle zufällig zu bestimmen. Beschränken Sie die maximale Anzahl an Wiederholungen.

Gewünschtes Ergebnis: Zu den typischen Komponenten eines verteilten Softwaresystems gehören Server, Load Balancer, Datenbanken und Server. DNS Während des normalen Betriebs können diese Komponenten auf Anfragen mit temporären oder begrenzten Fehlern sowie mit Fehlern antworten, die unabhängig von Wiederholungsversuchen dauerhaft bleiben würden. Wenn Clients Anfragen an Services stellen, verbrauchen die Anfragen Ressourcen wie Speicher, Threads, Verbindungen, Ports oder andere begrenzte Ressourcen. Die Steuerung und Einschränkung von Wiederholungsversuchen ist eine Strategie zur Freigabe und Minimierung des Ressourcenverbrauchs, sodass beanspruchte Systemkomponenten nicht überlastet werden.

Wenn Client-Anfragen eine Zeitüberschreitung oder Fehlerantworten erhalten, sollten sie entscheiden, ob sie es erneut versuchen möchten oder nicht. Wenn sie es erneut versuchen, tun sie dies mit exponentiellem Backoff mit Jitter und einem maximalen Wiederholungswert. Dadurch werden Backend-Services und -Prozesse entlastet und erhalten Zeit, um sich selbst zu reparieren, was zu einer schnelleren Wiederherstellung und einer erfolgreichen Bearbeitung von Anfragen führt.

Typische Anti-Muster:

- Wiederholungsversuche werden ohne exponentielles Backoff, Jitter und maximale Wiederholungswerte implementiert. Backoff und Jitter helfen dabei, künstliche Datenverkehrsspitzen zu vermeiden, die durch ungewollt koordinierte Wiederholungsversuche in regelmäßigen Intervallen entstehen.
- Implementierung von Wiederholungsversuchen, ohne deren Auswirkungen zu testen oder davon auszugehen, dass Wiederholungsversuche bereits in Szenarien integriert sind, und SDK ohne Wiederholungsversuche zu testen.
- Veröffentlichte Fehlercodes aus Abhängigkeiten werden nicht richtig interpretiert, was dazu führt, dass bei allen Fehlern eine Wiederholung versucht wird, auch dann, wenn die Ursache auf eine fehlende Berechtigung, einen Konfigurationsfehler oder ein anderes Problem hindeutet, das vorhersehbar nicht ohne manuelles Eingreifen behoben werden kann.
- Beobachtbarkeits-Praktiken, einschließlich der Überwachung und Meldung von Warnmeldungen bei wiederholten Serviceausfällen, damit die zugrunde liegenden Probleme bekannt werden und behoben werden können, werden nicht beachtet.
- Es werden benutzerdefinierte Wiederholungsmechanismen entwickelt, wenn integrierte Wiederholungsfunktionen oder Wiederholungsfunktionen von Drittanbietern ausreichen.

- Es werden Wiederholungsversuche auf mehreren Ebenen eines Anwendungsstapels auf eine Weise ausgeführt, die Wiederholungsversuche verstärkt, was die Ressourcen durch einen Wiederholungssturm weiter verbraucht. Vergewissern Sie sich, dass Sie verstehen, wie sich diese Fehler auf Ihre Anwendung und die Abhängigkeiten auswirken, auf die Sie sich verlassen, und führen Sie dann Wiederholungsversuche nur auf einer Ebene durch.
- Nicht idempotente Serviceaufrufe werden erneut versucht, was zu unerwarteten Nebeneffekten wie doppelten Ergebnissen führt.

Vorteile der Nutzung dieser bewährten Methode: Wiederholungsversuche helfen Clients dabei, die gewünschten Ergebnisse zu erzielen, wenn Anfragen fehlschlagen, verbrauchen aber auch mehr Zeit auf dem Server, um die gewünschten erfolgreichen Antworten zu erhalten. Wenn Fehler selten oder vorübergehend auftreten, funktionieren Wiederholungsversuche gut. Wenn Fehler durch Ressourcenüberlastung verursacht werden, können Wiederholungsversuche die Situation verschlimmern. Durch das Hinzufügen eines exponentiellen Backoffs mit Jitter zu den Client-Wiederholungsversuchen können Server sich erholen, wenn Ausfälle durch Ressourcenüberlastung verursacht werden. Jitter verhindert, dass Anfragen zu Datenverkehrsspitzen führen, und Backoff verringert die Lasteskalation, die durch das Hinzufügen von Wiederholungsversuchen zur normalen Anforderungslast verursacht wird. Schließlich ist es wichtig, eine maximale Anzahl von Wiederholungsversuchen oder die verstrichene Zeit zu konfigurieren, um zu vermeiden, dass Rückstände entstehen, die zu metastabilen Ausfällen führen.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Hoch

Implementierungsleitfaden

Steuern und begrenzen Sie Wiederholungsaufrufe. Verwenden Sie ein exponentielles Backoff, um Aufrufe nach zunehmend längeren Intervallen zu wiederholen. Nutzen Sie Jitter, um die Wiederholungsintervalle zu randomisieren, und legen Sie ein Limit für die Zahl der Wiederholungen fest.

Einige AWS SDKs implementieren standardmäßig Wiederholungsversuche und exponentielles Backoff. Verwenden Sie diese integrierten AWS Implementierungen, sofern dies für Ihren Workload relevant ist. Implementieren Sie eine ähnliche Logik in Ihrem Workload, wenn Sie Services aufrufen, die idempotent sind und bei denen Wiederholungsversuche die Verfügbarkeit Ihrer Clients verbessern. Legen Sie entsprechend Ihrem Anwendungsfall Zeitüberschreitungen fest und geben Sie an, wann Wiederholversuche gestoppt werden sollen. Erstellen Sie Testszenarien für diese Wiederholungsfälle und führen Sie sie aus.

Implementierungsschritte

- Ermitteln Sie die optimale Ebene in Ihrem Anwendungsstack, um Wiederholungsversuche für die Services zu implementieren, auf die sich Ihre Anwendung stützt.
- Seien Sie sich bewusstSDKs, dass es für die Sprache Ihrer Wahl bewährte Wiederholungsstrategien mit exponentiellem Backoff und Jitter gibt, und ziehen Sie diese dem Schreiben Ihrer eigenen Wiederholungsimplementierungen vor.
- Stellen Sie sicher, dass Services idempotent sind, bevor Sie Wiederholungen implementieren. Sobald Wiederholungsversuche implementiert wurden, stellen Sie sicher, dass sie sowohl getestet als auch regelmäßig in der Produktion ausgeführt werden.
- Verwenden Sie beim Aufrufen des AWS Dienstes die Option und und APIs machen Sie sich mit den Konfigurationsoptionen für AWS SDKsWiederholungen vertraut AWS CLI. Finden Sie heraus, ob die Standardeinstellungen für Ihren Anwendungsfall geeignet sind, testen Sie sie und passen Sie sie nach Bedarf an.

Ressourcen

Zugehörige bewährte Methoden:

- REL04-BP04 Festlegen von Mutationsoperationen als idempotent
- REL05-BP02 Drosselung von Anfragen
- REL05-BP04 Schnelles Scheitern und Begrenzen von Warteschlangen
- REL05-BP05 Client-Timeouts festlegen
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

Zugehörige Dokumente:

- Fehler bei Wiederholungen und exponentiellem Backoff in AWS
- Die Amazon Builders' Library: Timeouts, Wiederholungen und Backoff mit Jitter
- Exponential Backoff and Jitter
- Machen Sie Wiederholungen mit idempotent sicher APIs

Zugehörige Beispiele:

- Spring Retry

- Resilience4j Retry

Zugehörige Videos:

- Wiederholung, Backoff und Jitter: AWS re:Invent 2019: Einführung in die Amazon Builders' Library () DOP328

Zugehörige Tools:

- AWS SDKsund Tools: Verhalten bei Wiederholungen
- AWS Command Line Interface: Wiederholungen AWS CLI

REL05-BP04 Schnelles Scheitern und Begrenzen von Warteschlangen

Wenn ein Service nicht in der Lage ist, erfolgreich auf eine Anfrage zu antworten, sollte er schnell scheitern. Dies ermöglicht die Freigabe von mit einer Anfrage verbundenen Ressourcen und damit die Wiederherstellung eines Services, falls dieser nicht mehr über genügend Ressourcen verfügt. Schnelles Scheitern ist ein etabliertes Softwaredesignmuster, das genutzt werden kann, um hochzuverlässige Workloads in der Cloud aufzubauen. Warteschlangen sind ebenfalls ein etabliertes Integrationsmuster für Unternehmen. Sie sorgen für eine ausgeglichene Auslastung und ermöglichen es den Clients, Ressourcen freizugeben, wenn eine asynchrone Verarbeitung toleriert wird. Wenn ein Service unter normalen Bedingungen erfolgreich antworten kann, aber fehlschlägt, wenn die Anforderungsrate zu hoch ist, verwenden Sie eine Warteschlange, um Anfragen zwischenzuspeichern. Lassen Sie jedoch keine langen Warteschlangen zu. Sie können dazu führen, dass veraltete Anfragen verarbeitet werden, die ein Client bereits aufgegeben hat.

Gewünschtes Ergebnis: Wenn bei Systemen Ressourcenknappheit, Timeouts, Ausnahmen oder Grauausfälle auftreten, die Service-Level-Ziele unerreichbar machen, ermöglichen Strategien für schnelles scheitern eine schnellere Systemwiederherstellung. Systeme, die Traffic-Spitzen absorbieren müssen und asynchrone Verarbeitung ermöglichen, können die Zuverlässigkeit verbessern, indem sie es Clients ermöglichen, Anfragen schnell freizugeben, indem sie Warteschlangen verwenden, um Anfragen an Back-End-Services zu puffern. Beim Puffern von Anfragen in Warteschlangen werden Strategien zur Warteschlangenverwaltung implementiert, um nicht mehr aufzuholende Rückstände zu vermeiden.

Typische Anti-Muster:

- Implementierung von Nachrichtenwarteschlangen, aber keine Konfiguration von Warteschlangen für unzustellbare Nachrichten (DLQ) oder Alarmen für volle DLQs, um zu erkennen, wenn ein System ausfällt.
- Nichterfassung des Alters von Nachrichten in einer Warteschlange, einem Indikator für Latenz, um zu verstehen, wann Warteschlangenverbraucher mit der Verarbeitung nicht mehr hinterher kommen oder Fehler machen, was zu erneuten Versuchen führt.
- Kein Löschen von aufgestauten Nachrichten aus einer Warteschlange, wenn es keinen Sinn macht, diese Nachrichten zu verarbeiten, da kein Geschäftsbedarf mehr besteht.
- Die Konfiguration von First-in-First-Out (FIFO)-Warteschlangen, wenn Last-In-First-Out (LIFO)-Warteschlangen den Client-Anforderungen besser gerecht werden würden. Dies ist beispielsweise dann der Fall, wenn keine strenge Reihenfolge erforderlich ist und die Backlog-Verarbeitung alle neuen und zeitkritischen Anfragen verzögert, was dazu führt, dass alle Clients die Service-Levels nicht einhalten.
- Bereitstellung interner Warteschlangen für Clients, anstatt APIs verfügbar zu machen, die den Arbeitseingang verwalten und Anfragen in internen Warteschlangen platzieren.
- Wenn zu viele Arbeitsanforderungstypen in einer einzigen Warteschlange zusammengefasst werden, kann dies die Backlog-Bedingungen verschärfen, da der Ressourcenbedarf auf die verschiedenen Anforderungstypen verteilt wird.
- Verarbeitung komplexer und einfacher Anfragen in derselben Warteschlange, obwohl unterschiedliche Überwachungs-, Timeout- und Ressourcenzuweisungen erforderlich sind.
- Keine Validierung von Eingaben oder Nutzung von Aussagen, um Mechanismen für schnelles Scheitern in Software zu implementieren, die Ausnahmen an übergeordnete Komponenten weiterleiten, die Fehler problemlos verarbeiten können.
- Keine Entfernung fehlerhafter Ressourcen aus der Anforderungsweiterleitung, insbesondere bei Ausfällen ohne erkennbare Ursache mit sowohl erfolgreicher als auch fehlgeschlagener Verarbeitung aufgrund von Abstürzen und Neustarts, zeitweise auftretenden Abhängigkeitsfehlern, verringerter Kapazität oder Verlust von Netzwerkpaketen.

Vorteile der Nutzung dieser bewährten Methode: Systeme, die schnelles Scheitern nutzen, lassen sich leichter debuggen und korrigieren und weisen häufig Probleme im Code und in der Konfiguration auf, bevor Releases für die Produktion veröffentlicht werden. Systeme, die effektive Warteschlangenstrategien beinhalten, sind widerstandsfähiger und zuverlässiger bei Traffic-Spitzen und zeitweiligen Systemstörungen.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Hoch

Implementierungsleitfaden

Strategien für schnelles Scheitern können sowohl in Softwarelösungen als auch in der Infrastruktur konfiguriert werden. Warteschlangen scheitern nicht nur schnell, sondern sind auch eine einfache und dennoch leistungsstarke Architekturtechnik zur Entkopplung von Systemkomponenten für eine ausgeglichene Auslastung. Amazon CloudWatch bietet Funktionen zur Überwachung von Ausfällen und zur Warnung bei Ausfällen. Sobald erkannt wird, dass ein System ausfällt, können Strategien zur Schadensbegrenzung umgesetzt werden, darunter auch der Wechsel weg von knapp werdenden Ressourcen. Wenn in Systemen Warteschlangen mit Amazon SQS und anderen Warteschlangentechnologien implementiert werden, um eine ausgeglichene Auslastung zu gewährleisten, muss berücksichtigt werden, wie Warteschlangenrückstände sowie Fehler beim Nachrichtenabruf verwaltet werden können.

Implementierungsschritte

- Implementieren Sie programmatische Aussagen oder spezifische Metriken in Ihrer Software und verwenden Sie diese, um explizit Alarme bei Systemproblemen auszulösen. Amazon CloudWatch hilft Ihnen bei der Erstellung von Metriken und Alarmen auf der Grundlage des Anwendungsprotokollmusters und der SDK-Instrumentierung.
- Verwenden Sie CloudWatch-Metriken und Alarme, um knappe Ressourcen zu erkennen, die die Latenz bei der Verarbeitung erhöhen oder Anfragen wiederholt nicht bearbeiten können.
- Nutzen Sie asynchrone Verarbeitung, indem Sie APIs entwerfen, die Anfragen annehmen und an interne Warteschlangen anhängen. Verwenden Sie dazu Amazon SQS und senden Sie dann eine Erfolgsmeldung an den Nachrichten-Client, sodass der Client Ressourcen freigeben und mit anderen Arbeiten fortfahren kann, während die Verbraucher der Backend-Warteschlangen Anfragen verarbeiten.
- Messen und überwachen Sie die Latenz bei der Verarbeitung von Warteschlangen, indem Sie jedes Mal, wenn Sie eine Nachricht aus einer Warteschlange nehmen, eine CloudWatch-Metrik erstellen, indem Sie die aktuelle Uhrzeit mit dem Nachrichtenzeitstempel vergleichen.
- Wenn Fehler eine erfolgreiche Nachrichtenverarbeitung verhindern oder der Datenverkehr so stark ansteigt, dass er im Rahmen der Service Level Agreements nicht verarbeitet werden kann, wird älterer oder überschüssiger Datenverkehr in eine Überlaufwarteschlange ausgelagert. So können vorrangig neuere Aufträge verarbeitet werden. Ältere Aufträge werden verarbeitet, sobald Kapazitäten frei werden. Diese Technik ist eine Annäherung an die LIFO-Verarbeitung und ermöglicht eine normale Systemverarbeitung für alle neuen Aufträge.

- Verwenden Sie Warteschlangen für unzustellbare Nachrichten oder Redrive-Warteschlangen, um Nachrichten, die nicht verarbeitet werden können, aus dem Backlog an einen Ort zu verschieben, der später geprüft und verarbeitet werden kann.
- Versuchen Sie es entweder erneut oder, sofern dies tolerierbar ist, löschen Sie alte Nachrichten, indem Sie die tatsächliche Zeit mit dem Nachrichtenzeitstempel vergleichen und Nachrichten verwerfen, die für den anfragenden Client nicht mehr relevant sind.

Ressourcen

Zugehörige bewährte Methoden:

- REL04-BP02 Lose gekoppelte Abhängigkeiten implementieren
- REL05-BP02 Drosselung von Anfragen
- REL05-BP03 Steuerung und Begrenzung von Wiederholungsaufrufen
- REL06-BP02 Definieren und Berechnen von Metriken (Aggregierung)
- REL06-BP07 Überwachen der gesamten Nachverfolgung von Anfragen im System

Zugehörige Dokumente:

- Vermeiden von nicht mehr aufzuholenden Rückständen
- Fail Fast
- How can I prevent an increasing backlog of messages in my Amazon SQS queue?
- Elastic Load Balancing: Zonal Shift
- Amazon Application Recovery Controller: Routing-Steuerung für Datenverkehr-Failover

Zugehörige Beispiele:

- Enterprise Integration Patterns: Dead Letter Channel

Zugehörige Videos:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications

Zugehörige Tools:

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

REL05-BP05 Client-Timeouts festlegen

Legen Sie angemessene Zeitüberschreitungen für Verbindungen und Anfragen fest, überprüfen Sie sie systematisch und verlassen Sie sich nicht auf Standardwerte, da sie nicht Workload-spezifisch sind.

Gewünschtes Ergebnis: Client-Zeitüberschreitungen sollten die Kosten für Client, Server und Workload berücksichtigen, die mit dem Warten auf Anfragen verbunden sind, deren Bearbeitung ungewöhnlich lange dauert. Da es nicht möglich ist, die genaue Ursache einer Zeitüberschreitung zu ermitteln, müssen Clients ihr Wissen über Services nutzen, um Erwartungen hinsichtlich wahrscheinlicher Ursachen und geeigneter Zeitüberschreitungen zu entwickeln.

Bei Client-Verbindungen kommt es aufgrund der konfigurierten Werte zu einer Zeitüberschreitung. Nach einer Zeitüberschreitung entscheidet der Client entweder, die Anfrage abzubrechen und es erneut zu versuchen oder er öffnet einen Unterbrecher. Durch diese Muster wird vermieden, dass Anfragen gestellt werden, die einen zugrunde liegenden Fehlerzustand verschlimmern könnten.

Typische Anti-Muster:

- Systemzeitüberschreitungen oder standardmäßige Zeitüberschreitungen werden nicht beachtet.
- Normale Abschlusszeit für Anfragen ist nicht bekannt.
- Mögliche Ursachen, warum die Bearbeitung von Anfragen ungewöhnlich lange dauert, oder die Kosten für die Client-, Service- oder Workload-Leistung, die während des Wartens darauf, dass diese Anfragen abgeschlossen werden, anfallen, sind nicht bekannt.
- Die Wahrscheinlichkeit, dass ein gestörtes Netzwerk dazu führt, dass eine Anfrage erst dann fehlschlägt, wenn die Zeitüberschreitung erreicht ist, und die Kosten für die Client- und Workload-Leistung, die entstehen, wenn keine kürzere Zeitüberschreitung gewählt wird, sind nicht bekannt.
- Zeitüberschreitungsszenarien sowohl für Verbindungen als auch für Anfragen werden nicht getestet.
- Zu hohe Zeitüberschreitungen können zu langen Wartezeiten führen und die Ressourcenauslastung erhöhen.

- Zu niedrige Zeitüberschreitungen führen zu künstlichen Fehlschlägen.
- Muster zur Behandlung von Zeitüberschreitungsfehlern bei Remote-Aufrufen wie Unterbrecher und Wiederholungsversuchen werden übersehen.
- Die Überwachung der Fehlerraten bei Serviceaufrufen, der Service-Level-Ziele für die Latenz und der Latenzausreißer wird nicht in Betracht gezogen. Diese Metriken können Aufschluss über aggressive oder tolerante Zeitüberschreitungen geben.

Vorteile der Nutzung dieser bewährten Methode: Zeitüberschreitungen für Remote-Aufrufe sind konfiguriert und die Systeme sind so konzipiert, dass sie Zeitüberschreitungen ordnungsgemäß behandeln, sodass Ressourcen geschont werden, wenn Remote-Aufrufe ungewöhnlich langsam reagieren und Zeitüberschreitungsfehler von Service-Clients ordnungsgemäß behandelt werden.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Legen Sie eine Zeitüberschreitung für Verbindungen sowie Anfragen für alle Serviceabhängigkeitsaufrufe und generell für prozessübergreifende Aufrufe fest. Viele Frameworks bieten integrierte Zeitüberschreitungsfunktionen. Seien Sie jedoch vorsichtig, da einige Standardwerte unendlich oder höher als für Ihre Serviceziele akzeptabel sind. Ein zu hoher Wert reduziert die Nützlichkeit der Zeitbeschränkung, da Ressourcen weiterhin verbraucht werden, während der Client auf das Einsetzen der Zeitbeschränkung wartet. Ein zu niedriger Wert kann zu erhöhtem Datenverkehr im Backend und zu erhöhter Latenz führen, da zu viele Anfragen wiederholt werden. In einigen Fällen kann dies zu vollständigen Ausfällen führen, da alle Anfragen wiederholt werden.

Beachten Sie bei der Festlegung von Zeitüberschreitungsstrategien Folgendes:

- Die Bearbeitung von Anfragen kann aufgrund ihres Inhalts, Beeinträchtigungen eines Zieldienstes oder eines Ausfalls einer Netzwerkpartition länger als normal dauern.
- Anfragen mit ungewöhnlich aufwändigem Inhalt könnten unnötige Server- und Client-Ressourcen verbrauchen. In diesem Fall können Ressourcen geschont werden, wenn für diese Anfragen eine Zeitüberschreitung konfiguriert wird und es nicht erneut versucht wird. Services sollten sich auch durch Drosselungen und serverseitige Zeitüberschreitungen vor ungewöhnlich aufwändigen Inhalten schützen.
- Anfragen, die aufgrund einer Servicebeeinträchtigung ungewöhnlich lange dauern, können mit einer Zeitüberschreitung abgebrochen und erneut versucht werden. Die Servicekosten für die

Anfrage und den erneuten Versuch sollten berücksichtigt werden. Wenn die Ursache jedoch eine lokale Beeinträchtigung ist, ist ein erneuter Versuch wahrscheinlich nicht teuer und reduziert den Ressourcenverbrauch des Clients. Die Zeitüberschreitung kann je nach Art der Beeinträchtigung auch Serverressourcen freisetzen.

- Anfragen, deren Bearbeitung lange dauert, weil die Anfrage oder Antwort nicht vom Netzwerk zugestellt wurde, können mit einer Zeitüberschreitung abgebrochen und erneut versucht werden. Da die Anfrage oder Antwort nicht zugestellt wurde, würde sie unabhängig von der Länge der Zeitüberschreitung fehlschlagen. Durch eine Zeitüberschreitung werden in diesem Fall keine Serverressourcen, aber Client-Ressourcen freigegeben und die Workload-Leistung wird verbessert.

Nutzen Sie etablierte Entwurfsmuster wie Wiederholungsversuche und Schutzschalter, um Timeouts elegant zu handhaben und ausfallschnelle Ansätze zu unterstützen. AWS SDKsund AWS CLIermöglichen die Konfiguration von Verbindungs- und Anforderungs-Timeouts sowie von Wiederholungsversuchen mit exponentiellem Backoff und Jitter. AWS LambdaFunktionen unterstützen die Konfiguration von Timeouts. Damit können Sie Low-Code-Schutzschalter bauen AWS Step Functions, die die Vorteile der vorgefertigten Integrationen mit Diensten und nutzen. AWS SDKs AWS App Mesh Envoy bietet Funktionen für Zeitüberschreitungen und Unterbrecher an.

Implementierungsschritte

- Konfigurieren Sie Zeitüberschreitungen für Remote-Serviceaufrufe und nutzen Sie die integrierten sprachspezifischen Zeitüberschreitungsfunktionen oder Open-Source-Bibliotheken für Zeitüberschreitungen.
- Wenn Ihr Workload Aufrufe mit einem tätigt AWS SDK, finden Sie in der Dokumentation Informationen zur sprachspezifischen Timeout-Konfiguration.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++

- Wenn Sie AWS SDKs AWS CLI Or-Befehle in Ihrem Workload verwenden, konfigurieren Sie Standard-Timeout-Werte, indem Sie die AWS Konfigurationsstandardwerte für und festlegen. `connectTimeoutInMillis` `tlsNegotiationTimeoutInMillis`
- Wenden Sie Befehlszeilenoptionen `cli-read-timeout` an `cli-connect-timeout` und steuern Sie einmalige AWS CLI Befehle auf Dienste. AWS
- Überwachen Sie Remote-Serviceanfragen auf Zeitüberschreitungen und richten Sie Alarme für anhaltende Fehler ein, sodass Sie proaktiv mit Fehlerszenarien umgehen können.
- Implementieren Sie CloudWatch Metriken und CloudWatch Anomalieerkennung zu Anruffehlerraten, Service-Level-Zielen für Latenz und Latenzausreißer, um Einblicke in den Umgang mit übermäßig aggressiven oder toleranten Timeouts zu gewinnen.
- Konfigurieren Sie Zeitüberschreitungen für Lambda-Funktionen.
- APIGateway-Clients müssen bei der Behandlung von Timeouts ihre eigenen Wiederholungsversuche implementieren. APIGateway unterstützt ein Integrations-Timeout von 50 Millisekunden bis 29 Sekunden für Downstream-Integrationen und versucht es nicht erneut, wenn die Integration ein Timeout anfordert.
- Implementieren Sie das Unterbrecher-Muster, um zu vermeiden, dass Remote-Aufrufe getätigt werden, wenn Zeitüberschreitungen auftreten. Öffnen Sie die Leitung, um fehlschlagende Aufrufe zu vermeiden, und schließen Sie die Leitung, wenn die Aufrufe normal reagieren.
- Für containerbasierte Workloads können Sie die Funktionen von App Mesh Envoy nutzen, um von den integrierten Zeitüberschreitungen und Unterbrechern zu profitieren.
- Wird verwendet AWS Step Functions , um Low-Code-Schutzschalter für Remote-Serviceanrufe zu erstellen, insbesondere wenn AWS native SDKs und unterstützte Step Functions Functions-Integrationen aufgerufen werden, um Ihre Arbeitslast zu vereinfachen.

Ressourcen

Zugehörige bewährte Methoden:

- REL05-BP03 Steuerung und Begrenzung von Wiederholungsaufrufen
- REL05-BP04 Schnelles Scheitern und Begrenzen von Warteschlangen
- REL06-BP07 Überwachen der gesamten Nachverfolgung von Anfragen im System

Zugehörige Dokumente:

- AWS SDK: Wiederholungsversuche und Timeouts

- Die Amazon Builders' Library: Timeouts, Wiederholungen und Backoff mit Jitter
- Amazon API Gateway-Kontingente und wichtige Hinweise
- AWS Command Line Interface: Command line options
- AWS SDK for Java 2.x: API Timeouts konfigurieren
- AWS Botocore verwendet das Konfigurationsobjekt und die Konfigurationsreferenz
- AWS SDK für .NET: Retries and Timeouts
- AWS Lambda: Configuring Lambda function options

Zugehörige Beispiele:

- Verwenden des Circuit Breaker Patterns mit AWS Step Functions Amazon DynamoDB
- Martin Fowler: CircuitBreaker

Zugehörige Tools:

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

REL05-BP06 Systeme soweit möglich zustandslos machen

Systeme sollten entweder keinen Zustand erfordern oder ihn so auslagern, dass zwischen verschiedenen Client-Anfragen keine Abhängigkeit von lokal gespeicherten Daten auf der Festplatte und im Arbeitsspeicher besteht. Auf diese Weise können Server nach Belieben ersetzt werden, ohne dass dies Auswirkungen auf die Verfügbarkeit hat.

Wenn Benutzer oder Services mit einer Anwendung interagieren, führen sie häufig eine Reihe von Interaktionen aus, die eine Sitzung bilden. Bei einer Sitzung handelt es sich um eindeutige Daten für Benutzer, die zwischen Anfragen bestehen bleiben, während sie die Anwendung verwenden. Eine zustandslose Anwendung ist eine Anwendung, die keine Informationen zu früheren Interaktionen benötigt und keine Sitzungsinformationen speichert.

Sobald das System so konzipiert ist, dass es zustandslos ist, können Sie serverlose Rechendienste wie oder verwenden. AWS Lambda AWS Fargate

Neben dem Austausch von Servern besteht ein weiterer Vorteil statusfreier Anwendungen darin, dass sie horizontal skaliert werden können, da alle verfügbaren Rechenressourcen (wie EC2 Instanzen und AWS Lambda Funktionen) jede Anforderung bearbeiten können.

Vorteile der Nutzung dieser bewährten Methode: Systeme mit zustandslosem Design lassen sich besser an die horizontale Skalierung anpassen, sodass Kapazitäten je nach vorhandenem Datenverkehr und bestehender Nachfrage hinzugefügt oder entfernt werden können. Sie sind auch inhärent widerstandsfähig gegenüber Ausfällen und bieten Flexibilität und Agilität bei der Anwendungsentwicklung.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Erstellen Sie zustandslose Anwendungen. Zustandslose Anwendungen ermöglichen eine horizontale Skalierung und sind widerstandsfähig gegenüber Ausfällen einzelner Knoten. Analysieren Sie die Komponenten Ihrer Anwendung, die ihren Status innerhalb der Architektur beibehalten. Auf diese Weise können Sie die potenziellen Auswirkungen der Umstellung auf ein zustandsloses Design bewerten. Eine zustandslose Architektur entkoppelt Benutzerdaten und entlädt die Sitzungsdaten. Dies bietet die Flexibilität, jede Komponente unabhängig zu skalieren, um unterschiedlichen Workload-Anforderungen gerecht zu werden und die Ressourcenauslastung zu optimieren.

Implementierungsschritte

- Identifizieren und analysieren Sie die zustandsbehafteten Komponenten in Ihrer Anwendung.
- Entkoppeln Sie Daten, indem Sie Benutzerdaten von der Kernanwendungslogik trennen und verwalten.
  - Amazon Cognito kann Benutzerdaten mithilfe von Features wie Identitätspools, Benutzerpools und Amazon Cognito Sync von Anwendungscode entkoppeln.
  - Sie können AWS Secrets Manager verwenden, um Benutzerdaten zu entkoppeln, indem Sie Secrets an einem sicheren, zentralen Ort speichern. Das bedeutet, dass der Anwendungscode keine Secrets speichern muss, was seine Sicherheit erhöht.
  - Erwägen Sie die Verwendung von Amazon S3, um große, unstrukturierte Daten wie Bilder und Dokumente zu speichern. Ihre Anwendung kann diese Daten bei Bedarf abrufen, sodass sie nicht im Arbeitsspeicher gespeichert werden müssen.

  - Verwenden Sie Amazon DynamoDB, um Informationen wie Benutzerprofile zu speichern. Ihre Anwendung kann diese Daten nahezu in Echtzeit abfragen.
- Verlagern Sie Sitzungsdaten in eine Datenbank, einen Cache oder externe Dateien.
  - Amazon ElastiCache, Amazon DynamoDB, Amazon Elastic File System (AmazonEFS) und Amazon MemoryDB sind Beispiele für AWS Dienste, die Sie zum Auslagern von Sitzungsdaten verwenden können.
- Entwerfen Sie eine zustandslose Architektur, nachdem Sie festgelegt haben, welche Zustands- und Benutzerdaten in Ihrer bevorzugten Speicherlösung abgelegt werden müssen.

Ressourcen

Zugehörige bewährte Methoden:

- REL11-BP03 Automatisieren Sie die Heilung auf allen Ebenen

Zugehörige Dokumente:

- Die Amazon Builders' Library: Vermeiden von Fallback in verteilten Systemen
- Die Amazon Builders' Library: Vermeiden von nicht mehr aufholbaren Warteschlangen-Rückständen
- Die Amazon Builders' Library: Herausforderungen und Strategien für das Caching
- Bewährte Methoden für Stateless Web Tier auf AWS

REL05-BP07 Nothebel einbauen

Nothebel sind schnelle Prozesse, die die Auswirkungen auf die Verfügbarkeit Ihres Workloads mindern können.

Nothebel bewirken, dass das Verhalten von Komponenten oder Abhängigkeiten mithilfe bekannter und getesteter Mechanismen deaktiviert, gedrosselt oder geändert wird. Dadurch können Beeinträchtigungen des Workloads, die durch die Erschöpfung von Ressourcen aufgrund unerwarteter Nachfragesteigerungen verursacht werden, gemildert und die Auswirkungen von Ausfällen bei nicht kritischen Komponenten innerhalb Ihres Workloads reduziert werden.

Gewünschtes Ergebnis: Durch die Implementierung von Nothebeln können Sie bewährte Prozesse einrichten, um die Verfügbarkeit kritischer Komponenten in Ihrem Workload aufrechtzuerhalten.

Der Workload sollte sich problemlos reduzieren lassen und auch während der Aktivierung eines Nothebels weiterhin seine geschäftskritischen Funktionen ausführen. Weitere Informationen zur graziösen Degradation finden Sie unter REL05-BP01 Implementieren Sie eine graziöse Degradation, um entsprechende harte Abhängigkeiten in weiche Abhängigkeiten umzuwandeln.

Typische Anti-Muster:

- Der Ausfall von nicht kritischen Abhängigkeiten wirkt sich auf die Verfügbarkeit Ihres Kern-Workloads aus.
- Das Verhalten kritischer Komponenten wird während der Beeinträchtigung unkritischer Komponenten nicht getestet oder überprüft.
- Es sind keine klaren und deterministischen Kriterien für die Aktivierung oder Deaktivierung eines Nothebels definiert.

Vorteile der Nutzung dieser bewährten Methode: Die Implementierung von Nothebeln kann die Verfügbarkeit der kritischen Komponenten Ihres Workloads verbessern, indem Ihre Resolver mit bewährten Prozessen ausgestattet werden, um auf unerwartete Nachfragespitzen oder Ausfälle von nicht kritischen Abhängigkeiten zu reagieren.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Ermitteln Sie die kritischen Komponenten in Ihrem Workload.
- Entwerfen und gestalten Sie die kritischen Komponenten Ihres Workloads so, dass sie Ausfällen von nicht kritischen Komponenten standhalten.
- Führen Sie Tests durch, um das Verhalten Ihrer kritischen Komponenten beim Ausfall von nicht kritischen Komponenten zu überprüfen.
- Definieren und überwachen Sie relevante Metriken oder Auslöser für die Einleitung von Nothebeln.
- Definieren Sie die Verfahren (manuell oder automatisiert), die Bestandteil des Nothebels sind.

Implementierungsschritte

- Ermitteln Sie die kritischen Komponenten in Ihrem Workload.
  - Jede technische Komponente Ihres Workloads sollte der entsprechenden Geschäftsfunktion zugeordnet und als kritisch oder nicht kritisch eingestuft werden. Beispiele für wichtige

und unkritische Funktionen bei Amazon finden Sie unter Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second.

  - Hierbei handelt es sich sowohl um eine technische als auch um eine geschäftliche Entscheidung, die je nach Organisation und Workload unterschiedlich ausfallen kann.
- Entwerfen und gestalten Sie die kritischen Komponenten Ihres Workloads so, dass sie Ausfällen von nicht kritischen Komponenten standhalten.
  - Berücksichtigen Sie bei der Abhängigkeitsanalyse alle potenziellen Fehlermodi und stellen Sie sicher, dass Ihre Notfallmechanismen die kritischen Funktionen an nachgelagerte Komponenten weitergeben.
- Führen Sie Tests durch, um das Verhalten Ihrer kritischen Komponenten bei der Aktivierung Ihrer Nothebel zu überprüfen.
  - Vermeiden Sie bimodales Verhalten. Weitere Informationen finden Sie unter REL11-BP05 Verwenden Sie statische Stabilität, um bimodales Verhalten zu verhindern.
- Definieren und überwachen Sie relevante Metriken und lassen Sie gegebenenfalls einen Alarm auslösen, um einen Nothebel einzuleiten.
  - Die richtigen Metriken zur Überwachung zu finden, hängt von Ihrem Workload ab. Einige Beispielmetriken sind die Latenzzeit oder die Anzahl der fehlgeschlagenen Anfragen an eine Abhängigkeit.
- Definieren Sie die manuellen oder automatisierten Verfahren, die Bestandteil des Nothebels sind.
  - Dazu können Mechanismen wie Lastabwurf, Drosselung von Anfragen oder die Implementierung einer ordnungsgemäßen Funktionsminderung gehören.

Ressourcen

Zugehörige bewährte Methoden:

- REL05-BP01 Implementieren Sie eine schrittweise Degradation, um anwendbare harte Abhängigkeiten in weiche Abhängigkeiten umzuwandeln
- REL05-BP02 Drosseln Sie Anfragen
- REL11-BP05 Verwenden Sie statische Stabilität, um bimodales Verhalten zu verhindern

Zugehörige Dokumente:

- Automating safe, hands-off deployments

- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second

Zugehörige Videos:

- AWS re:Invent 2020: Zuverlässigkeit, Konsistenz und Vertrauen durch Unveränderlichkeit

Änderungsmanagement

Änderungen an Ihrer Workload oder der Umgebung müssen vorausgesehen und berücksichtigt werden, um einen zuverlässigen Betrieb der Workload zu erreichen. Zu diesen Änderungen gehören z. B. Bedarfsspitzen sowie interne Änderungen wie Featurebereitstellungen und Sicherheitspatches, die sich auf Ihre Workloads auswirken.

In den folgenden Abschnitten werden die bewährten Methoden für die Änderungsverwaltung erläutert.

Themen

Überwachen von Workload-Ressourcen

Protokolle und Metriken sind leistungsstarke Tools, mit denen Sie sich einen Überblick über den Zustand Ihrer Workload verschaffen können. Sie können Ihre Workload so konfigurieren, dass Protokolle und Metriken überwacht und Benachrichtigungen gesendet werden, wenn Schwellenwerte überschritten werden oder wichtige Ereignisse auftreten. Dank der Überwachung kann die Workload erkennen, wenn Schwellenwerte für eine niedrige Leistung unterschritten werden oder Ausfälle auftreten, sodass als Reaktion drauf eine automatische Wiederherstellung erfolgen kann.

Die Überwachung ist wichtig, um sicherzustellen, dass Sie Ihre Verfügbarkeitsanforderungen erfüllen. Ausschlaggebend ist eine effektive Fehlererkennung. Die größte Herausforderung sind nicht angezeigte Fehler, bei denen die Funktionalität nicht mehr gegeben ist, was aber nur indirekt erkennbar ist. Ihre Kunden stellen dies vor Ihnen fest. Zu den vorrangigen Zwecken der Überwachung zählt, dass Sie bei Problemen benachrichtigt werden. Alarme sollten so weit wie möglich von Ihren Systemen entkoppelt werden. Wenn aufgrund einer Serviceunterbrechung keine Benachrichtigungen mehr gesendet können, verzögert sich die Behebung.

Bei AWS instrumentieren wir unsere Anwendungen auf mehreren Ebenen. Wir erfassen die Latenz, die Fehlerraten und die Verfügbarkeit für die einzelnen Anfragen, für alle Abhängigkeiten und für wichtige Vorgänge innerhalb des Prozesses. Außerdem erfassen wir Metriken zu den wichtigsten Vorgängen. Damit können wir drohende Probleme noch vor ihrem Auftreten erkennen. Wir berücksichtigen nicht nur die durchschnittliche Latenz. Wir konzentrieren uns noch genauer auf

Latenz-Ausreißer wie das 99,9. und 99,99. Perzentil. Denn selbst wenn nur eine Anfrage von 1 000 oder 10 000 langsam verarbeitet wird, ist das eine schlechte Leistung. Wenn der Durchschnittswert in Ordnung ist, aber eine von 100 Anfragen bei wachsendem Datenverkehr eine extreme Latenz verursacht, wird sich dies letztlich zu einem Problem entwickeln.

Die Überwachung bei AWS besteht aus den folgenden fünf spezifischen Phasen:

1. Generierung – Überwachen aller Komponenten für die Workload
2. Aggregierung – Definieren und Berechnen von Metriken
3. Verarbeitung und Benachrichtigung in Echtzeit – Senden von Benachrichtigungen und Automatisieren von Antworten
4. Speicher und Analytik

Bewährte Methoden

- REL06-BP01 Überwachen aller Komponenten der Workload (Generierung)
- REL06-BP02 Definieren und Berechnen von Metriken (Aggregierung)
- REL06-BP03 Senden von Benachrichtigungen (Verarbeitung und Benachrichtigung in Echtzeit)
- REL06-BP04 Automatisieren von Antworten (Verarbeitung und Benachrichtigung in Echtzeit)
- REL06-BP05 Analysieren von Protokollen
- REL06-BP06 Regelmäßiges Durchführen von Prüfungen von Umfang und Metriken
- REL06-BP07 Überwachen der gesamten Nachverfolgung von Anfragen im System

REL06-BP01 Überwachen aller Komponenten der Workload (Generierung)

Überwachen Sie die Komponenten der Workload mit Amazon CloudWatch oder Tools von Drittanbietern. Überwachen Sie AWS-Services mit dem AWS Health Dashboard.

Alle Komponenten Ihres Workloads sollten überwacht werden, einschließlich Frontend, Geschäftslogik und Speicherstufen. Definieren Sie Schlüsselmetriken, beschreiben Sie, wie Sie diese gegebenenfalls aus Protokollen extrahieren, und legen Sie Schwellenwerte für das Auslösen entsprechender Alarmereignisse fest. Stellen Sie sicher, dass die Metriken für die wichtigen Leistungskennzahlen (KPIs) Ihrer Workload relevant sind und verwenden Sie Metriken und Protokolle, um frühe Warnzeichen einer Serviceverschlechterung zu identifizieren. Beispielsweise kann eine mit Geschäftsergebnissen zusammenhängende Metrik wie etwa die Anzahl der pro Minute erfolgreich verarbeiteten Bestellungen schneller auf Workload-Probleme hinweisen als eine

technische Metrik wie etwa die CPU-Auslastung. Verwenden Sie das AWS Health Dashboard für eine personalisierte Ansicht der Leistung und Verfügbarkeit der AWS-Services, die Ihren AWS-Ressourcen.

Die Überwachung in der Cloud bietet neue Möglichkeiten. Die meisten Cloudanbieter haben anpassbare Hooks entwickelt und können Einblicke liefern, mit denen Sie mehrere Ebenen Ihrer Workload überwachen können. AWS-Services wie Amazon CloudWatch wenden statistische und Machine-Learning-Algorithmen an, um Metriken von Systemen und Anwendungen kontinuierlich zu analysieren, normale Basiswerte zu erkennen und Oberflächenanomalien anhand eines minimalen Benutzereingriffs aufzudecken. Anomalieerkennungsalgorithmen berücksichtigen saisonale und trendbasierte Änderungen von Metriken.

AWS stellt zahlreiche Überwachungs- und Protokollinformationen bereit, die genutzt werden können, um workload-spezifische Metriken und Bedarfsänderungsprozesse zu definieren und Machine-Learning-Verfahren unabhängig von der ML-Erfahrung einzuführen.

Zudem können Sie auch all Ihre externen Endpunkte überwachen, um sicherzustellen, dass diese von Ihrer Basisimplementierung unabhängig sind. Diese aktive Überwachung kann anhand von synthetischen Transaktionen erfolgen (auch Benutzer-Canaries genannt, jedoch nicht zu verwechseln mit Canary-Bereitstellungen). Diese führen regelmäßig eine Reihe gängiger Aufgaben aus, die mit Aktionen übereinstimmen, die von Clients der Workload durchgeführt werden. Diese Aufgaben sollten nicht zu lang sein und Sie sollten darauf achten, Ihre Workload beim Testen nicht zu überlasten. Mit Amazon CloudWatch Synthetics können Sie synthetische Canaries erstellen, um Ihre Endpunkte und APIs zu überwachen. Sie können die synthetischen Canary-Client-Knoten auch mit der AWS X-Ray-Konsole kombinieren, um zu bestimmen, bei welchen synthetischen Canaries im ausgewählten Zeitraum Probleme mit Fehlern, Störungen oder Drosselungsraten auftreten.

Gewünschtes Ergebnis:

Erfassen und Nutzen kritischer Metriken aus allen Komponenten der Workload, um die Workload-Zuverlässigkeit und eine optimale Benutzererfahrung sicherzustellen. Wenn Sie erkennen, dass mit einem Workload keine Geschäftsergebnisse erzielt werden, können Sie schnell einen Systemausfall deklarieren und das System nach einem Vorfall wiederzustellen.

Typische Anti-Muster:

- Es werden nur externe Schnittstellen zum Workload überwacht.
- Es werden keine workload-spezifischen Metriken erzeugt und Sie verlassen sich nur auf Metriken, die Ihnen von den AWS-Services, die Ihre Workload verwendet, bereitgestellt werden.

- Es werden nur technische Metriken in Ihrer Workload verwendet und es werden keinerlei Metriken im Zusammenhang mit nicht-technischen KPIs, zu denen die Workload beiträgt, überwacht.
- Sie verlassen sich auf den Produktionsdatenverkehr und einfache Zustandsprüfungen für die Überwachung und Bewertung des Workload-Status.

Vorteile der Nutzung dieser bewährten Methode: Durch die Überwachung aller Ebenen Ihrer Workload können Sie Probleme in den darin enthaltenen Komponenten schneller vorhersehen und beheben.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

1. Aktivieren Sie die Protokollierung, wann immer verfügbar. Von allen Workload-Komponenten sollten Überwachungsdaten erzielt werden. Aktivieren Sie eine zusätzliche Protokollierung, wie etwa S3 Access Logs, und ermöglichen Sie es Ihrer Workload, die workload-spezifischen Daten zu protokollieren. Erfassen Sie Metriken für die Durchschnittswerte zu CPU, Netzwerk-E/A und Laufwerk-E/A von Services wie Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling und Amazon EMR Unter AWS Services That Publish CloudWatch Metrics finden Sie eine Liste an AWS-Services, die Metriken in CloudWatch veröffentlichen.
2. Sehen Sie sich alle Standardmetriken an, um mehr über mögliche Datenerfassungslücken zu erfahren. Jeder Service generiert Standardmetriken. Durch die Erfassung von Standardmetriken erhalten Sie ein besseres Verständnis über die Abhängigkeiten zwischen Workload-Komponenten und darüber, wie die Komponentenzuverlässigkeit und -leistung die Workload beeinträchtigen. Sie können auch Ihre eigenen Metriken in CloudWatch unter Verwendung der AWS CLI oder einer API erstellen und veröffentlichen.
3. Bewerten Sie alle Metriken, um zu entscheiden, für welche eine Warnmeldung für jeden AWS-Service in Ihrer Workload eingerichtet werden soll. Sie können eine Metriken-Untergruppe auswählen, die eine höhere Auswirkung auf die Workload-Zuverlässigkeit hat. Wenn Sie sich auf kritische Metriken und Schwellenwerte konzentrieren, können Sie die Anzahl an Warnmeldungen genauer definieren und so Falschmeldungen reduzieren.
4. Definieren Sie Warnungen und den Wiederherstellungsprozess für Ihre Workload nach dem Auslösen der Warnmeldung. Das Definieren von Warnmeldungen ermöglicht es Ihnen, schnell zu benachrichtigen, zu eskalieren und die für die Wiederherstellung nach einem Vorfall erforderlichen Schritte durchzuführen, um so Ihren festgelegten Recovery Time Objective (RTO) zu erfüllen. Sie können Amazon CloudWatch-Alarme für das Aufrufen von automatisierten Workflows und

die Initiierung von Wiederherstellungsverfahren basierend auf definierten Schwellenwerten verwenden.

5. Erfahren Sie mehr über die Verwendung von synthetischen Transaktionen für das Erfassen relevanter Daten zum Workload-Status. Die synthetische Überwachung folgt denselben Routen und führt dieselben Aktionen aus wie ein Kunde. Dadurch haben Sie die Möglichkeit, die Kundenerfahrung kontinuierlich zu überprüfen, selbst, wenn Sie keinen Kundendatenverkehr auf Ihren Workloads haben. Durch die Verwendung von synthetischen Transaktionen können Sie Probleme erkennen, bevor Ihre Kunden dies tun.

Ressourcen

Zugehörige bewährte Methoden:

- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen

Zugehörige Dokumente:

- Getting started with your AWS Health Dashboard – Your account health
- AWS Services That Publish CloudWatch Metrics
- Access Logs for Your Network Load Balancer
- Access logs for your application load balancer
- Accessing Amazon CloudWatch Logs for AWS Lambda
- Amazon-S3-Server-Zugriffsprotokollierung
- Enable Access Logs for Your Classic Load Balancer
- Exporting log data to Amazon S3
- Install the CloudWatch agent on an Amazon EC2 instance
- Publishing Custom Metrics
- Using Amazon CloudWatch Dashboards
- Verwenden von Amazon-CloudWatch-Metriken
- Using Canaries (Amazon CloudWatch Synthetics)
- What are Amazon CloudWatch Logs?

Benutzerhandbücher:

- Creating a trail
- Monitoring memory and disk metrics for Amazon EC2 Linux instances
- Using CloudWatch Logs with container instances
- VPC Flow Logs
- What is Amazon DevOps Guru?
- Was ist AWS X-Ray?

Verwandte Blogs:

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray

Zugehörige Beispiele:

- The Amazon Builders' Library: Instrumentieren verteilter Systeme für Einblicke in die Betriebsabläufe
- Workshop zur Beobachtbarkeit

REL06-BP02 Definieren und Berechnen von Metriken (Aggregierung)

Erfassen Sie Metriken und Protokolle aus Ihren Workload-Komponenten und berechnen Sie daraus relevante aggregierte Metriken. Diese Metriken bieten eine umfassende und tiefgehende Beobachtbarkeit über Ihre Workloads und können Ihre Resilienzlage erheblich verbessern.

Beobachtbarkeit bedeutet mehr als nur das Erfassen von Metriken aus Workload-Komponenten und die Möglichkeit, diese einzusehen und entsprechende Warnmeldungen zu erstellen. Es geht vielmehr darum, ein ganzheitliches Verständnis des Verhaltens Ihrer Workloads zu erhalten. Diese Verhaltensinformationen stammen aus allen Komponenten Ihrer Workloads, einschließlich der Cloud-Services, von denen sie abhängen, gut ausgearbeiteten Protokolle und Metriken. Diese Daten geben Ihnen einen Überblick über das Verhalten Ihrer Workloads insgesamt sowie ein detailliertes Verständnis der Interaktion jeder Komponente mit jeder Arbeitseinheit.

Gewünschtes Ergebnis:

- Sie erfassen Protokolle Ihrer Workload-Komponenten und AWS-Serviceabhängigkeiten und veröffentlichen sie an einem zentralen Ort, wo sie leicht abgerufen und verarbeitet werden können.
- Ihre Protokolle enthalten äußerst präzise Zeitstempel.

- Ihre Protokolle enthalten relevante Informationen über den Verarbeitungskontext, wie z. B. eine Ablaufverfolgungs-ID, eine Benutzer- oder Kontokennung und eine Remote-IP-Adresse.
- Sie erstellen aus Ihren Protokollen aggregierte Metriken, die das Verhalten Ihrer Workloads aus einer übergeordneten Perspektive darstellen.
- Sie können Ihre aggregierten Protokolle abfragen, um tiefe und relevante Einblicke in Ihre Workloads zu gewinnen und tatsächliche und potenzielle Probleme zu identifizieren.

Typische Anti-Muster:

- Sie erfassen keine relevanten Protokolle oder Metriken von den Computing-Instanzen, auf denen Ihre Workloads ausgeführt werden, oder von den Cloud-Services, die sie verwenden.
- Sie übersehen die Erfassung von Protokollen und Metriken, die Ihre Leistungskennzahlen (KPIs) betreffen.
- Sie analysieren die auf Workloads bezogene Telemetrie isoliert, ohne Aggregation und Korrelation.
- Sie lassen zu, dass Metriken und Protokolle zu schnell ablaufen, was Trendanalysen und die Identifizierung wiederkehrender Probleme erschwert.

Vorteile der Nutzung dieser bewährten Methoden: Sie können mehr Anomalien erkennen und Ereignisse und Metriken zwischen verschiedenen Komponenten Ihrer Workloads korrelieren. Sie können anhand Ihrer Workload-Komponenten Erkenntnisse gewinnen, die auf Informationen in Protokollen basieren, die häufig nicht nur in Form von Metriken verfügbar sind. Sie können Fehlerursachen schneller ermitteln, indem Sie Ihre Protokolle in großem Umfang abfragen.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Hoch

Implementierungsleitfaden

Identifizieren Sie die Quellen von Telemetriedaten, die für Ihre Workloads und deren Komponenten relevant sind. Diese Daten stammen nicht nur aus Komponenten, die Metriken veröffentlichen, wie Ihrem Betriebssystem (OS) und Anwendungslaufzeiten wie Java, sondern auch aus Anwendungs- und Cloud-Serviceprotokollen. Beispielsweise protokollieren Webserver in der Regel jede Anfrage mit detaillierten Informationen wie Zeitstempel, Verarbeitungslatenz, Benutzer-ID, Remote-IP-Adresse, Pfad und Abfragezeichenfolge. Der Detaillierungsgrad dieser Protokolle hilft Ihnen dabei, detaillierte Abfragen durchzuführen und Metriken zu generieren, die sonst möglicherweise nicht verfügbar gewesen wären.

Erfassen Sie die Metriken und Protokolle mithilfe geeigneter Tools und Prozesse. Protokolle, die von Anwendungen generiert werden, die auf einer Amazon-EC2-Instance ausgeführt werden, können von einem Agenten wie dem Amazon CloudWatch Agent erfasst und in einem zentralen Speicherservice wie Amazon CloudWatch Logs veröffentlicht werden. AWS-verwaltete Computing-Services wie AWS LambdaAmazon Elastic Container Service veröffentlichen Protokolle automatisch in CloudWatch Logs für Sie. Aktivieren Sie die Protokollerfassung für AWS-Speicher- und Verarbeitungsservices, die von Ihren Workloads wie Amazon CloudFront, Amazon S3, Elastic Load Balancing und Amazon API Gateway verwendet werden.

Reichern Sie Ihre Telemetriedaten mit Dimensionen an, anhand derer Sie Verhaltensmuster klarer erkennen und korrelierte Probleme anhand von Gruppen verwandter Komponenten isolieren können. Nach dem Hinzufügen können Sie das Verhalten der Komponenten detaillierter beobachten, korrelierte Fehler erkennen und geeignete Abhilfemaßnahmen ergreifen. Beispiele für nützliche Dimensionen sind Availability Zone, EC2-Instance-ID und Container-Task- oder Pod-ID.

Sobald Sie die Metriken und Protokolle erfasst haben, können Sie Abfragen erstellen und daraus aggregierte Metriken generieren, die nützliche Einblicke in normales und anomales Verhalten bieten. Sie können beispielsweise Amazon CloudWatch Logs Insights verwenden, um benutzerdefinierte Metriken aus Ihren Anwendungsprotokollen abzuleiten, Amazon CloudWatch Metrics Insights, um Ihre Metriken in großem Umfang abzufragen, Amazon CloudWatch Container Insights zum Sammeln, Aggregieren und Zusammenfassen von Metriken und Protokollen aus Ihren containerisierten Anwendungen und Microservices oder Amazon CloudWatch Lambda Insights, wenn Sie AWS Lambda-Funktionen verwenden. Um eine aggregierte Fehlerratenmetrik zu erstellen, können Sie einen Zähler jedes Mal erhöhen, wenn eine Fehlerantwort oder Meldung in Ihren Komponentenprotokollen gefunden wird, oder Sie können den Gesamtwert einer vorhandenen Fehlerratenmetrik berechnen. Sie können diese Daten verwenden, um Histogramme zu generieren, die das Verhalten von Vorgängen zeigen, z. B. die Anfragen oder Prozesse mit der schlechtesten Leistung. Mithilfe von Lösungen wie der Anomalieerkennung von CloudWatch Logs können Sie diese Daten auch in Echtzeit nach anomalen Mustern durchsuchen. Diese Erkenntnisse können in Dashboards platziert werden, sodass sie Ihren Anforderungen und Präferenzen entsprechend organisiert werden können.

Mithilfe von Protokollabfragen können Sie besser verstehen, wie bestimmte Anfragen von Ihren Workload-Komponenten bearbeitet wurden, und Anforderungsmuster oder andere Kontexte aufdecken, die sich auf die Resilienz Ihres Workloads auswirken. Es kann nützlich sein, Abfragen auf der Grundlage Ihres Wissens über das Verhalten Ihrer Anwendungen und anderer Komponenten im Voraus zu recherchieren und vorzubereiten, so dass Sie sie bei Bedarf einfacher ausführen können. Beispielsweise können Sie mit CloudWatch Logs Insights Protokolldaten in Amazon CloudWatch

Logs interaktiv durchsuchen und analysieren. Sie können auch Amazon Athena verwenden, um Protokolle aus mehreren Quellen, einschließlich zahlreicher AWS-Services, im Petabyte-Bereich abzufragen.

Wenn Sie eine Richtlinie zur Aufbewahrung von Protokollen definieren, sollten Sie den Wert historischer Protokolle berücksichtigen. Historische Protokolle können dabei helfen, langfristige Nutzungs- und Verhaltensmuster, Regressionen und Leistungsverbesserungen Ihrer Workloads zu identifizieren. Dauerhaft gelöschte Protokolle können später nicht analysiert werden. Der Wert historischer Protokolle nimmt jedoch über längere Zeiträume hinweg tendenziell ab. Wählen Sie eine Richtlinie, die Ihren Anforderungen angemessen Rechnung trägt und alle gesetzlichen oder vertraglichen Anforderungen erfüllt, denen Sie möglicherweise unterliegen.

Implementierungsschritte

1. Wählen Sie die Erfassungs-, Speicher-, Analyse- und Anzeigemechanismen für Ihre Beobachtbarkeitsdaten.
2. Installieren und konfigurieren Sie Metrik- und Protokollkollektoren für die entsprechenden Komponenten Ihrer Workloads (z. B. für Amazon-EC2-Instances und in Sidecar-Containern). Konfigurieren Sie diese Kollektoren so, dass sie automatisch neu gestartet werden, wenn sie unerwartet beendet werden. Aktivieren Sie die Festplatten- oder Speicherpufferung für die Kollektoren, sodass temporäre Veröffentlichungsfehler Ihre Anwendungen nicht beeinträchtigen oder zu Datenverlusten führen.
3. Aktivieren Sie die Protokollierung für AWS-Services, die Sie im Rahmen Ihrer Workloads verwenden, und leiten Sie diese Protokolle an den ausgewählten Speicherservice weiter, wenn notwendig. Ausführliche Anweisungen finden Sie in den Benutzer- oder Entwicklerhandbüchern der jeweiligen Services.
4. Definieren Sie die für Ihre Workloads relevanten betrieblichen Metriken, die auf Ihren Telemetriedaten basieren. Diese können auf direkten Metriken basieren, die von Ihren Workload-Komponenten ausgegeben werden, zu denen auch Metriken im Zusammenhang mit Geschäftskennzahlen oder die Ergebnisse aggregierter Berechnungen wie Summen, Raten, Perzentile oder Histogramme gehören können. Berechnen Sie diese Metriken mit Ihrem Protokollanalysator und platzieren Sie sie gegebenenfalls in Dashboards.
5. Bereiten Sie geeignete Protokollabfragen vor, um Workload-Komponenten, Anfragen oder das Transaktionsverhalten nach Bedarf zu analysieren.
6. Definieren und aktivieren Sie eine Protokollaufbewahrungsrichtlinie für Ihre Komponentenprotokolle. Löschen Sie regelmäßig Protokolle, wenn sie älter sind, als es die Richtlinie zulässt.

Ressourcen

Zugehörige bewährte Methoden:

- REL06-BP01 Überwachen aller Komponenten des Workloads (Generierung)
- REL06-BP03 Senden von Benachrichtigungen (Verarbeitung und Benachrichtigung in Echtzeit)
- REL06-BP04 Automatisieren von Antworten (Verarbeitung und Benachrichtigung in Echtzeit)
- REL06-BP05 Analysieren von Protokollen
- REL06-BP06 Regelmäßiges Durchführen von Prüfungen von Umfang und Metriken
- REL06-BP07 Überwachen der gesamten Nachverfolgung von Anfragen im System

Zugehörige Dokumentation:

- Funktionsweise von Amazon CloudWatch
- Amazon Managed Prometheus
- Amazon Managed Grafana
- Analysieren von Protokolldaten mit CloudWatch Logs Insights
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Abfragen Ihrer Metriken mit CloudWatch Metrics Insights
- AWS Distro for OpenTelemetry
- Amazon CloudWatch Logs Insights Sample Queries
- Debuggen mit Amazon CloudWatch Synthetics und AWS X-Ray
- Suchen und Filtern von Protokolldaten
- Senden von Protokollen direkt an Amazon S3
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen

Zugehörige Workshops:

- Workshop zur Beobachtbarkeit

Zugehörige Tools:

- AWS Distro for OpenTelemetry (GitHub)

REL06-BP03 Senden von Benachrichtigungen (Verarbeitung und Benachrichtigung in Echtzeit)

Wenn Organisationen potenzielle Probleme erkennen, senden sie Benachrichtigungen und Warnungen in Echtzeit an das entsprechende Personal und die entsprechenden Systeme, um schnell und effektiv auf diese Probleme reagieren zu können.

Gewünschtes Ergebnis: Durch die Konfiguration relevanter Alarme auf der Grundlage von Service- und Anwendungsmetriken ist eine schnelle Reaktion auf operative Ereignisse möglich. Bei Überschreitung der Alarmschwellen werden das entsprechende Personal und die entsprechenden Systeme benachrichtigt, damit sie die zugrunde liegenden Probleme beseitigen können.

Typische Anti-Muster:

- Sie konfigurieren Alarme mit einem übermäßig hohen Schwellenwert, was dazu führt, dass wichtige Benachrichtigungen nicht gesendet werden können.
- Sie konfigurieren Alarme mit einem zu niedrigen Schwellenwert, was dazu führt, dass bei wichtigen Warnungen aufgrund des Lärms übermäßiger Benachrichtigungen keine Aktion erfolgt.
- Sie aktualisieren keine Alarme und ihre Schwellenwerte, wenn sich die Nutzung ändert.
- Bei Alarmen, die am besten durch automatische Aktionen behoben werden, führt das Senden der Benachrichtigung an das Personal, anstatt die automatische Aktion zu generieren, dazu, dass übermäßig viele Benachrichtigungen gesendet werden.

Vorteile der Nutzung dieser bewährten Methode: Das Senden von Benachrichtigungen und Warnungen in Echtzeit an das entsprechende Personal und die entsprechenden Systeme ermöglicht eine frühzeitige Erkennung von Problemen und eine schnelle Reaktion auf betriebliche Vorfälle.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Hoch

Implementierungsleitfaden

Workloads sollten mit der Verarbeitung und Benachrichtigung in Echtzeit ausgestattet sein, um die Erkennbarkeit von Problemen zu verbessern, die sich auf die Verfügbarkeit der Anwendung auswirken und als Auslöser für automatische Reaktionen dienen könnten. Organisationen können die Verarbeitung und Benachrichtigung in Echtzeit durchführen, indem sie Warnungen mit definierten

Metriken erstellen, um Benachrichtigungen zu erhalten, wenn wichtige Ereignisse eintreten oder eine Metrik einen Schwellenwert überschreitet.

Amazon CloudWatch ermöglicht es Ihnen, Metrik-Alarme und zusammengesetzte Alarme mithilfe von CloudWatch-Alarmen zu erstellen, die auf statischen Schwellenwerten, der Erkennung von Unregelmäßigkeiten und anderen Kriterien basieren. Weitere Informationen zu den Alarmtypen, die Sie mit CloudWatch konfigurieren können, finden Sie im Abschnitt über Alarme in der CloudWatch-Dokumentation.

Sie können benutzerdefinierte Ansichten von Metriken und Warnungen Ihrer AWS-Ressourcen für Ihre Teams erstellen, indem Sie CloudWatch-Dashboards nutzen. Die anpassbaren Startseiten in der CloudWatch-Konsole ermöglichen es Ihnen, Ihre Ressourcen in einer einzigen Ansicht über mehrere Regionen hinweg zu überwachen.

Alarme können mindestens eine Aktion ausführen, z. B. das Senden einer Benachrichtigung an ein Amazon SNS-Thema, das eine Amazon EC2-Aktion oder eine Amazon EC2 Auto Scaling-Aktion durchgeführt, oder das Erstellen eines OpsItem-Elements oder Vorfalls in AWS Systems Manager.

Amazon CloudWatch verwendet Amazon SNS zum Senden von Benachrichtigungen, wenn sich der Status des Alarms ändert, und ermöglicht so die Nachrichtenzustellung von den Publishern (Produzenten) an die Subscriber (Verbraucher). Weitere Informationen zum Einrichten von Amazon SNS-Benachrichtigungen finden Sie unter Configuring Amazon SNS.

CloudWatch sendet EventBridge-Ereignisse, wenn ein CloudWatch-Alarm erstellt, aktualisiert oder gelöscht wird oder sich sein Status ändert. Sie können EventBridge mit diesen Ereignissen verwenden, um Regeln zu erstellen, die Aktionen ausführen, z. B. Sie benachrichtigen, wenn sich der Status eines Alarms ändert, oder automatisch Ereignisse in Ihrem Konto mit Systems Manager-Automatisierung auslösen.

Bleiben Sie auf dem Laufenden mit AWS Health. AWS Health ist die maßgebliche Informationsquelle über den Zustand Ihrer AWS Cloud-Ressourcen. Verwenden Sie AWS Health, um über bestätigte Serviceereignisse benachrichtigt zu werden, so dass Sie schnell Maßnahmen ergreifen können, um etwaige Auswirkungen zu minimieren. Erstellen Sie maßgeschneiderte AWS Health-Ereignisbenachrichtigungen für E-Mail- und Chat-Kanäle über AWS-Benutzerbenachrichtigungen, und integrieren Sie sie programmgesteuert in Ihre Überwachungs- und Warnmeldungstools über Amazon EventBridge. Wenn Sie AWS Organizations verwenden, können Sie AWS Health-Ereignisse kontenübergreifend zusammenfassen.

Wann sollten Sie EventBridge im Vergleich zu Amazon SNS verwenden?

Sowohl EventBridge als auch Amazon SNS können zur Entwicklung ereignisgesteuerter Anwendungen verwendet werden. Ihre Wahl hängt von Ihren spezifischen Anforderungen ab.

Amazon EventBridge wird empfohlen, wenn Sie eine Anwendung erstellen möchten, die auf Ereignisse aus Ihren eigenen Anwendungen, SaaS-Anwendungen und AWS-Services reagiert. EventBridge ist der einzige ereignisbasierte Service, der direkt in SaaS-Partner von Drittanbietern integriert werden kann. EventBridge nimmt außerdem automatisch Ereignisse von über 200 AWS-Services auf, ohne dass Entwickler Ressourcen in ihrem Konto erstellen müssen.

EventBridge verwendet eine definierte JSON-basierte Struktur für Ereignisse und hilft Ihnen bei der Erstellung von Regeln, die auf den gesamten Ereignistext angewendet werden, um Ereignisse auszuwählen, die an ein Ziel weitergeleitet werden sollen. EventBridge unterstützt derzeit über 20 AWS-Services als Ziele, darunter AWS Lambda, Amazon SQS, Amazon SNS, Amazon Kinesis Data Streams und Amazon Data Firehose.

Amazon SNS wird für Anwendungen empfohlen, die eine hohe Verteilung benötigen (Tausende oder Millionen von Endpunkten). Ein gängiges Muster, das wir beobachten, ist, dass Kunden Amazon SNS als Ziel für ihre Regel verwenden, um die Ereignisse zu filtern, die sie benötigen, und dann an mehrere Endpunkte zu verteilen.

Nachrichten sind unstrukturiert und können in jedem Format vorliegen. Amazon SNS unterstützt die Weiterleitung von Nachrichten an sechs verschiedene Zieltypen, darunter Lambda, Amazon SQS, HTTP/S-Endpunkte, SMS, mobile Push-Benachrichtigungen und E-Mail. Die typische Latenz von Amazon SNS liegt unter 30 Millisekunden. Eine Vielzahl von AWS-Services sendet Amazon SNS-Nachrichten, indem sie den Service entsprechend konfigurieren (mehr als 30, einschließlich Amazon EC2, Amazon S3, and Amazon RDS).

Implementierungsschritte

1. Erstellen Sie einen Alarm mithilfe von Amazon CloudWatch-Alarmen.
    a. Ein metrischer Alarm überwacht eine einzelne CloudWatch-Metrik oder einen Ausdruck, der von CloudWatch-Metriken abhängig ist. Der Alarm initiiert eine oder mehrere Aktionen auf der Grundlage des Werts der Metrik oder des Ausdrucks im Vergleich zu einem Schwellenwert über eine Reihe von Zeitintervallen. Die Aktion kann darin bestehen, eine Benachrichtigung an ein Amazon SNS-Thema zu senden, das eine Amazon EC2-Aktion oder eine Amazon EC2 Auto Scaling-Aktion durchführt, oder ein OpsItem-Element oder einen Vorfall in AWS Systems Manager zu erstellen.
    b. Ein zusammengesetzter Alarm besteht aus einem Regelausdruck, der die Alarmbedingungen anderer von Ihnen erstellter Alarme berücksichtigt. Der zusammengesetzte Alarm wechselt nur

dann in den Alarmstatus, wenn alle Regelbedingungen erfüllt sind. Die im Regelausdruck eines zusammengesetzten Alarms angegebenen Alarme können metrische Alarme und zusätzliche zusammengesetzte Alarme enthalten. Zusammengesetzte Alarme können Amazon SNS-Benachrichtigungen senden, wenn sich ihr Status ändert, und sie können Systems Manager OpsItems-Elemente oder Vorfälle auslösen, wenn sie in den Alarmzustand wechseln, aber sie können weder Amazon EC2- noch Auto Scaling-Aktionen ausführen.

2. Richten Sie Amazon SNS-Benachrichtigungen ein. Wenn Sie einen CloudWatch-Alarm erstellen, können Sie ein Amazon SNS-Thema hinzufügen, um eine Benachrichtigung zu senden, wenn sich der Status des Alarms ändert.
3. Erstellen Sie Regeln in EventBridge, die bestimmten CloudWatch-Alarmen entsprechen. Jede Regel unterstützt mehrere Ziele, einschließlich Lambda-Funktionen. Jede Regel unterstützt mehrere Ziele, einschließlich Lambda-Funktionen. Sie können beispielsweise einen Alarm definieren, der initiiert wird, wenn der verfügbare Festplattenspeicher knapp wird, wodurch über eine EventBridge-Regel eine Lambda-Funktion ausgelöst wird, um den Speicherplatz zu bereinigen. Weitere Informationen zu EventBridge-Zielen finden Sie unter EventBridge-Ziele.

Ressourcen

Zugehörige bewährte Methoden für Well-Architected:

- REL06-BP01 Überwachen aller Komponenten der Workload (Generierung)
- REL06-BP02 Definieren und Berechnen von Metriken (Aggregierung)
- REL12-BP01 Untersuchen von Fehlern mit Playbooks

Zugehörige Dokumente:

- Amazon CloudWatch
- CloudWatch Logs Insights
- Verwenden von Amazon-CloudWatch-Alarmen
- Using Amazon CloudWatch dashboards
- Verwenden von Amazon-CloudWatch-Metriken
- Setting up Amazon SNS notifications
- CloudWatch anomaly detection
- CloudWatch Logs data protection

- Amazon EventBridge
- Amazon Simple Notification Service

Zugehörige Videos:

- reinvent 2022 observability videos
- AWS re:Invent 2022 - Observability best practices at Amazon

Zugehörige Beispiele:

- Workshop zur Beobachtbarkeit
- Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms

REL06-BP04 Automatisieren von Antworten (Verarbeitung und Benachrichtigung in Echtzeit)

Automatisieren Sie bei Erkennung von Ereignissen die erforderlichen Maßnahmen, wie etwa den Austausch fehlerhafter Komponenten.

Die automatische Echtzeitverarbeitung von Alarmen ist implementiert, sodass die Systeme bei Auslösung von Alarmen schnell korrigierend eingreifen und versuchen können, Ausfälle oder Beeinträchtigungen des Services zu verhindern. Zu den automatisierten Reaktionen auf Alarme könnten der Austausch ausgefallener Komponenten, die Anpassung der Rechenkapazität, die Umleitung des Datenverkehrs auf fehlerfreie Hosts, Availability Zones oder andere Regionen sowie die Benachrichtigung der Betreiber gehören.

Gewünschtes Ergebnis: Echtzeitalarme werden ermittelt und die automatische Verarbeitung von Alarmen wird eingerichtet, um die entsprechenden Maßnahmen zur Einhaltung von Service-Level-Zielen und Service Level Agreements (SLAs) einzuleiten. Die Automatisierung kann von der Selbstreparatur einzelner Komponenten bis hin zum Failover eines ganzen Standorts reichen. Die Automatisierung kann von der Selbstreparatur einzelner Komponenten bis hin zum Failover eines ganzen Standorts reichen.

Typische Anti-Muster:

- Fehlen einer genauen Bestandsaufnahme oder eines Katalogs der wichtigsten Echtzeitalarme

- Keine automatischen Reaktionen auf kritische Alarme (z. B. automatische Skalierung, wenn die Rechenkapazität fast erschöpft ist)
- Widersprüchliche Alarmreaktionen
- Fehlen von Standard-Betriebsabläufen (SOPs), an die sich die Bediener halten müssen, wenn sie Alarmmeldungen erhalten
- Keine Überwachung von Konfigurationsänderungen, da unentdeckte Konfigurationsänderungen zu Ausfallzeiten bei Workloads führen können
- Keine Strategie, um unbeabsichtigte Konfigurationsänderungen rückgängig zu machen

Vorteile der Nutzung dieser bewährten Methode: Die Automatisierung der Alarmverarbeitung kann die Ausfallsicherheit des Systems verbessern. Das System ergreift automatisch Korrekturmaßnahmen und reduziert so manuelle Tätigkeiten, bei denen es zu einem menschlichen, fehleranfälligen Eingreifen kommen kann. Der Workload-Betrieb erfüllt die Verfügbarkeitsziele und reduziert Serviceunterbrechungen.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

Zur wirksamen Verwaltung von Alarmen und zur Automatisierung ihrer Beantwortung kategorisieren Sie die Alarme nach ihrer Kritikalität und Auswirkung, dokumentieren die Reaktionsverfahren und planen die Reaktionen, bevor Sie die Aufgaben einordnen.

Ermitteln Sie Aufgaben, die bestimmte Aktionen erfordern (oft in Runbooks detailliert beschrieben), und untersuchen Sie alle Runbooks und Playbooks, um festzustellen, welche Aufgaben automatisiert werden können. Lassen sich Aktionen definieren, können sie oft auch automatisiert werden. Wenn Aktionen nicht automatisiert werden können, dokumentieren Sie die manuellen Schritte in einer SOP und schulen Sie die Mitarbeiter darin. Hinterfragen Sie kontinuierlich manuelle Prozesse und suchen Sie nach Möglichkeiten zur Automatisierung, um einen Plan für die Automatisierung von Alarmreaktionen zu erstellen und zu verwalten.

Implementierungsschritte

1. Erstellen eines Inventars von Alarmen: Um eine Liste aller Alarme zu erhalten, können Sie die AWS CLI mit dem Amazon CloudWatch-Befehl `describe-alarms` verwenden. Je nachdem, wie viele Alarme Sie eingerichtet haben, müssen Sie möglicherweise eine Paginierung verwenden, um eine Untergruppe von Alarmen für jeden Anruf aufzurufen. Alternativ können Sie das AWS-SDK verwenden, um die Alarme über einen API-Aufruf aufzurufen.

2. Dokumentieren aller Alarmaktionen: Aktualisieren Sie ein Runbook mit allen Alarmen und ihren Aktionen, unabhängig davon, ob sie manuell oder automatisiert sind. AWS Systems Manager bietet vordefinierte Runbooks. Ausführliche Informationen zum Anzeigen von Runbook-Inhalten finden Sie unter Working with runbooks. Ausführliche Informationen zum Anzeigen von Runbook-Inhalten finden Sie unter View runbook content.
3. Einrichten und Verwalten von Alarmaktionen: Für jeden der Alarme, die eine Aktion erfordern, geben Sie die automatische Aktion mithilfe des CloudWatch-SDK an. So können Sie beispielsweise den Zustand Ihrer Amazon EC2-Instances automatisch auf Grundlage eines CloudWatch-Alarms ändern, indem Sie Aktionen für einen Alarm erstellen und aktivieren oder Aktionen für einen Alarm deaktivieren.

   Sie können Amazon EventBridge auch verwenden, um automatisch auf Systemereignisse zu reagieren, z. B. auf Probleme mit der Anwendungsverfügbarkeit oder auf Ressourcenänderungen. Sie können Regeln erstellen, um anzugeben, an welchen Ereignissen Sie interessiert sind, und welche Aktionen auszuführen sind, wenn ein Ereignis mit einer Regel übereinstimmt. Zu den Aktionen, die automatisch ausgelöst werden können, gehören der Aufruf einer AWS Lambda-Funktion, der Aufruf von Amazon EC2 `Run Command`, die Weiterleitung des Ereignisses an Amazon Kinesis Data Streams und Automatisieren von Amazon EC2 mithilfe von EventBridge.
4. Standard-Betriebsabläufe (SOPs): Basierend auf den Komponenten Ihrer Anwendung empfiehlt AWS Resilience Hub mehrere SOP-Vorlagen. Sie können diese SOPs verwenden, um alle Prozesse zu dokumentieren, die ein Bediener im Falle eines Alarms befolgen sollte. Sie können auch eine SOP auf Grundlage von Resilience Hub-Empfehlungen erstellen, für die Sie eine Resilience Hub-Anwendung mit einer zugehörigen Resilienzrichtlinie sowie eine historische Resilienzbewertung für diese Anwendung benötigen. Die Empfehlungen für Ihre SOP ergeben sich aus der Resilienzbewertung.

   Resilience Hub arbeitet mit Systems Manager zusammen, um die einzelnen Schritte Ihrer SOPs zu automatisieren. Dazu erhalten Sie eine Reihe von SSM-Dokumenten, die Sie als Grundlage für diese SOPs verwenden können. So kann Resilience Hub zum Beispiel eine SOP für das Hinzufügen von Speicherplatz auf Grundlage eines bestehenden SSM-Automatisierungsdokuments empfehlen.
5. Durchführen automatisierter Aktionen mit Amazon DevOps Guru: Sie können Amazon DevOps Guru verwenden, um Anwendungsressourcen automatisch auf anomales Verhalten zu überwachen und gezielte Empfehlungen für eine schnellere Problemerkennung und -behebung zu geben. Mit DevOps Guru können Sie Ströme von Betriebsdaten aus verschiedenen Quellen wie Amazon CloudWatch-Metriken, AWS Config, AWS CloudFormation und AWS X-Ray nahezu in Echtzeit überwachen. Sie können DevOps Guru auch verwenden, um automatisch OpsItems

in OpsCenter zu erstellen und Ereignisse an EventBridge zu senden, um eine zusätzliche Automatisierung zu erreichen.

Ressourcen

Zugehörige bewährte Methoden:

- REL06-BP01 Überwachen aller Komponenten der Workload (Generierung)
- REL06-BP02 Definieren und Berechnen von Metriken (Aggregierung)
- REL06-BP03 Senden von Benachrichtigungen (Verarbeitung und Benachrichtigung in Echtzeit)
- REL08-BP01 Verwenden von Runbooks für Standardaktivitäten wie die Bereitstellung

Zugehörige Dokumente:

- AWS Systems Manager Automation
- Creating an EventBridge Rule That Triggers on an Event from an AWS Resource
- Workshop zur Beobachtbarkeit
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen
- What is Amazon DevOps Guru?
- Working with Automation Documents (Playbooks)

Zugehörige Videos:

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2020: Automate anything with AWS Systems Manager
- Introduction to AWS Resilience Hub
- Create Custom Ticket Systems for Amazon DevOps Guru Notifications
- Enable Multi-Account Insight Aggregation with Amazon DevOps Guru

Zugehörige Beispiele:

- Workshop zu Amazon CloudWatch und Systems Manager

REL06-BP05 Analysieren von Protokollen

Erfassen Sie Protokolldateien und Metrikverläufe und analysieren Sie diese, um allgemeine Trends zu erkennen und Workload-Einblicke zu erhalten.

Amazon CloudWatch Logs Insights unterstützt eine einfache, aber leistungsstarke Abfragesprache, mit der Sie Protokolldaten analysieren können. Amazon CloudWatch Logs unterstützt auch Abonnements, mit denen Daten nahtlos zu Amazon S3 fließen können, wo Sie sie nutzen oder Amazon Athena verwenden können, um die Daten abzufragen. Abfragen für eine große Auswahl von Formaten werden ebenfalls unterstützt. Unter Supported SerDes and Data Formats im Amazon Athena-Benutzerhandbuch finden Sie weitere Informationen dazu. Für die Analyse riesiger Protokolldateisätze können Sie einen Amazon EMR-Cluster ausführen, um Analysen im Petabyte-Bereich auszuführen.

Es gibt es eine Reihe von Werkzeugen von AWS-Partnern und externen Anbietern, die Aggregierung, Verarbeitung, Speicherung und Analyse ermöglichen. Dazu gehören u. a. die Tools New Relic, Splunk, Loggly, Logstash, CloudHealth und Nagios. Die Generierung außerhalb von System- und Anwendungsprotokollen weicht jedoch bei jedem Cloud-Anbieter und häufig sogar bei den einzelnen Services ab.

Ein häufig übersehener Teil des Überwachungsprozesses ist die Datenverwaltung. Sie müssen Aufbewahrungsanforderungen für die Überwachung von Daten definieren und anschließend entsprechende Lebenszyklusrichtlinien anwenden. Amazon S3 unterstützt die Lebenszyklusverwaltung auf der Ebene von S3-Buckets. Diese Lebenszyklusverwaltung kann auf unterschiedliche Weise auf verschiedene Pfade im Bucket angewendet werden. Gegen Ende des Lebenszyklus können Sie die Daten zur Langzeitspeicherung an Amazon Glacier übertragen und die Speicherung nach Ablauf des Aufbewahrungszeitraums beenden. Die S3 Intelligent-Tiering-Speicherklasse wurde entwickelt, um die Kosten zu optimieren. Daten werden automatisch in die kostengünstigste Zugriffsstufe verschoben, ohne Auswirkungen auf die Leistung oder höheren Betriebsaufwand.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

- Mit CloudWatch Logs Insights können Sie Protokolldaten in Amazon CloudWatch Logs interaktiv durchsuchen und analysieren.
  - Analyzing Log Data with CloudWatch Logs Insights
  - Amazon CloudWatch Logs Insights Sample Queries

- Senden Sie mit Amazon CloudWatch Logs Protokolle an Amazon S3. Dort können Sie die Daten mit Amazon Athena abfragen.
  - Wie verwende ich Amazon Athena, um meine Amazon-S3-Serverzugriffsprotokolle zu analysieren?
    - Erstellen Sie eine S3-Lebenszyklusrichtlinie für Ihren Bucket mit den Serverzugriffsprotokollen. Konfigurieren Sie die Richtlinie so, dass Protokolldateien regelmäßig entfernt werden. Dadurch wird die Menge der Daten reduziert, die Athena in einer Abfrage analysiert.
      - How Do I Create a Lifecycle Policy for an S3 Bucket?

Ressourcen

Zugehörige Dokumente:

- Amazon CloudWatch Logs Insights Sample Queries
- Analyzing Log Data with CloudWatch Logs Insights
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- How Do I Create a Lifecycle Policy for an S3 Bucket?
- Wie verwende ich Amazon Athena, um meine Amazon-S3-Serverzugriffsprotokolle zu analysieren?
- Workshop zur Beobachtbarkeit
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen

REL06-BP06 Regelmäßiges Durchführen von Prüfungen von Umfang und Metriken

Prüfen Sie häufig, wie die Workload-Überwachung implementiert ist, und aktualisieren Sie sie, wenn sich Ihre Workloads und ihre Architektur weiterentwickeln. Regelmäßige Prüfungen Ihrer Überwachung tragen dazu bei, das Risiko zu verringern, dass Fehlerindikatoren übersehen werden. Sie helfen außerdem Ihrem Workload, die Verfügbarkeitsziele zu erreichen.

Eine effektive Überwachung ist in wichtigen Geschäftsmetriken verankert, die entsprechend neuen Geschäftsprioritäten geändert werden. Ihr Monitoring-Überprüfungsprozess sollte den Schwerpunkt auf Service-Level-Indikatoren (SLIs) legen und Erkenntnisse aus Infrastruktur, Anwendungen, Clients und Benutzern einbeziehen.

Gewünschtes Ergebnis: Sie verfügen über eine effektive Überwachungsstrategie, die regelmäßig überprüft und in regelmäßigen Abständen sowie nach allen wichtigen Ereignissen oder Änderungen aktualisiert wird. Sie stellen sicher, dass die wichtigsten Indikatoren für den Zustand Ihrer Anwendungen relevant bleiben, wenn sich Ihre Workloads und Ihre Geschäftsanforderungen weiterentwickeln.

Typische Anti-Muster:

- Sie erfassen nur Standardmetriken.
- Sie richten eine Überwachungsstrategie ein, überprüfen sie aber nie.
- Bei der Bereitstellung größerer Änderungen wird die Überwachung nicht berücksichtigt.
- Sie vertrauen veralteten Metriken, um den Zustand eines Workloads zu bestimmen.
- Ihre operativen Teams werden aufgrund veralteter Metriken und Schwellenwerte mit Fehlalarmen überlastet.
- Ihnen fehlt die Beobachtbarkeit von Anwendungskomponenten, die nicht überwacht werden.
- Sie konzentrieren sich bei der Überwachung nur auf technische Metriken auf untergeordneter Ebene und schließen geschäftliche Metriken aus.

Vorteile der Nutzung dieser bewährten Methode: Wenn Sie die Überwachung regelmäßig überprüfen, können Sie potenzielle Probleme antizipieren und sicherstellen, dass Sie diese erkennen. Außerdem können Sie so blinde Flecken aufdecken, die Sie bei früheren Überprüfungen möglicherweise übersehen haben, was Ihre Fähigkeit, Probleme zu erkennen, weiter verbessert.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Überprüfen Sie Metriken und Umfang der Überwachung im Rahmen Ihrer Operational Readiness Review, ORR). Führen Sie regelmäßige Prüfungen der operativen Bereitschaft nach einem konsistenten Zeitplan durch, um festzustellen, ob zwischen Ihren aktuellen Workloads und der von Ihnen konfigurierten Überwachung Lücken bestehen. Der Aufbau einer Struktur mit regelmäßigen Überprüfungen der operativen Leistung und einem Wissensaustausch verbessert Ihre Fähigkeit, höhere Leistungen bei Ihren operativen Teams zu erzielen. Prüfen Sie, ob die vorhandenen Schwellenwerte für Warnmeldungen immer noch ausreichend sind, und prüfen Sie, ob operative Teams falsch-positive Warnmeldungen erhalten oder Aspekte der Anwendung, die überwacht werden sollten, nicht überwachen.

Das Framework für die Resilienzanalyse bietet nützliche Hinweise, die Ihnen bei der Steuerung des Prozesses helfen können. Der Schwerpunkt des Frameworks liegt auf der Identifizierung potenzieller Arten von Ausfällen und der präventiven und korrigierenden Maßnahmen, mit denen Sie ihre Auswirkungen abmildern können. Dieses Wissen kann Ihnen helfen, die richtigen Metriken und Ereignisse zu identifizieren, die Sie überwachen und bei denen Sie gewarnt werden sollten.

Implementierungsschritte

1. Planen und prüfen Sie die Workload-Dashboards regelmäßig. Was die Gründlichkeit der Untersuchungen angeht, sind unterschiedliche Intervalle denkbar.
2. Spüren Sie Trends in den Metriken auf. Vergleichen Sie die Metrikwerte mit Werten aus der Vergangenheit, um Trends zu erkennen, die darauf hinweisen könnten, dass etwas untersucht werden muss. Beispiele hierfür sind: zunehmende Latenz, Nachlassen der primären Geschäftsfunktion und zunehmende Anzahl von Reaktionen auf Fehler.
3. Suchen Sie in Ihren Metriken nach Ausreißern und Anomalien, die durch Durchschnitts- oder Medianwerte maskiert sein können. Sehen Sie sich die höchsten und niedrigsten Werte in einem bestimmten Zeitraum an und untersuchen Sie die Ursachen für Beobachtungen, die weit außerhalb der normalen Grenzen liegen. Beseitigen Sie nach und nach die Ursachen und legen Sie dabei einen immer engeren Maßstab für die erwarteten Metriken an, um auf die verbesserte Konsistenz der Workload-Leistung zu reagieren.
4. Spüren Sie plötzliche Änderungen im Verhalten auf. Eine plötzliche Veränderung in der Menge oder Richtung einer Metrik kann auf eine Änderung in der Anwendung hindeuten. Sie kann aber auch ein Hinweis auf externe Faktoren sein, für deren Verfolgung sie möglicherweise weitere Metriken hinzufügen müssen.
5. Prüfen Sie, ob die aktuelle Überwachungsstrategie für die Anwendung weiterhin relevant ist. Beurteilen Sie auf der Grundlage einer Analyse früherer Vorfälle (oder des Frameworks für die Resilienzanalyse), ob es weitere Aspekte der Anwendung gibt, die in den Überwachungsumfang aufgenommen werden sollten.
6. Überprüfen Sie Ihre RUM-Metriken (Real User Monitoring), um festzustellen, ob es Lücken bei der Abdeckung der Anwendungsfunktionen gibt.
7. Prüfen Sie Ihren Änderungsmanagementprozess. Aktualisieren Sie Ihre Verfahren bei Bedarf um einen Schritt zur Überwachung und Analyse, der durchgeführt werden sollte, bevor Sie eine Änderung genehmigen.
8. Implementieren Sie die Überprüfung der Überwachung als Teil Ihrer Prozesse zur Überprüfung der operativen Bereitschaft und zur Korrektur von Fehlern.

Ressourcen

Zugehörige bewährte Methoden:

- REL06-BP01 Überwachen aller Komponenten des Workloads (Generierung)
- REL06-BP02 Definieren und Berechnen von Metriken (Aggregierung)
- REL06-BP07 Überwachen der gesamten Nachverfolgung von Anfragen im System
- REL12-BP02 Durchführen von Analysen nach Vorfällen
- REL12-BP06 Regelmäßiges Durchführen von Gamedays

Zugehörige Dokumente:

- Warum sollten Sie eine Fehlerkorrektur (Correction of Error,CoE) entwickeln)
- Amazon CloudWatch Dashboards verwenden
- Erstellen von Dashboards für operative Sichtbarkeit
- Erweiterte Multi-AZ-Resilienzmuster – Graue Ausfälle
- Amazon CloudWatch Logs Insights Sample Queries
- Debuggen mit Amazon CloudWatch Synthetics und AWS X-Ray
- Workshop zur Beobachtbarkeit
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen
- Using Amazon CloudWatch Dashboards
- AWS Bewährte Methoden zur Beobachtbarkeit für
- Framework für Resilienzanalyse
- Framework für die Resilienzanalyse – Beobachtbarkeit
- Überprüfen der operativen Bereitschaft – ORR

REL06-BP07 Überwachen der gesamten Nachverfolgung von Anfragen im System

Verfolgen Sie Anfragen während der Bearbeitung durch die Servicekomponenten, damit Produktteams Probleme einfacher analysieren und beheben und die Leistung verbessern können.

Gewünschtes Ergebnis: Workloads mit umfassender Nachverfolgung über alle Komponenten hinweg lassen sich leicht debuggen und verbessern so die durchschnittliche Zeit für die Behebung (MTTR) von Fehlern und Latenz durch eine vereinfachte Ursachenerkennung. Die durchgängige Nachverfolgung reduziert die Zeit, die benötigt wird, um betroffene Komponenten zu erkennen und die Ursachen von Fehlern oder Latenzen genau zu ermitteln.

Typische Anti-Muster:

- Nachverfolgung wird für einige Komponenten verwendet, aber nicht für alle. Ohne Nachverfolgung in AWS Lambda können Teams beispielsweise die durch Kaltstarts bei hohen Workloads verursachte Latenz nicht genau nachvollziehen.
- Synthetische Canaries oder Real-User Monitoring (RUM) sind nicht für Nachverfolgung konfiguriert. Ohne Canaries oder RUM wird die Telemetrie der Client-Interaktion in der Spurenanalyse ausgelassen, was zu einem unvollständigen Leistungsprofil führt.
- Hybride Workloads umfassen sowohl cloudnative Nachverfolgungs-Tools als auch Tools von Drittanbietern, es wurden jedoch keine Schritte unternommen, um eine einzige Nachverfolgungs-Lösung auszuwählen und vollständig zu integrieren. Basierend auf der gewählten Nachverfolgungs-Lösung sollten cloudnative Nachverfolgungs-SDKs verwendet werden, um Komponenten zu instrumentieren, die nicht cloudnativ sind. Oder Tools von Drittanbietern sollten so konfiguriert werden, dass sie cloudnative Nachverfolgungstelemetrie aufnehmen.

Vorteile der Nutzung dieser bewährten Methode: Wenn Entwicklungsteams über Probleme informiert werden, können sie sich ein vollständiges Bild der Interaktionen zwischen den Systemkomponenten machen, einschließlich der Beziehung zwischen Komponenten, Protokollierung, Leistung und Ausfällen. Da die Nachverfolgung die visuelle Identifizierung der Ursachen erleichtert, können diese schneller untersucht werden. Teams, die die Interaktionen der Komponenten im Detail verstehen, treffen bessere und schnellere Entscheidungen bei der Lösung von Problemen. Entscheidungen, z. B. wann ein Notfallwiederherstellung (DR)-Failover eingeleitet werden sollte oder wo Strategien zur Selbstreparatur am besten implementiert werden sollten, können durch die Analyse von Systemprotokollen verbessert werden, was letztlich die Kundenzufriedenheit mit Ihren Services erhöht.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Teams, die verteilte Anwendungen betreiben, können mithilfe von Nachverfolgungs-Tools eine Korrelationskennung einrichten, Spuren von Anfragen erfassen und Service-Maps für verbundene

Komponenten erstellen. Alle Anwendungskomponenten sollten in den Anforderungsspuren enthalten sein, einschließlich Service-Clients, Middleware-Gateways und Event Busse, Rechenkomponenten und Speicher, einschließlich Schlüssel-Wert-Speicher und -Datenbanken. Integrieren Sie synthetische Canaries und Real-User Monitoring in Ihre Konfiguration für die gesamte Nachverfolgung, um die Interaktionen und Latenz von Remote-Clients zu messen, sodass Sie die Leistung Ihres Systems anhand Ihrer Service Level Agreements und Ziele genau bewerten können.

Nutzen Sie Instrumentierungsservices wie AWS X-Ray und Amazon CloudWatch-Anwendungsüberwachung, um einen vollständigen Überblick über die Anfragen zu erhalten, die in Ihrer Anwendung verarbeitet werden. X-Ray erfasst Anwendungstelemetrie und ermöglicht es Ihnen, diese nach Payloads, Funktionen, Spuren, Services und APIs zu visualisieren und zu filtern. Sie kann für Systemkomponenten aktiviert werden, bei denen kein Code oder Low-Code verwendet wird. Die CloudWatch-Anwendungsüberwachung umfasst ServiceLens, um Ihre Spuren in Metriken, Protokollen und Alarmen zu integrieren. Die CloudWatch-Anwendungsüberwachung umfasst auch synthetische Funktionen zur Überwachung Ihrer Endpunkte und APIs sowie Real-User Monitoring zur Instrumentierung Ihrer Webanwendungsclients.

Implementierungsschritte

- Verwenden Sie AWS X-Ray X-Ray auf allen unterstützten nativen Services wie Amazon S3, AWS Lambda und Amazon API Gateway. Diese AWS-Services ermöglichen X-Ray mit Konfigurationsschaltern unter Verwendung von Infrastruktur als Code, AWS SDKs oder der AWS-Managementkonsole.
- Instrumentenanwendungen AWS Distro for Open Telemetry und X-Ray oder Erfassungs-Agenten von Drittanbietern.
- Im AWS X-Ray-Entwicklerhandbuch finden Sie weitere Informationen für die programmiersprachenspezifische Implementierung. In diesen Dokumentationsabschnitten wird detailliert beschrieben, wie HTTP-Anfragen, SQL-Abfragen und andere Prozesse, die für Ihre Anwendungsprogrammiersprache spezifisch sind, instrumentiert werden.
- Verwenden Sie X-Ray-Nachverfolgung für Amazon CloudWatch synthetische Canaries und Amazon CloudWatch RUM, um den Anforderungspfad von Ihrem Endbenutzer-Client durch Ihre AWS-Infrastruktur zu analysieren.
- Konfigurieren Sie CloudWatch-Metriken und -Alarme auf der Grundlage des Ressourcenzustands und der Canary-Telemetrie, sodass Teams schnell über Probleme informiert werden und dann mit ServiceLens Spuren und Servicemaps eingehend untersuchen können.

- Aktivieren Sie die X-Ray-Integration für Nachverfolgungs-Tools von Drittanbietern wie Datadog, New Relic oder Dynatrace, wenn Sie Tools von Drittanbietern als primäre Nachverfolgungslösung verwenden.

Ressourcen

Zugehörige bewährte Methoden:

- REL06-BP01 Überwachen aller Komponenten der Workload (Generierung)
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

Zugehörige Dokumente:

- Was ist AWS X-Ray?
- Amazon CloudWatch-Anwendungsüberwachung
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen
- Integrating AWS X-Ray with other AWS services
- AWS Distro for OpenTelemetry und AWS X-Ray
- Amazon CloudWatch: Using synthetic monitoring
- Amazon CloudWatch: Use CloudWatch RUM
- Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm
- Verfügbarkeit und mehr: Verdeutlichung und Verbesserung der Ausfallsicherheit bei verteilten Systemen in AWS

Zugehörige Beispiele:

- Workshop zur Beobachtbarkeit

Zugehörige Videos:

- AWS re:Invent 2.022 - How to monitor applications across multiple accounts
- How to Monitor your AWS Applications

Zugehörige Tools:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

Entwerfen einer Workloads, die sich an Bedarfsänderungen anpasst

Eine skalierbare Workload bietet Elastizität, sodass Ressourcen automatisch hinzugefügt oder entfernt werden können, damit sie dem aktuellen Bedarf zu einem bestimmten Zeitpunkt genau entsprechen.

Bewährte Methoden

- REL07-BP01 Automatisiertes Abrufen oder Skalieren von Ressourcen
- REL07-BP02 Abrufen von Ressourcen bei Erkennen einer Beeinträchtigung einer Workload
- REL07-BP03 Abrufen von Ressourcen bei Feststellung, dass für eine Workload mehr Ressourcen benötigt werden
- REL07-BP04 Belastungstest Ihr Workload

REL07-BP01 Automatisiertes Abrufen oder Skalieren von Ressourcen

Ein Eckpfeiler der Zuverlässigkeit in der Cloud ist die programmatische Definition, Bereitstellung und Verwaltung Ihrer Infrastruktur und Ihrer Ressourcen. Die Automatisierung hilft Ihnen dabei, die Ressourcenbereitstellung zu optimieren, konsistente und sichere Bereitstellungen zu ermöglichen und Ressourcen in Ihrer gesamten Infrastruktur zu skalieren.

Gewünschtes Ergebnis: Sie verwalten Ihre Infrastructure as Code (IaC). Sie definieren und verwalten Ihren Infrastrukturcode in Versionsverwaltungssystemen (VCS, Version Control Systems). Sie delegieren die Bereitstellung von AWS-Ressourcen an automatisierte Mechanismen und nutzen verwaltete Services wie Application Load Balancer (ALB), Network Load Balancer (NLB) und Auto-Scaling-Gruppen. Sie stellen Ihre Ressourcen mithilfe von CI/CD-Pipelines (Continuous Integration/Continuous Delivery) bereit, sodass Codeänderungen automatisch Ressourcenaktualisierungen einleiten, einschließlich Aktualisierungen Ihrer Auto-Scaling-Konfigurationen.

Typische Anti-Muster:

- Sie stellen Ressourcen manuell über die Befehlszeile oder in der AWS-Managementkonsole bereit (auch als ClickOps bezeichnet).
- Sie koppeln Ihre Anwendungskomponenten oder Ressourcen eng miteinander und schaffen dadurch unflexible Architekturen.
- Sie implementieren starre Skalierungsrichtlinien, die sich nicht an geänderte Geschäftsanforderungen, Datenverkehrsmuster oder neue Ressourcentypen anpassen.
- Sie schätzen die Kapazität manuell ein, um den erwarteten Bedarf zu decken.

Vorteile der Einführung dieser bewährten Methode: Mit Infrastructure as Code (IaC) kann Infrastruktur programmgesteuert definiert werden. Auf diese Weise können Sie Infrastrukturänderungen über denselben Softwareentwicklungszyklus verwalten wie Anwendungsänderungen. Dies fördert die Konsistenz und Wiederholbarkeit und verringert das Risiko manueller, fehleranfälliger Aufgaben. Sie können den Prozess der Bereitstellung und Aktualisierung von Ressourcen weiter optimieren, indem Sie IaC mit automatisierten Bereitstellungspipelines implementieren. Sie können Infrastrukturupdates zuverlässig und effizient bereitstellen, ohne dass manuelle Eingriffe erforderlich sind. Diese Flexibilität ist besonders wichtig, wenn Ressourcen skaliert werden, um schwankenden Anforderungen gerecht zu werden.

Sie·erstellen·aus·Ihren·Protokollen·aggregierte·Metriken,·die·das·Verhalten·Ihrer·Workloads aus·einer·übergeordneten·Perspektive·darstellen. Durch die Überwachung wichtiger Metriken und die Anwendung vordefinierter Skalierungsrichtlinien kann Auto Scaling Ressourcen nach Bedarf automatisch bereitstellen oder die Bereitstellung aufheben, was die Leistung und Kosteneffizienz verbessert. Dadurch wird das Risiko manueller Fehler oder Verzögerungen bei der Reaktion auf Änderungen der Anwendungs- oder Workload-Anforderungen reduziert.

Die Kombination aus IaC, automatisierten Bereitstellungspipelines und Auto Scaling hilft Unternehmen dabei, ihre Umgebungen zuverlässig bereitzustellen, zu aktualisieren und zu skalieren. Diese Automatisierung ist für die Aufrechterhaltung einer reaktionsschnellen, belastbaren und effizient verwalteten Cloud-Infrastruktur unerlässlich.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Um die Automatisierung mit CI/CD-Pipelines und Infrastructure as Code (IaC) für Ihre AWS-Architektur einzurichten, wählen Sie ein Versionskontrollsystem wie Git, um Ihre IaC-Vorlagen und die Konfiguration zu speichern. Diese Vorlagen können mit Tools wie AWS CloudFormation

geschrieben werden. Definieren Sie zunächst Ihre Infrastrukturkomponenten (z. B. AWSVPCs, Auto-Scaling-Gruppen in Amazon EC2 und Amazon-RDS-Datenbanken) innerhalb dieser Vorlagen.

Integrieren Sie dann diese IaC-Vorlagen in eine CI/CD-Pipeline, um den Bereitstellungsprozess zu automatisieren. AWS CodePipelinebietet eine nahtlose, AWS-native Lösung, Sie können aber auch andere CI/CD-Lösungen von Drittanbietern verwenden. Erstellen Sie eine Pipeline, die bei Änderungen an Ihrem Versionsverwaltungs-Repository aktiviert wird. Konfigurieren Sie die Pipeline so, dass sie Stufen enthält, die Ihre IaC-Vorlagen erweitern und validieren, die Infrastruktur in einer Staging-Umgebung bereitstellen, automatisierte Tests ausführen und schließlich für die Produktion bereitstellen. Integrieren Sie bei Bedarf Genehmigungsschritte, um die Kontrolle über Änderungen zu behalten. Diese automatisierte Pipeline beschleunigt nicht nur die Bereitstellung, sondern sorgt auch für Konsistenz und Zuverlässigkeit in allen Umgebungen.

Konfigurieren Sie Auto Scaling für Ressourcen wie Amazon-EC2-Instances, Amazon-ECS-Aufgaben und Datenbankreplikate in Ihrer IaC, um bei Bedarf ein automatisches Aufskalieren und Abskalieren zu ermöglichen. Auf diese Weise verbessert sich die Verfügbarkeit und Leistung der Anwendungen und die Kosten werden optimiert, da die Ressourcen dynamisch an den Bedarf angepasst werden. Eine Liste der unterstützten Ressourcen finden Sie unter Amazon EC2 Auto Scaling und AWS Auto Scaling.

Implementierungsschritte

1. Erstellen und verwenden Sie ein Quellcode-Repository, um den Code zu speichern, der Ihre Infrastrukturkonfiguration steuert. Übernehmen Sie die Änderungen in dieses Repository, um alle laufenden Änderungen widerzuspiegeln, die Sie vornehmen möchten.
2. Wählen Sie eine Infrastructure-as-Code-Lösung wie AWS CloudFormation, um Ihre Infrastruktur auf dem neuesten Stand zu halten und Abweichungen (Drift) gegenüber dem geplanten Zustand zu erkennen.
3. Integrieren Sie Ihre IaC-Plattform mit Ihrer CI/CD-Pipeline, um Bereitstellungen zu automatisieren.
4. Ermitteln und erfassen Sie die geeigneten Metriken für eine automatische Skalierung von Ressourcen.
5. Konfigurieren Sie die automatische Skalierung von Ressourcen mithilfe von Richtlinien für das Auf- und Abskalieren, die für Ihre Workload-Komponenten geeignet sind. Erwägen Sie die Verwendung einer geplanten Skalierung bei vorhersehbaren Nutzungsmustern.
6. Überwachen Sie Bereitstellungen, um Fehler und Regressionen zu erkennen. Implementieren Sie Rollback-Mechanismen innerhalb Ihrer CI/CD-Plattform, um Änderungen bei Bedarf rückgängig zu machen.

Ressourcen

Zugehörige Dokumente:

- AWS Auto Scaling: Funktionsweise von Skalierungsplänen
- AWS Marketplace: Für Auto Scaling geeignete Produkte
- Automatische Verwaltung der Durchsatzkapazität mit DynamoDB-Auto-Scaling
- Nutzen eines Load Balancer mit einer Auto-Scaling-Gruppe
- Was ist AWS Global Accelerator?
- Was ist Amazon EC2 Auto Scaling?
- Was ist AWS Auto Scaling?
- Was ist Amazon CloudFront?
- Was ist Amazon Route 53?
- Was ist Elastic Load Balancing?
- Was ist ein Network Load Balancer?
- Was ist ein Application Load Balancer?
- Integrieren von Jenkins mit AWS CodeBuild und AWS CodeDeploy
- Erstellen einer vierstufigen Pipeline mit AWS CodePipeline

Zugehörige Videos:

- Zurück zu den Grundlagen: Stellen Sie Ihren Code in Amazon EC2 bereit
- AWS bietet Ihnen Unterstützung | Infrastructure-as-Code-Lösung mithilfe von AWS CloudFormation-Vorlagen starten
- Optimieren des Prozesses zur Software-Veröffentlichung mit AWS CodePipeline
- Überwachen von AWS-Ressourcen mithilfe von Amazon-CloudWatch-Dashboards
- Erstellen von konto- und regionsübergreifenden CloudWatch-Dashboards | Amazon Web Services

REL07-BP02 Abrufen von Ressourcen bei Erkennen einer Beeinträchtigung einer Workload

Skalieren Sie Ressourcen bei Bedarf reaktiv, wenn die Verfügbarkeit beeinträchtigt ist, um die Verfügbarkeit der Workload wiederherzustellen.

Sie müssen zunächst Zustandsprüfungen und die Kriterien für diese Prüfungen konfigurieren, um anzugeben, wann die Verfügbarkeit durch fehlende Ressourcen beeinträchtigt wird. Benachrichtigen Sie anschließend entweder die zuständigen Mitarbeiter, um die Ressource manuell zu skalieren, oder starten Sie die Automatisierung, um sie automatisch zu skalieren.

Die Skalierung kann manuell an Ihre Workload angepasst werden, z. B. indem Sie die Anzahl der EC2-Instances in einer Auto Scaling-Gruppe ändern oder den Durchsatz einer DynamoDB-Tabelle über die AWS-Managementkonsole oder AWS CLI. Wann immer es möglich ist, sollte jedoch Automatisierung eingesetzt werden (siehe Automatisiertes Abrufen oder Skalieren von Ressourcen).

Gewünschtes Ergebnis: Skalierungsaktivitäten (entweder automatisch oder manuell) werden eingeleitet, um die Verfügbarkeit wiederherzustellen, sobald ein Ausfall oder eine Verschlechterung der Kundenerfahrung festgestellt wird.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Implementieren Sie Beobachtbarkeit und Überwachung für alle Komponenten Ihres Workloads, um die Kundenerfahrung zu überwachen und Fehler zu erkennen. Definieren Sie die manuellen oder automatischen Verfahren zur Skalierung der erforderlichen Ressourcen. Weitere Informationen finden Sie unter REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler.

Implementierungsschritte

- Definieren Sie die manuellen oder automatisierten Verfahren, mit denen die erforderlichen Ressourcen skaliert werden.
  - Die Skalierungsverfahren hängen davon ab, wie die verschiedenen Komponenten innerhalb Ihres Workloads gestaltet sind.
  - Die Skalierungsverfahren variieren auch je nach der zugrunde liegenden Technologie, die verwendet wird.
    - Komponenten, die AWS Auto Scaling verwenden, können Skalierungspläne nutzen, um eine Reihe von Anweisungen für die Skalierung Ihrer Ressourcen zu konfigurieren. Wenn Sie mit AWS CloudFormation arbeiten oder AWS-Ressourcen Tags hinzufügen, können Sie pro Anwendung Skalierungspläne für verschiedene Ressourcengruppen einrichten. Auto Scaling bietet Empfehlungen für Skalierungsstrategien, die auf die einzelnen Ressourcen zugeschnitten sind. Nachdem Sie einen Skalierungsplan erstellt haben, kombiniert Auto Scaling zur Unterstützung Ihrer Skalierungsstrategie Methoden für die dynamische

und prädiktive Skalierung. Weitere Informationen finden Sie unter Funktionsweise von Skalierungsplänen.

- Mit Amazon EC2 Auto Scaling können Sie sicherstellen, dass Ihnen die richtige Anzahl von Amazon EC2-Instances zur Verfügung steht, um die Anwendungslast zu bewältigen. Sie erstellen Sammlungen von EC2 Instances, die als Auto Scaling-Gruppen bezeichnet werden. In jeder Auto Scaling-Gruppe können Sie die Mindestanzahl von Instances angeben. Amazon EC2 Auto Scaling stellt dann sicher, dass die Gruppe diese Größe nie unter- oder überschreitet. Weitere Informationen finden Sie unter Was ist Amazon EC2 Auto Scaling?
- Amazon DynamoDB-Auto-Scaling verwendet den -Application-Auto-Scaling-Service, um die bereitgestellte Durchsatzkapazität in Ihrem Namen als Reaktion auf tatsächliche Datenverkehrsmuster dynamisch anzupassen. Auf diese Weise kann eine Tabelle oder ein globaler sekundärer Index die bereitgestellte Lese- und Schreibkapazität zum Verarbeiten eines plötzlichen Datenverkehrsanstiegs ohne Drosselung erhöhen. Weitere Informationen finden Sie unter Automatische Verwaltung der Durchsatzkapazität mit DynamoDB-Auto-Scaling.

Ressourcen

Zugehörige bewährte Methoden:

- REL07-BP01 Automatisiertes Abrufen oder Skalieren von Ressourcen
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

Zugehörige Dokumente:

- AWS Auto Scaling: Funktionsweise von Skalierungsplänen
- Automatische Verwaltung der Durchsatzkapazität mit DynamoDB-Auto-Scaling
- Was ist Amazon EC2 Auto Scaling?

REL07-BP03 Abrufen von Ressourcen bei Feststellung, dass für eine Workload mehr Ressourcen benötigt werden

Eine der wertvollsten Funktionen von Cloud Computing ist die Fähigkeit, Ressourcen dynamisch bereitzustellen.

In herkömmlichen On-Premises-Computerumgebungen müssen Sie im Voraus genügend Kapazität identifizieren und bereitstellen, um Zeiten mit höchster Auslastung bewältigen zu können. Dies ist ein Problem, da es teuer ist und die Verfügbarkeit gefährdet sein kann, wenn Sie die Spitzenkapazitätsanforderungen des Workloads unterschätzen.

In der Cloud ist dies nicht notwendig. Hier können Sie die Rechen-, Datenbank- und sonstigen Ressourcenkapazitäten nach Bedarf bereitstellen, um den aktuellen und prognostizierten Bedarf zu decken. Automatisierte Lösungen wie Amazon EC2 Auto Scaling und Application Auto Scaling können Ressourcen basierend auf von Ihnen angegebenen Metriken online für Sie bereitstellen. Dies kann den Skalierungsprozess einfacher und vorhersehbarer machen. Ihre Workloads sind jetzt erheblich zuverlässiger, da sichergestellt wird, dass Ihnen jederzeit genügend Ressourcen zur Verfügung stehen.

Gewünschtes Ergebnis: Sie konfigurieren die automatische Skalierung von Rechenleistung und anderen Ressourcen, um den Bedarf zu decken. Sie sorgen in Ihren Skalierungsrichtlinien für ausreichend Spielraum, damit Datenverkehrsspitzen aufgefangen werden können, während zusätzliche Ressourcen online geschaltet werden.

Typische Anti-Muster:

- Sie stellen eine feste Anzahl von skalierbaren Ressourcen bereit.
- Sie wählen eine Skalierungsmetrik, die nicht dem tatsächlichen Bedarf entspricht.
- Sie bieten in Ihren Skalierungsplänen nicht genügend Spielraum, um Nachfragespitzen Rechnung zu tragen.
- Ihre Skalierungsrichtlinien erhöhen die Kapazität zu spät, dadurch kommt es zu einer Erschöpfung der Kapazitäten und zu einem schlechten Service, während zusätzliche Ressourcen online geschaltet werden.
- Sie konfigurieren die minimale und maximale Anzahl an Ressourcen nicht korrekt, was zu Skalierungsfehlern führt.

Vorteile der Einführung dieser bewährten Methode: Ausreichend Ressourcen zur Deckung des aktuellen Bedarfs sind entscheidend, um eine hohe Verfügbarkeit Ihres Workloads zu gewährleisten und Ihre definierten Servicelevel-Ziele (SLOs) einzuhalten. Durch die automatische Skalierung können Sie die richtige Menge an Rechen-, Datenbank- und sonstigen Ressourcen bereitstellen, die Ihr Workload benötigt, um den aktuellen und prognostizierten Bedarf zu decken. Sie müssen den Spitzenkapazitätsbedarf nicht ermitteln und Ressourcen nicht statisch zuweisen, um dem Bedarf gerecht zu werden. Sie haben vielmehr die Möglichkeit, bei steigendem Bedarf mehr Ressourcen

zuzuweisen und bei einem Rückgang des Bedarfs Ressourcen zu deaktivieren, um die Kosten zu senken.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Stellen Sie zunächst fest, ob die Workload-Komponente für die automatische Skalierung geeignet ist. Diese Komponenten werden als horizontal skalierbar bezeichnet, da sie dieselben Ressourcen bereitstellen und sich identisch verhalten. Beispiele für horizontal skalierbare Komponenten sind EC2-Instances, die gleich konfiguriert sind, Aufgaben von Amazon Elastic Container Service (ECS) und Pods, die auf Amazon Elastic Kubernetes Service (EKS) ausgeführt werden. Diese Rechenressourcen befinden sich in der Regel hinter einem Load Balancer und werden als Replikate bezeichnet.

Weitere replizierte Ressourcen können Datenbanklesereplikate, Amazon-DynamoDB-Tabellen und Amazon ElastiCache (Redis OSS)-Cluster sein. Eine vollständige Liste der unterstützten Ressourcen finden Sie unter AWS-Services, die Sie mit Application Auto Scaling verwenden können.

Bei containerbasierten Architekturen müssen Sie möglicherweise auf zwei verschiedene Arten skalieren. Zunächst müssen Sie möglicherweise die Container skalieren, die horizontal skalierbare Services bereitstellen. Zweitens müssen Sie möglicherweise die Rechenressourcen skalieren, um Platz für neue Container zu schaffen. Für jede Ebene gibt es unterschiedliche automatische Skalierungsmechanismen. Um ECS-Aufgaben zu skalieren, können Sie Application Auto Scaling verwenden. Um Kubernetes-Pods zu skalieren, können Sie Horizontal Pod Autoscaler (HPA) oder Kubernetes Event-Driven Autoscaling (KEDA) verwenden. Zum Skalieren der Rechenressourcen können Sie Kapazitätsanbieter für ECS oder, für Kubernetes, Karpenter oder Cluster Autoscaler verwenden.

Wählen Sie als Nächstes aus, wie Sie die automatische Skalierung durchführen möchten. Es gibt drei Hauptoptionen: metrikbasierte Skalierung, geplante Skalierung und prädiktive Skalierung.

Metrikbasierte Skalierung

Bei der metrikbasierten Skalierung werden Ressourcen auf der Grundlage des Werts einer oder mehrerer Skalierungsmetriken bereitgestellt. Eine Skalierungsmetrik entspricht dem Bedarf Ihres Workloads. Eine gute Methode, um geeignete Skalierungsmetriken zu ermitteln, besteht darin, Lasttests in einer Nicht-Produktionsumgebung durchzuführen. Halten Sie während der Lasttests die Anzahl skalierbarer Ressourcen konstant und erhöhen Sie die Nachfrage langsam (z. B. Durchsatz, Parallelität oder simulierte Benutzer). Suchen Sie dann nach Metriken, die sich bei steigender

Nachfrage erhöhen (oder verringern) und sich umgekehrt bei sinkender Nachfrage verringern (oder erhöhen). Zu den typischen Skalierungsmetriken gehören die CPU-Auslastung, die Tiefe der Arbeitswarteschlange (z. B. einer Amazon SQS-Warteschlange), die Anzahl der aktiven Benutzer und der Netzwerkdurchsatz.

> **Note**
>
> AWS hat beobachtet, dass bei den meisten Anwendungen die Speicherauslastung in der Aufwärmphase der Anwendung zunimmt und dann einen konstanten Wert erreicht. Wenn die Nachfrage sinkt, bleibt die Speicherauslastung in der Regel erhöht und geht nicht parallel zurück. Da die Speicherauslastung in beiden Richtungen nicht dem Bedarf entspricht, also nicht mit dem Bedarf steigt und fällt, sollten Sie es sich genau überlegen, bevor Sie diese Metrik für die automatische Skalierung auswählen.

Die metrikbasierte Skalierung ist eine latente Operation. Es kann mehrere Minuten dauern, bis die Nutzungsmetriken auf Auto-Scaling-Mechanismen übertragen werden. Diese Mechanismen warten in der Regel auf ein deutliches Signal für eine erhöhte Nachfrage, bevor sie reagieren. Wenn der Autoscaler dann neue Ressourcen erstellt, kann es zusätzliche Zeit dauern, bis diese voll funktionsfähig sind. Aus diesem Grund ist es wichtig, dass Sie Ihre Skalierungsmetrikziele nicht zu nahe an der Vollauslastung festlegen (z. B. 90 % CPU-Auslastung). Ansonsten besteht die Gefahr, dass die vorhandene Ressourcenkapazität ausgeschöpft wird, bevor zusätzliche Kapazität verfügbar ist. Um eine optimale Verfügbarkeit zu erreichen, können die Ziele für die Ressourcennutzung typischerweise zwischen 50 und 70 % liegen, je nach Bedarfsmuster und Zeitaufwand für die Bereitstellung zusätzlicher Ressourcen.

Geplante Skalierung

Bei der geplanten Skalierung werden Ressourcen basierend auf dem Kalender oder der Tageszeit bereitgestellt oder entfernt. Sie wird häufig für Workloads mit vorhersehbarem Bedarf verwendet, wenn es beispielsweise zu den Geschäftszeiten an Wochentagen oder bei Verkaufsveranstaltungen zu einer Spitzenauslastung kommt. Sowohl Amazon EC2 Auto Scaling als auch Application Auto Scaling unterstützen die geplante Skalierung. Der Cron Scaler von KEDA unterstützt die geplante Skalierung von Kubernetes-Pods.

Prädiktive Skalierung

Die prädiktive Skalierung verwendet Machine Learning, um Ressourcen automatisch basierend auf der erwarteten Nachfrage zu skalieren. Bei der prädiktiven Skalierung wird der historische Wert

einer von Ihnen angegebenen Nutzungsmetrik analysiert und ihr zukünftiger Wert kontinuierlich prognostiziert. Der prognostizierte Wert wird dann verwendet, um die Ressource nach oben oder unten zu skalieren. Amazon EC2 Auto Scaling kann eine prädiktive Skalierung durchführen.

Implementierungsschritte

1. Ermitteln Sie, ob die Workload-Komponente für die automatische Skalierung geeignet ist.
2. Finden Sie heraus, welche Art von Skalierungsmechanismus für den Workload am besten geeignet ist: metrikbasierte Skalierung, geplante Skalierung oder prädiktive Skalierung.
3. Wählen Sie den geeigneten automatischen Skalierungsmechanismus für die Komponente aus. Verwenden Sie Amazon EC2 Auto Scaling für Amazon-EC2-Instances. Verwenden Sie Application Auto Scaling für andere AWS-Services. Für Kubernetes-Pods (wie z. B. solche, die in einem Amazon-EKS-Cluster ausgeführt werden) können Sie Horizontal Pod Autoscaler (HPA) oder Kubernetes Event-Driven Autoscaling (KEDA) in Betracht ziehen. Für Kubernetes- oder EKS-Knoten sollten Sie Karpenter und Cluster Auto Scaler (CAS) in Betracht ziehen.
4. Führen Sie für die metrikbasierte oder geplante Skalierung Lasttests durch, um die geeigneten Skalierungsmetriken und Zielwerte für Ihren Workload zu ermitteln. Ermitteln Sie für die geplante Skalierung die benötigte Ressourcenanzahl zu den von Ihnen ausgewählten Daten und Uhrzeiten. Ermitteln Sie die maximale Anzahl an Ressourcen, die benötigt werden, um die erwartete Verkehrsspitze zu bewältigen.
5. Konfigurieren Sie den Autoscaler auf der Grundlage der oben gesammelten Informationen. Weitergehende Informationen finden Sie in der Dokumentation zum Auto-Scaling-Service. Stellen Sie sicher, dass die Ober- und Untergrenze für die Skalierung korrekt konfiguriert sind.
6. Vergewissern Sie sich, dass die Skalierungskonfiguration wie erwartet funktioniert. Führen Sie Lasttests in einer Nicht-Produktionsumgebung durch, beobachten Sie, wie das System reagiert und nehmen Sie gegebenenfalls Anpassungen vor. Wenn Sie Auto Scaling in der Produktion aktivieren, konfigurieren Sie entsprechende Alarme, damit Sie über unerwartetes Verhalten informiert werden.

Ressourcen

Zugehörige Dokumente:

- Was ist Amazon EC2 Auto Scaling?
- AWS – Präskriptive Leitlinien: Anwendungen für Lasttests
- AWS Marketplace: Für Auto Scaling geeignete Produkte

- Automatische Verwaltung der Durchsatzkapazität mit DynamoDB-Auto-Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Berichte über das Gesetz von Little

REL07-BP04 Belastungstest Ihr Workload

Messen Sie anhand von Lasttests, ob die Skalierung den Workload-Anforderungen gerecht wird.

Es ist wichtig, regelmäßige Lasttests durchzuführen. Auslastungstests sollten die Belastungsgrenze ermitteln und die Leistung Ihres Workloads testen. AWS macht es einfach, temporäre Testumgebungen einzurichten, die den Umfang Ihrer Produktionslast modellieren. Sie können in der Cloud bei Bedarf eine Testumgebung in Produktionsgröße einrichten, Ihre Tests abschließen und die Ressourcen dann wieder stilllegen. Weil Sie für die Testumgebung nur dann zahlen, wenn sie genutzt wird, können Sie Ihre Live-Umgebung zu einem Bruchteil der Kosten testen, die Sie an einem On-Premises-Standort hätten.

Lasttests in der Produktion sollten auch im Rahmen von Ernstfallübungen durchgeführt werden, bei denen das Produktionssystem in einem Zeitraum mit geringer Kundennutzung stark belastet wird. Alle Mitarbeiter sollten an dieser Übung beteiligt sein, die Ergebnisse gemeinsam interpretieren und auftretende Probleme beheben.

Typische Anti-Muster:

- Es werden Lasttests für Bereitstellungen durchgeführt, die nicht mit der Konfiguration der Produktionsumgebung übereinstimmen.
- Lasttests werden nur für einzelne Teile, nicht aber für den gesamten Workload durchgeführt.
- Es werden Lasttests mit einer Teilmenge von Anfragen durchgeführt, aber nicht mit einer repräsentativen Gruppe tatsächlicher Anfragen.
- Es werden Lasttests mit einem kleinen Sicherheitsfaktor durchgeführt, der über der erwarteten Last liegt.

Vorteile der Nutzung dieser bewährten Methode: Sie wissen, welche Komponenten in der Architektur unter Last ausfallen, und können die zu überwachenden Metriken festlegen, die rechtzeitig auf die Annäherung an die Belastungsgrenze hinweisen, damit Sie das Problem beheben und entsprechende Auswirkungen vermeiden können.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Bestimmen Sie anhand von Lasttests, welcher Aspekt der Workload angeben soll, dass Kapazität hinzugefügt oder entfernt werden muss. Bei Lasttests sollte ein repräsentativer Datenverkehr zum Einsatz kommen, der dem in der Produktion ähnelt. Erhöhen Sie unter Beobachtung der instrumentierten Metriken die Last, um zu bestimmen, welche Metrik angibt, wann Ressourcen hinzugefügt oder entfernt werden müssen.
  - Distributed Load Testing auf AWS: Simulieren Sie Tausende verbundener Benutzer
    - Ermitteln Sie die Zusammensetzung von Anfragen. Möglicherweise haben Sie unterschiedliche Zusammensetzungen von Anfragen. Daher sollten Sie sich bei der Ermittlung der Zusammensetzung des Datenverkehrs verschiedene Zeiträume ansehen.
    - Implementieren Sie einen Lasttreiber. Zum Implementieren eines Lasttreibers können Sie Software mit eigenem Code, Open-Source-Software oder kommerzielle Software verwenden.
    - Führen Sie Lasttests zunächst mit geringer Kapazität durch. Schon bei der Erhöhung der Last für eine Einheit mit geringerer Kapazität, etwa einer einzelnen Instance oder einem einzelnen Container, stellen Sie unmittelbare Auswirkungen fest.
    - Führen Sie Lasttests mit größerer Kapazität durch. Bei einer verteilten Last sehen die Auswirkungen anders aus. Daher müssen Sie bei Tests Bedingungen herstellen, die der Produktionsumgebung möglichst nahekommen.

Ressourcen

Zugehörige Dokumente:

- Distributed Load Testing aktiviert AWS: simulieren Sie Tausende verbundener Benutzer
- Anwendungen für Lasttests

Zugehörige Videos:

- AWS Summit ANZ 2023: Mit AWS Distributed Load Testing können Sie mit Zuversicht schneller vorankommen

Implementierung von Änderungen

Kontrollierte Änderungen sind erforderlich, um neue Funktionen bereitzustellen und sicherzustellen, dass für Workloads und Betriebsumgebung bekannte, ordnungsgemäß gepatchte Software ausgeführt wird. Wenn solche Änderungen nicht kontrolliert sind, ist es schwierig, die Auswirkungen der Änderungen vorherzusagen oder Probleme zu lösen, die sich aus ihnen ergeben.

Zusätzliche Bereitstellungsmuster zur Risikominderung

Feature-Flags (auch als Feature-Toggles bezeichnet) sind Konfigurationsoptionen für eine Anwendung. Sie können die Software mit einem ausgeschalteten Feature bereitstellen, sodass Kunden das Feature nicht sehen. Anschließend können Sie das Feature wie bei einer Canary-Bereitstellung aktivieren oder die Änderungsgeschwindigkeit auf 100 % setzen, um die Auswirkungen zu beobachten. Wenn die Bereitstellung Probleme bereitet, können Sie das Feature einfach wieder ausschalten, ohne ein Rollback durchführen zu müssen.

Zonale Bereitstellung mit Fehlerisolation: Eine der wichtigsten Regeln, die AWS für seine eigenen Bereitstellungen definiert hat, besteht darin zu vermeiden, gleichzeitig mehrere Availability Zones innerhalb einer Region zu berühren. Damit wird sichergestellt, dass Availability Zones zum Zwecke unserer Verfügbarkeitsberechnungen unabhängig sind. Wir empfehlen, bei Ihren Bereitstellungen ähnlichen Überlegungen zu folgen.

Überprüfungen der betrieblichen Bereitschaft (Operational Readiness Reviews, ORRs)

AWS betrachtet es als sinnvoll, die betriebliche Bereitschaft zu überprüfen, um die Vollständigkeit der Testverfahren, die Fähigkeit zur Überwachung und insbesondere die Fähigkeit zur Bewertung der Anwendungsleistung in Bezug auf die zugehörigen SLAs zu bewerten sowie Daten bereitzustellen, wenn es zu einer Unterbrechung oder sonstigen Anomalien kommt. Eine formale ORR erfolgt vor der ersten Produktionsbereitstellung. AWS wiederholt ORRs regelmäßig (einmal pro Jahr oder vor wichtigen Leistungszeiträumen), um damit sicherzustellen, dass keine Abweichungen von den betrieblichen Erwartungen vorliegen. Weitere Informationen zur betrieblichen Bereitschaft finden Sie in der Säule „Operative Exzellenz“ des AWS Well-Architected Framework.

Bewährte Methoden

- REL08-BP01 Verwenden von Runbooks für Standardaktivitäten wie die Bereitstellung
- REL08-BP02 Integrieren von Funktionstests in die Bereitstellung
- REL08-BP03 Integrieren von Ausfallsicherheitstests in die Bereitstellung
- REL08-BP04 Bereitstellung mit einer unveränderlichen Infrastruktur

- REL08-BP05 Automatisieren von Änderungen

REL08-BP01 Verwenden von Runbooks für Standardaktivitäten wie die Bereitstellung

Runbooks sind vordefinierte Verfahren, die ein bestimmtes Ergebnis verfolgen. Verwenden Sie Runbooks, um Standardaktivitäten manuell oder automatisch durchzuführen. Beispiele für solche Aktivitäten sind etwa die Bereitstellung und das Patchen einer Workload oder das Vornehmen von DNS-Änderungen.

Sie können z. B. Prozesse einrichten, um bei Bereitstellungen die Rollback-Sicherheit zu gewährleisten. Wenn Sie eine Bereitstellung ohne Unterbrechung für Ihre Kunden zurücksetzen können, steigert das die Zuverlässigkeit Ihres Service.

Für Runbook-Verfahren sollten Sie mit einem gültigen, effektiven manuellen Prozess beginnen, diesen in Code implementieren und ggf. die automatische Ausführung auslösen.

Selbst bei anspruchsvollen Workloads mit umfassender Automatisierung sind Runbooks nützlich, um Ernstfallübungen auszuführen oder strenge Berichterstellungs- und Auditing-Anforderungen zu erfüllen.

Playbooks werden als Reaktion auf bestimmte Vorfälle verwendet und mit Runbooks sollen bestimmte Ergebnisse erzielt werden. Häufig werden Runbooks für Routineaktivitäten genutzt, während Playbooks für die Reaktion auf außerplanmäßige Ereignisse verwendet werden.

Typische Anti-Muster:

- Durchführen ungeplanter Änderungen an der Konfiguration in der Produktion.
- Überspringen von Schritten in Ihrem Plan, um eine schnellere Bereitstellung durchzuführen, was zu einer fehlgeschlagenen Bereitstellung führt.
- Vornehmen von Änderungen, ohne die Umkehrung der Änderung zu testen.

Vorteile der Nutzung dieser bewährten Methode: Die effektive Änderungsplanung erhöht Ihre Fähigkeit, die Änderung erfolgreich auszuführen, da Sie sich aller betroffenen Systeme bewusst sind. Die Validierung Ihrer Änderungen in Testumgebungen erhöht Ihre Sicherheit.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Unterstützen Sie konsistente und schnelle Reaktionen auf gut bekannte Ereignisse, indem Sie Verfahren in Runbooks dokumentieren.
- Verwenden Sie zur Definition Ihrer Infrastruktur den Grundsatz „Infrastructure as Code“. Wenn Sie Ihre Infrastruktur mit AWS CloudFormation oder dem vertrauenswürdigen Tool eines Drittanbieters definieren, können Sie Änderungen mithilfe einer Versionskontrollsoftware versionieren und nachverfolgen.
  - Nutzen Sie zur Definition Ihrer Infrastruktur AWS CloudFormation (oder das vertrauenswürdige Tool eines Drittanbieters).
    - Was ist AWS CloudFormation?
  - Erstellen Sie unter Anwendung guter Grundsätze für das Softwaredesign Vorlagen, die getrennt und entkoppelt sind.
    - Ermitteln Sie die für die Implementierung erforderlichen Berechtigungen, Vorlagen und zuständigen Parteien.
      - Zugriffssteuerung mit AWS Identity and Access Management
    - Verwenden Sie ein gehostetes Quellcodeverwaltungssystem, das auf einer verbreiteten Technologie wie Git basiert, um Ihren Quellcode und Ihre Infrastruktur-as-Code (IaC)-Konfiguration zu speichern.

Ressourcen

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie beim Erstellen automatisierter Bereitstellungslösungen unterstützen können
- AWS Marketplace: Produkte zur Automatisierung Ihrer Bereitstellungen
- Was ist AWS CloudFormation?

Zugehörige Beispiele:

- Automating operations with Playbooks and Runbooks

REL08-BP02 Integrieren von Funktionstests in die Bereitstellung

Verwenden Sie Techniken wie Modultests und Integrationstests, die die erforderliche Funktionalität validieren.

Bei Modultests testen Sie die kleinste funktionale Codeeinheit, um ihr Verhalten zu überprüfen. Mit Integrationstests soll überprüft werden, ob jede Anwendungsfunktion den Softwareanforderungen entspricht. Während sich Modultests auf die isolierte Prüfung eines Teils einer Anwendung konzentrieren, berücksichtigen Integrationstests auch Nebenwirkungen (z. B. den Effekt, dass Daten durch eine Mutationsoperation verändert werden). In beiden Fällen sollten die Tests in eine Bereitstellungs-Pipeline integriert werden. Wenn die Erfolgskriterien nicht erfüllt sind, wird die Pipeline angehalten oder es wird ein Rollback durchgeführt. Diese Tests werden in einer Vorproduktionsumgebung ausgeführt, die vor der Produktion in der Pipeline bereitgestellt wird.

Im Idealfall sollten diese Tests automatisch als Teil von Build- und Bereitstellungsaktionen ausgeführt werden. Mit AWS CodePipeline übertragen Entwickler beispielsweise Änderungen an ein Quell-Repository, in dem CodePipeline die Änderungen automatisch erkennt. Die Anwendung wird erstellt und es werden Modultests ausgeführt. Nachdem die Modultests bestanden wurden, wird der erstellte Code für weitere Tests auf Staging-Servern bereitgestellt. Auf dem Staging-Server führt CodePipeline weitere Tests aus, z. B. Integrations- oder Belastungstests. Nach dem erfolgreichen Abschluss dieser Tests stellt CodePipeline den getesteten und genehmigten Code für Produktions-Instances bereit.

Gewünschtes Ergebnis: Sie nutzen die Automatisierung zur Durchführung von Modul- und Integrationstests, um zu überprüfen, ob sich Ihr Code erwartungsgemäß verhält. Diese Tests sind in den Bereitstellungsprozess integriert, bei einem fehlgeschlagenen Test wird die Bereitstellung abgebrochen.

Typische Anti-Muster:

- Sie ignorieren oder umgehen Testfehler und Testpläne während des Bereitstellungsprozesses, um die Bereitstellung zu beschleunigen.
- Sie führen Tests außerhalb der Bereitstellungspipeline manuell durch.
- Sie überspringen Testschritte in Ihrer Automatisierung durch manuelle Notfall-Workflows.
- Sie führen automatisierte Tests in einer Umgebung durch, die der Produktionsumgebung nicht sehr ähnlich ist.
- Sie erstellen eine Testsuite, die nicht flexibel genug ist und sich nur schwer warten, aktualisieren oder skalieren lässt, wenn sich die Anwendung weiterentwickelt.

Vorteile der Einführung dieser bewährten Methode: Durch automatisierte Tests während des Bereitstellungsprozesses werden Probleme frühzeitig erkannt. Dadurch verringert sich das Risiko einer Produktionsfreigabe mit Fehlern oder unerwartetem Verhalten. Modultests überprüfen, ob sich der Code wie gewünscht verhält und API-Verträge eingehalten werden. Integrationstests bestätigen, dass das System den angegebenen Anforderungen entsprechend funktioniert. Diese Arten von Tests werden verwendet, um die beabsichtigte Funktionsreihenfolge von Komponenten wie Benutzeroberflächen, APIs, Datenbanken und Quellcode zu überprüfen.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Hoch

Implementierungsleitfaden

Führen Sie einen Ansatz für eine testgesteuerte Entwicklung (TDD, Test-driven Development) für das Schreiben von Software ein, bei dem Sie Testfälle entwickeln, um Ihren Code zu spezifizieren und zu validieren. Erstellen Sie zunächst Testfälle für jede Funktion. Wenn der Test fehlschlägt, schreiben Sie neuen Code, um den Test zu bestehen. Dieser Ansatz hilft Ihnen, das erwartete Ergebnis jeder Funktion zu überprüfen. Führen Sie Modultests aus und überprüfen Sie, ob diese bestanden werden, bevor Sie Code in ein Quellcode-Repository übertragen.

Implementieren Sie sowohl Modul- als auch Integrationstests im Rahmen der Erstellungs-, Test- und Bereitstellungsphasen der CI/CD-Pipeline. Automatisieren Sie die Tests und leiten Sie automatisch Tests ein, wenn eine neue Version der Anwendung bereit für die Bereitstellung ist. Wenn die Erfolgskriterien nicht erfüllt sind, wird die Pipeline angehalten oder rückgängig gemacht.

Wenn es sich bei der Anwendung um eine Web-App oder eine mobile App handelt, führen Sie automatisierte Integrationstests auf mehreren Desktop-Browsern oder echten Geräten durch. Dieser Ansatz ist besonders nützlich, um die Kompatibilität und Funktionalität mobiler Apps auf einer Vielzahl von Geräten zu überprüfen.

Implementierungsschritte

1. Schreiben Sie Modultests, bevor Sie funktionalen Code schreiben (testgesteuerte Entwicklung oder TDD). Legen Sie Coderichtlinien fest, sodass das Schreiben und Ausführen von Modultests zu den nicht-funktionalen Codierungsanforderungen gehört.
2. Erstellen Sie eine Suite automatisierter Integrationstests, die die identifizierten testbaren Funktionen abdecken. Diese Tests sollten Benutzerinteraktionen simulieren und die erwarteten Ergebnisse validieren.

3. Erstellen Sie die erforderliche Testumgebung für die Ausführung der Integrationstests. Hierzu können Staging- oder Vorproduktionsumgebungen gehören, die der Produktionsumgebung sehr ähnlich sind.
4. Richten Sie Ihre Quell-, Erstellungs-, Test- und Bereitstellungsphasen über die AWS CodePipeline-Konsole oder die AWS Command Line Interface (CLI) ein.
5. Stellen Sie die Anwendung bereit, sobald der Code erstellt und getestet wurde. AWS CodeDeploy kann die Anwendung in Ihren Staging- (Test-) und Produktionsumgebungen bereitstellen. Zu diesen Umgebungen können Amazon-EC2-Instances, AWS Lambda-Funktionen oder On-Premises-Server gehören. Für die Bereitstellung der Anwendung in allen Umgebungen sollte derselbe Bereitstellungsmechanismus verwendet werden.
6. Überwachen Sie den Fortschritt Ihrer Pipeline und den Status jeder Phase. Verwenden Sie Qualitätsprüfungen, um die Pipeline auf der Grundlage des Teststatus zu blockieren. Außerdem können Sie Benachrichtigungen über jeden Ausfall einer Pipeline-Phase oder den Abschluss einer Pipeline erhalten.
7. Überwachen Sie kontinuierlich die Testergebnisse und suchen Sie nach Mustern, Regressionen oder Bereichen, die mehr Aufmerksamkeit erfordern. Verwenden Sie diese Informationen, um die Testsuite zu verbessern, Bereiche der Anwendung zu identifizieren, in denen robustere Tests erforderlich sind, und den Bereitstellungsprozess zu optimieren.

Ressourcen

Zugehörige bewährte Methoden:

- REL07-BP04 Durchführen von Lasttests für den Workload
- REL08-BP03 Integrieren von Ausfallsicherheitstests in die Bereitstellung
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- Präskriptive AWS-Leitlinien: Testautomatisierung
- Kontinuierliche Bereitstellung und kontinuierliche Integration
- Anzeigen für Funktionstests
- Überwachen von Pipelines
- Verwenden von AWS CodePipeline mit AWS CodeBuild zum Testen von Code und Ausführen von Builds

- AWS Device Farm

REL08-BP03 Integrieren von Ausfallsicherheitstests in die Bereitstellung

Integrieren Sie Resilienztests, indem Sie bewusst Fehler in Ihr System einleiten, um dessen Leistungsfähigkeit im Falle von Störszenarien zu messen. Resilienztests unterscheiden sich von Geräte- und Funktionstests, die normalerweise in Bereitstellungszyklen integriert werden, da sie sich auf die Identifizierung unerwarteter Ausfälle in Ihrem System konzentrieren. Es ist zwar sicher, in der Vorproduktion mit der Integration von Resilienztests zu beginnen, aber setzen Sie sich das Ziel, diese Tests im Rahmen Ihrer GameDays in der Produktion zu implementieren.

Gewünschtes Ergebnis: Resilienztests tragen dazu bei, Vertrauen in die Fähigkeit des Systems aufzubauen, Beeinträchtigungen in der Produktion standzuhalten. Experimente identifizieren Schwachstellen, die zu Ausfällen führen könnten. Auf diese Weise können Sie Ihr System verbessern, um Ausfälle und Beeinträchtigungen automatisch und effizient zu beheben.

Typische Anti-Muster:

- Mangelnde Beobachtbarkeit und Überwachung in Bereitstellungsprozessen
- Verlass auf Menschen, um Systemausfälle zu beheben
- Analysemechanismen von schlechter Qualität
- Fokus auf bekannte Probleme in einem System und das Fehlen von Experimenten, um unbekannte Probleme zu identifizieren
- Identifizieren von Fehlern, aber keine Lösung
- Keine Dokumentation der Erkenntnisse und keine Runbooks

Vorteile der Nutzung dieser bewährten Methode: Resilienztests, die in Ihre Bereitstellungen integriert sind, helfen dabei, unbekannte Probleme im System zu identifizieren, die andernfalls unbemerkt bleiben und zu Produktionsausfällen führen können. Die Identifizierung dieser unbekannten Probleme in einem System hilft Ihnen, Ergebnisse zu dokumentieren, Tests in Ihren CI/CD-Prozess zu integrieren und Runbooks zu erstellen, die die Schadensbegrenzung durch effiziente, wiederholbare Mechanismen vereinfachen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Die gängigsten Formen von Resilienztests, die in die Bereitstellungen Ihres Systems integriert werden können, sind die Notfallwiederherstellung und Chaos-Engineering.

- Fügen Sie Aktualisierungen Ihrer Notfallwiederherstellungspläne und Standardarbeitsanweisungen (SOPs) bei jeder wichtigen Bereitstellung hinzu.
- Integrieren Sie Zuverlässigkeitstests in Ihre automatisierten Bereitstellungspipelines. Services wie AWS Resilience Hub können in Ihre CI/CD-Pipeline integriert werden, um kontinuierliche Resilienzbewertungen zu erstellen, die im Rahmen jeder Bereitstellung automatisch bewertet werden.
- Definieren Sie Ihre Anwendungen in AWS Resilience Hub. Resilienzanalysen generieren Codefragmente, die Sie bei der Erstellung von Wiederherstellungsprozeduren als AWS Systems Manager-Dokumente für Ihre Anwendungen unterstützen und eine Liste mit empfohlenen Amazon CloudWatch-Monitoren und -Alarmen enthalten.
- Sobald Ihre DR-Pläne und SOPs aktualisiert sind, führen Sie Notfallwiederherstellungstests durch, um sicherzustellen, dass sie wirksam sind. Mithilfe von Notfallwiederherstellungstests können Sie feststellen, ob Sie Ihr System nach einem Ereignis wiederherstellen und zum normalen Betrieb zurückkehren können. Sie können verschiedene Notfallwiederherstellungsstrategien simulieren und feststellen, ob Ihre Planung ausreicht, um Ihre Verfügbarkeitsanforderungen zu erfüllen. Zu den gängigen Notfallwiederherstellungsstrategien gehören Backup und Wiederherstellung, Pilot Light, Cold Standby, Warm Standby, Hot Standby und Aktiv-Aktiv. Sie alle unterscheiden sich in Kosten und Komplexität. Vor dem Notfallwiederherstellungstest empfehlen wir, dass Sie Ihr Recovery Time Objective (RTO) und Ihr Recovery Point Objective (RPO) definieren, um die Wahl der zu simulierenden Strategie zu vereinfachen. AWS bietet Notfallwiederherstellungstools wie AWS Elastic Disaster Recovery, die Ihnen unter anderem den Einstieg in Ihre Planung und Tests erleichtern.
- Experimente im Bereich des Chaos-Engineering führen zu Störungen im System, wie z. B. Netzwerk- und Serviceausfällen. Durch die Simulation mit kontrollierten Ausfällen können Sie die Sicherheitsschwachstellen Ihres Systems erkennen und gleichzeitig die Auswirkungen der eingeführten Fehler eindämmen. Führen Sie wie bei den anderen Strategien kontrollierte Ausfallsimulationen in Umgebungen außerhalb der Produktion durch, indem Sie beispielsweise Services wie AWS Fault Injection Service nutzen, um vor dem Einsatz in der Produktion Vertrauen zu gewinnen.

Ressourcen

Zugehörige Dokumente:

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

Zugehörige Videos:

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

REL08-BP04 Bereitstellung mit einer unveränderlichen Infrastruktur

Eine unveränderliche Infrastruktur sieht vor, dass Updates, Sicherheits-Patches oder Konfigurationsänderungen nicht direkt in Produktions-Workloads durchgeführt werden. Wenn eine Änderung erforderlich ist, wird die Architektur auf einer neuen Infrastruktur eingerichtet und für die Produktion bereitgestellt.

Verfolgen Sie eine Strategie zur Bereitstellung einer unveränderlichen Infrastruktur, um die Zuverlässigkeit, Konsistenz und Reproduzierbarkeit Ihrer Workload-Bereitstellungen zu erhöhen.

Gewünschtes Ergebnis: Bei einer unveränderlichen Infrastruktur sind keine direkten Änderungen an den Infrastrukturressourcen innerhalb eines Workloads erlaubt. Wenn eine Änderung erforderlich ist, wird stattdessen ein neuer Satz aktualisierter Infrastrukturressourcen, der alle erforderlichen Änderungen enthält, parallel zu Ihren vorhandenen Ressourcen bereitgestellt. Diese Bereitstellung wird automatisch validiert und bei Erfolg wird der Datenverkehr schrittweise auf die neuen Ressourcen verlagert.

Diese Bereitstellungsstrategie gilt unter anderem für Softwareupdates, Sicherheits-Patches, Infrastrukturänderungen, Konfigurationsupdates und Anwendungsupdates.

Typische Anti-Muster:

- Implementieren von Änderungen an laufenden Infrastruktur-Ressourcen vor Ort.

Vorteile der Nutzung dieser bewährten Methode:

- Erhöhte Konsistenz zwischen verschiedenen Umgebungen: Da es keine Unterschiede bei den Infrastrukturressourcen zwischen den Umgebungen gibt, wird die Konsistenz erhöht und das Testen vereinfacht.
- Verringerung der Konfigurationsabweichungen: Durch das Ersetzen von Infrastrukturressourcen durch eine bekannte und versionskontrollierte Konfiguration wird die Infrastruktur in einen bekannten, getesteten und vertrauenswürdigen Zustand versetzt, wodurch Konfigurationsabweichungen vermieden werden.
- Zuverlässige atomare Bereitstellungen: Entweder werden die Verteilungen erfolgreich abgeschlossen oder es ändert sich nichts, was die Konsistenz und Zuverlässigkeit des Verteilungsprozesses erhöht.
- Vereinfachte Bereitstellungen: Bereitstellungen werden vereinfacht, da sie keine Upgrades unterstützen müssen. Upgrades sind einfach neue Bereitstellungen.
- Sicherere Bereitstellungen mit schnellen Rollback- und Wiederherstellungsprozessen: Bereitstellungen sind sicherer, da die vorherige funktionierende Version nicht geändert wird. Sie können einen Rollback zur vorherigen Version durchführen, wenn Fehler erkannt werden.
- Verbesserte Sicherheitslage: Indem Sie keine Änderungen an der Infrastruktur zulassen, können Fernzugriffsmechanismen (wie SSH) deaktiviert werden. Dadurch wird der Angriffsvektor reduziert und die Sicherheitslage Ihrer Organisation verbessert.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Automatisierung

Bei der Definition einer Strategie zur Bereitstellung einer unveränderlichen Infrastruktur empfiehlt es sich, die Automatisierung so weit wie möglich zu nutzen, um die Reproduzierbarkeit zu erhöhen und das Potenzial für menschliche Fehler zu minimieren. Weitere Informationen finden Sie unter REL08-BP05 Automatisierung von Änderungen und Automating safe, hands-off deployments.

Mit Infrastructure as code (IaC) werden Schritte zur Bereitstellung, Orchestrierung und Implementierung der Infrastruktur auf programmatische, beschreibende und deklarative Weise definiert und in einem Quellkontrollsystem gespeichert. Die Nutzung von Infrastructure as Code vereinfacht die Automatisierung der Infrastrukturbereitstellung und trägt zur Unveränderbarkeit der Infrastruktur bei.

Bereitstellungsmuster

Wenn eine Änderung des Workloads erforderlich ist, schreibt die Strategie der unveränderlichen Infrastrukturbereitstellung vor, dass ein neuer Satz von Infrastrukturressourcen bereitgestellt wird, einschließlich aller erforderlichen Änderungen. Es ist wichtig, dass diese neuen Ressourcen nach einem Muster eingeführt werden, das die Auswirkungen auf die Benutzer minimiert. Für diese Bereitstellung gibt es zwei Hauptstrategien:

Canary-Bereitstellung: Hierbei wird eine kleine Anzahl Ihrer Kunden auf die neue Version umgestellt, die in der Regel auf einer einzelnen Service-Instance (dem Canary) ausgeführt wird. Anschließend überprüfen Sie sämtliche Verhaltensänderungen oder Fehler, die generiert werden. Sie können Datenverkehr aus der Canary-Umgebung entfernen, wenn kritische Probleme auftreten, und die Benutzer auf die vorherige Version zurücksetzen. Wenn die Bereitstellung erfolgreich verläuft, können Sie das gewünschte Tempo beibehalten und die Änderungen auf Fehler überwachen, bis der Bereitstellungsvorgang vollständig abgeschlossen ist. Sie können AWS CodeDeploy mit einer Bereitstellungskonfiguration konfigurieren, die eine Canary-Bereitstellung ermöglicht.

Blue/Green-Bereitstellung: Verhält sich ähnlich wie die Canary-Bereitstellung, nur dass eine komplette Flotte der Anwendung parallel bereitgestellt wird. Sie können Ihre Bereitstellungen über die zwei Stacks (blau und grün) alternieren. Auch hier können Sie Datenverkehr an die neue Version senden und einen Failback auf die alte Version durchführen, wenn bei der Bereitstellung Probleme auftreten. Normalerweise wird der gesamte Datenverkehr auf einmal umgeschaltet. Sie können Ihren Datenverkehr aber auch auf die Versionen verteilen, um die Einführung der neuen Version mithilfe der gewichteten DNS-Routing-Funktionen von AmazonRoute 53 durchzuführen. Sie können AWS CodeDeploy und AWS Elastic Beanstalk mit einer Bereitstellungskonfiguration konfigurieren, die eine Blue/Green-Bereitstellung ermöglicht.

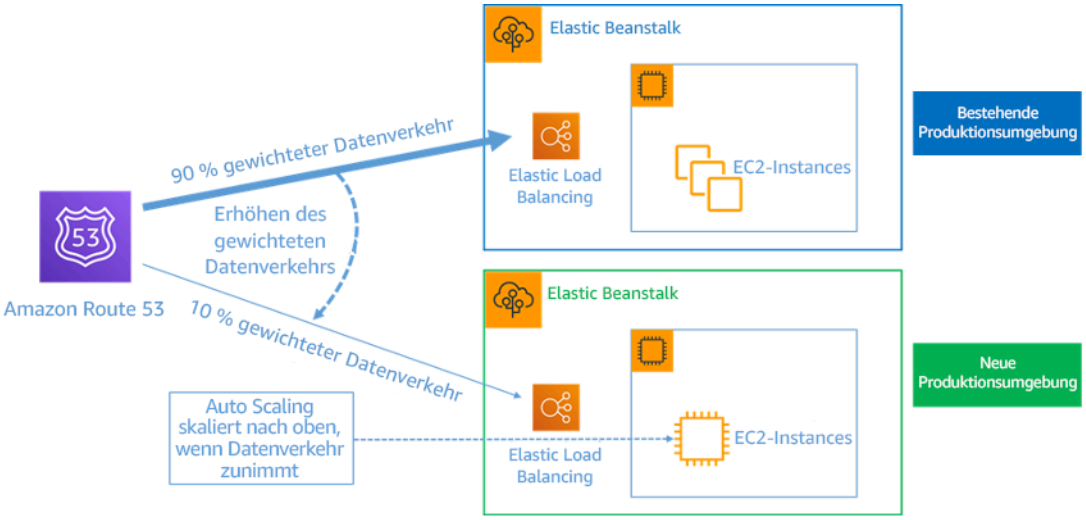


Abbildung 8: Blue/Green-Bereitstellung mit AWS Elastic Beanstalk und Amazon Route 53

Erkennung von Abweichungen

Als Abweichung wird jede Änderung bezeichnet, die dazu führt, dass eine Infrastrukturressource einen anderen Zustand oder eine andere Konfiguration aufweist als erwartet. Jede Art von nicht verwalteter Konfigurationsänderung widerspricht dem Konzept der unveränderlichen Infrastruktur und sollte erkannt und behoben werden, um eine erfolgreiche Implementierung der unveränderlichen Infrastruktur zu gewährleisten.

Implementierungsschritte

- Untersagen Sie die Änderung laufender Infrastruktur-Ressourcen an Ort und Stelle.
  - Sie können AWS Identity and Access Management (IAM) verwenden, um festzulegen, wer oder was auf Services und Ressourcen in AWS zugreifen darf, fein abgestufte Berechtigungen zentral verwalten und den Zugriff analysieren, um die Berechtigungen über AWS hinweg zu optimieren.
- Automatisieren Sie die Bereitstellung von Infrastrukturressourcen, um die Reproduzierbarkeit zu erhöhen und das Potenzial für menschliche Fehler zu minimieren.
  - Wie im Whitepaper Introduction to DevOps on AWS beschrieben, ist die Automatisierung ein Eckpfeiler der AWS-Services und wird intern in allen Services, Features und Angeboten unterstützt.
  - Durch die Vorberechnung Ihres Amazon Machine Image (AMI) können Sie die Zeit bis zum Start verkürzen. EC2 Image Builder ist ein vollständig verwalteter AWS-Service, der Ihnen hilft, die Erstellung, Wartung, Validierung, Freigabe und Bereitstellung von benutzerdefinierten, sicheren und aktuellen Linux- oder Windows-AMIs zu automatisieren.

- Zu den Services, die die Automatisierung unterstützen, gehören:
  - AWS Elastic Beanstalk ist ein Service zur schnellen Bereitstellung und Skalierung von Webanwendungen, die mit Java, .NET, PHP, Node.js, Python, Ruby, Go und Docker auf bekannten Servern wie Apache, NGINX, Passenger und IIS entwickelt wurden.
  - AWS Proton unterstützt Plattformteams dabei, all die verschiedenen Tools zu verbinden und zu koordinieren, die Ihre Entwicklungsteams für die Bereitstellung der Infrastruktur, die Bereitstellung von Code, die Überwachung und Updates benötigen. AWS Proton ermöglicht die automatisierte Bereitstellung von Infrastruktur als Code und die Bereitstellung von Serverless und containerbasierten Anwendungen.
- Die Nutzung von Infrastructure as Code erleichtert die Automatisierung der Infrastrukturbereitstellung und trägt zur Unveränderbarkeit der Infrastruktur bei. AWS bietet Services, die die Erstellung, Bereitstellung und Wartung der Infrastruktur auf programmatische, beschreibende und deklarative Weise ermöglichen.
  - AWS CloudFormation hilft Entwicklern dabei, AWS-Ressourcen in einer geordneten und vorhersehbaren Weise zu erstellen. Ressourcen werden in Textdateien im JSON- oder YAML-Format geschrieben. Die Vorlagen erfordern eine bestimmte Syntax und Struktur, die von den Arten der zu erstellenden und zu verwaltenden Ressourcen abhängt. Sie erstellen Ihre Ressourcen in JSON oder YAML mit einem beliebigen Code-Editor, checken sie in ein Versionskontrollsystem ein und CloudFormation baut dann die angegebenen Services auf sichere, wiederholbare Weise auf.
  - AWS Serverless Application Model (AWS SAM) ist ein Open-Source-Framework, mit dem Sie Serverless-Anwendungen in AWS erstellen können. AWS SAM lässt in anderen AWS-Services integrieren und ist eine Erweiterung von CloudFormation.
  - AWS Cloud Development Kit (AWS CDK) ist ein Open-Source-Framework für die Softwareentwicklung zur Modellierung und Bereitstellung Ihrer Cloud-Anwendungsressourcen mit Hilfe gängiger Programmiersprachen. Sie können AWS CDK verwenden, um die Anwendungsinfrastruktur mit TypeScript, Python, Java und .NET zu modellieren. AWS CDK verwendet CloudFormation im Hintergrund, um Ressourcen auf sichere, wiederholbare Weise bereitzustellen.
  - AWS -Cloud-Control- API führt einen gemeinsamen Satz von APIs zum Erstellen, Lesen, Aktualisieren, Löschen und Auflisten ein, mit denen Entwickler ihre Cloud-Infrastruktur auf einfache und konsistente Weise verwalten können. Die gemeinsamen Cloud Control API-APIs ermöglichen es Entwicklern, den Lebenszyklus von AWS- und Drittanbieter-Services einheitlich zu verwalten.
- Implementieren Sie Bereitstellungsmuster, die die Auswirkungen auf die Benutzer minimieren.

  - Canary-Bereitstellungen:
    - Einrichten einer API Gateway-Canary-Release-Bereitstellung
    - Erstellen einer Pipeline mit Canary-Bereitstellungen für Amazon ECS mit AWS App Mesh
  - Blau/Grün-Bereitstellungen: Das Whitepaper Blue/Green Deployments on AWS beschreibt Beispieltechniken zur Umsetzung von Blau/Grün-Bereitstellungsstrategien.
- Erkennen Sie Konfigurations- oder Zustandsabweichungen. Weitere Informationen finden Sie unter Detecting unmanaged configuration changes to stacks and resources.

Ressourcen

Zugehörige bewährte Methoden:

- REL08-BP05 Automatisieren von Änderungen

Zugehörige Dokumente:

- Automatisierung sicherer, vollautomatischer Bereitstellungen
- Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank
- Infrastructure as Code
- Implementing an alarm to automatically detect drift in AWS CloudFormation stacks

Zugehörige Videos:

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

REL08-BP05 Automatisieren von Änderungen

Bereitstellungen und Patches werden automatisiert, um negative Auswirkungen zu vermeiden.

Änderungen an Produktionssystemen gehören in vielen Organisationen zu den größten Risikofaktoren. Neben den geschäftlichen Problemen, die durch die Software behoben werden, betrachten wir Bereitstellungen als vorrangiges Problem, das es zu lösen gilt. Heutzutage bedeutet das, wenn immer möglich und sinnvoll, Vorgänge zu automatisieren. Dazu gehören Tests und die Bereitstellung von Änderungen, das Hinzufügen oder Entfernen von Kapazität und das Migrieren von Daten.

Gewünschtes Ergebnis: Sie integrieren automatische Bereitstellungssicherheit in den Veröffentlichungsprozess mit umfangreichen Tests vor der Produktion, automatischen Rollbacks und gestaffelten Produktionsbereitstellungen. Diese Automatisierung minimiert die potenziellen Auswirkungen auf die Produktion, die durch fehlgeschlagene Bereitstellungen verursacht werden, und Entwickler müssen die Bereitstellungen nicht mehr aktiv bis zur Produktion beobachten.

Typische Anti-Muster:

- Sie führen manuelle Änderungen durch.
- Sie überspringen Schritte in Ihrer Automatisierung durch manuelle Notfall-Workflows.
- Sie folgen nicht Ihren etablierten Plänen und Prozessen zugunsten beschleunigter Zeitpläne.
- Sie führen schnelle Folgebereitstellungen durch, ohne entsprechende Bake-Zeit einzuräumen.

Vorteile der Nutzung dieser bewährten Methode: Wenn Sie alle Änderungen mithilfe der Automatisierung implementieren, vermeiden Sie das Risiko menschlicher Fehler und bieten die Möglichkeit, Tests durchzuführen, bevor Sie die Produktion ändern. Wenn Sie diesen Vorgang vor dem Produktionsstart durchführen, wird sichergestellt, dass Ihre Pläne vollständig sind. Darüber hinaus kann ein automatisches Rollback in Ihren Veröffentlichungsprozess Produktionsprobleme identifizieren und Ihren Workload wieder in den ursprünglichen Betriebszustand versetzen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Automatisieren Sie Ihre Bereitstellungs-Pipeline. Mit Bereitstellungs-Pipelines können Sie Tests und die Entdeckung von Anomalien automatisieren und die Pipeline an einem bestimmten Schritt vor der Bereitstellung in der Produktion anhalten oder eine Änderung automatisch zurückführen. Ein integraler Bestandteil davon ist die Einführung einer Kultur der Continuous Integration und Continuous Delivery/Deployment (CI/CD), bei der ein Commit oder eine Codeänderung verschiedene automatische Stage-Gates von der Build- und Testphase bis zur Bereitstellung in Produktionsumgebungen durchläuft.

Obwohl es immer noch als sinnvoll erachtet wird, Personen bei den komplexesten betrieblichen Abläufen einzubinden, empfehlen wir, diese Abläufe wegen ihrer Komplexität zu automatisieren.

Implementierungsschritte

Sie können Bereitstellungen automatisieren, um manuelle Operationen zu vermeiden, indem Sie die folgenden Schritte ausführen:

- Richten Sie ein Code-Repository ein, um Ihren Code sicher zu speichern: Verwenden Sie ein gehostetes Quellcodeverwaltungssystem, das auf einer beliebten Technologie wie Git basiert, um Ihren Quellcode und Ihre Infrastructure-as-Code-Konfiguration (IaC) zu speichern.
- Konfigurieren eines Continuous-Integration-Services, um Ihren Quellcode zu kompilieren, Tests auszuführen und Bereitstellungsartefakte zu erstellen: Informationen zum Einrichten eines Build-Projekts für diesen Zweck finden Sie unter Getting started with AWS CodeBuild using the console.
- Einrichten eines Bereitstellungsservices, der Anwendungsbereitstellungen automatisiert und die Komplexität von Anwendungsupdates mindert, ohne auf fehleranfällige manuelle Bereitstellungen angewiesen zu sein: AWS CodeDeploy automatisiert Softwarebereitstellungen für eine Vielzahl von Computing-Services wie Amazon EC2, AWS Fargate, AWS Lambda, und auf Ihren On-Premises-Servern. Informationen zur Konfiguration dieser Schritte finden Sie unter Getting started with CodeDeploy.
- Konfigurieren eines Continuous-Integration-Services zur Automatisierung der Veröffentlichungspipelines für schnellere und zuverlässigere Anwendungs- und Infrastrukturupdates: Erwägen Sie die Verwendung von AWS CodePipeline, um die Automatisierung Ihrer Veröffentlichungspipelines zu unterstützen. Weitere Informationen finden Sie in den CodePipeline-Tutorials.

Ressourcen

Zugehörige bewährte Methoden:

- OPS05-BP04 Einsatz von Systemen zur Build- und Bereitstellungsverwaltung
- OPS05-BP10 Vollständige Automatisierung von Integration und Bereitstellung
- OPS06-BP02 Testbereitstellungen
- OPS06-BP04 Automatisieren von Tests und Rollback

Zugehörige Dokumente:

- Continuous Delivery of Nested AWS CloudFormation Stacks Using AWS CodePipeline
- APN-Partner: Partner, die Sie beim Erstellen automatisierter Bereitstellungslösungen unterstützen können
- AWS Marketplace: Produkte zur Automatisierung Ihrer Bereitstellungen
- Automate chat messages with webhooks.
- Die Amazon Builders' Library: Gewährleistung von Rollback-Sicherheit während der Bereitstellung

- Die Amazon Builders' Library: Schneller mit kontinuierlicher Lieferung
- Was ist AWS CodePipeline?
- Was ist CodeDeploy?
- AWS Systems Manager Patch Manager
- Was ist Amazon SES?
- Was ist Amazon Simple Notification Service?

Zugehörige Videos:

- AWS Summit 2019: CI/CD on AWS

Fehlerverwaltung

> Fehler treten immer wieder auf und Ausfälle sind von Zeit zu Zeit normal: ob bei Routern oder Festplatten, bei Betriebssystemen oder Speichereinheiten, die TCP-Pakete beschädigen. Außerdem können vorübergehende als auch permanente Fehler auftreten. Dies gilt unabhängig davon, ob Sie hochwertige Hardware oder kostengünstige Komponenten verwenden. – Werner Vogels, CTO – Amazon.com

Ausfälle von grundlegenden Hardwarekomponenten gehören im On-Premises-Rechenzentrum zum Alltag. In der Cloud sollten Sie jedoch vor den meisten dieser Ausfälle geschützt sein. Amazon-EBS-Volumes werden beispielsweise in einer bestimmten Availability Zone platziert, in der sie automatisch repliziert werden, um Sie vor dem Ausfall einer einzelnen Komponente zu schützen. Alle EBS-Volumes sind für eine Verfügbarkeit von 99,999 % ausgelegt. Amazon-S3-Objekte werden in mindestens drei Availability Zones gespeichert und bieten eine Dauerhaftigkeit von 99,999999999 % über den Zeitraum eines Jahres. Unabhängig von Ihrem Cloud-Anbieter können sich Ausfälle auf Ihre Workload auswirken. Daher müssen Sie Maßnahmen ergreifen, um für Ausfallsicherheit zu sorgen, wenn Sie auf eine zuverlässige Workload angewiesen sind.

Eine Voraussetzung für die Anwendung der hier beschriebenen bewährten Methoden besteht darin, zu gewährleisten, dass die Mitarbeiter, die Ihre Workloads entwerfen, implementieren und betreiben, mit den Geschäftszielen und der zum Erreichen dieser Ziele erforderlichen Zuverlässigkeit vertraut sind. Diese Mitarbeiter müssen die Zuverlässigkeitsanforderungen kennen und entsprechend geschult werden.

In den folgenden Abschnitten werden die bewährten Methoden für die Verwaltung von Fehlern erläutert, mit denen Sie Auswirkungen auf Ihre Workload verhindern können.

Themen

Sicherung von Daten

Sichern Sie Daten, Anwendungen und Konfigurationen, um die Anforderungen von Recovery Time Objective (RTO) und Recovery Point Objective (RPO) zu erfüllen.

Best Practices

- REL09-BP01 Ermitteln und Sichern aller zu sichernden Daten oder Reproduzieren der Daten aus Quellen
- REL09-BP02 Schützen und Verschlüsseln von Backups
- REL09-BP03 Automatische Daten-Backups
- REL09-BP04 Verifizieren der Sicherungsintegrität und -verfahren durch regelmäßiges Wiederherstellen der Daten

REL09-BP01 Ermitteln und Sichern aller zu sichernden Daten oder Reproduzieren der Daten aus Quellen

Sie sollten die Backup-Funktionen der von dem Workload genutzten Daten-Services und -Ressourcen verstehen und nutzen. Die meisten Services bieten Funktionen zur Sicherung von Workload-Daten.

Gewünschtes Ergebnis: Die Datenquellen wurden identifiziert und nach ihrer Bedeutung klassifiziert. Anschließend legen Sie eine auf dem RPO basierende Strategie für die Datenwiederherstellung fest. Diese Strategie involviert entweder die Sicherung dieser Datenquellen oder die Möglichkeit, Daten aus anderen Quellen zu reproduzieren. Im Falle eines Datenverlusts ermöglicht die implementierte Strategie die Wiederherstellung oder Reproduktion von Daten innerhalb der definierten RPO und RTO.

„Cloud-Reife“-Phase: Grundlegend

Typische Anti-Muster:

- Nicht alle Datenquellen für die Workload und deren Kritikalität sind bekannt.
- Es erfolgen keine Backups kritischer Datenquellen.
- Es erfolgen nur Backups von manchen Datenquellen ohne die Verwendung von Kritikalität als Kriterium.
- Es wurde kein RPO definiert oder die Backup-Häufigkeit kann den RPO nicht erfüllen.

- Es erfolgt keine Bewertung, ob ein Backup erforderlich ist oder ob Daten aus anderen Quellen reproduziert werden können.

Vorteile der Nutzung dieser bewährten Methode: Die Identifizierung der Stellen, an denen Backups erforderlich sind, und die Implementierung eines Mechanismus zur Erstellung von Backups oder die Möglichkeit, die Daten aus einer externen Quelle zu reproduzieren, verbessern die Fähigkeit zur Wiederherstellung und Wiederbeschaffung von Daten während eines Ausfalls.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Alle AWS-Datenspeicher bieten Backup-Möglichkeiten. Services wie Amazon RDS und Amazon DynamoDB unterstützen zusätzlich ein automatisiertes Backup, das eine zeitpunktbezogene Wiederherstellung (PITR) ermöglicht. So können Sie Backups zu einem beliebigen Zeitpunkt bis zu fünf Minuten oder weniger vor dem aktuellen Zeitpunkt wiederherstellen. Viele AWS-Services bieten die Möglichkeit, Backups in eine andere AWS-Region zu kopieren. AWS Backup ist ein Tool, das Ihnen die Möglichkeit gibt, den Schutz Ihrer Daten über AWS-Services hinweg zu zentralisieren und zu automatisieren. Mit AWS Elastic Disaster Recovery können Sie komplette Workloads von Servern kopieren und eine kontinuierliche Datensicherung von On-Premises-Ressourcen, AZ-übergreifenden Ressourcen oder Regionen hinweg aufrechterhalten. Das Recovery Point Objective (RPO) liegt dabei im Sekundenbereich.

Amazon S3 kann als Backup-Ziel für selbstverwaltete und AWS-verwaltete Datenquellen verwendet werden. AWS-Services wie Amazon EBS, Amazon RDS und Amazon DynamoDB bieten integrierte Möglichkeiten zur Backup-Erstellung. Sicherungssoftware von Drittanbietern kann ebenfalls eingesetzt werden.

On-Premises-Daten können in der AWS Cloud mit AWS Storage Gateway oder AWS DataSync gesichert werden. Mit Amazon S3-Buckets können Sie diese Daten auf AWS speichern. Amazon S3 bietet mehrere Speicherebenen wie Amazon Glacier oder Amazon Glacier Deep Archive, um die Kosten für den Datenspeicher zu senken.

Möglicherweise können Sie Ihre Datenwiederherstellungs-Anforderungen erfüllen, indem Sie Daten aus anderen Quellen reproduzieren. Zum Beispiel könnten Amazon ElastiCache-Replikat-Knoten oder Amazon RDS-Lesereplikate verwendet werden, um Daten zu reproduzieren, wenn der primäre Knoten verloren geht. In Fällen, in denen solche Quellen verwendet werden können, um Ihr Recovery Point Objective (RPO) und Recovery Time Objective (RTO) zu erfüllen, benötigen Sie möglicherweise

kein Backup. Ein weiteres Beispiel: Wenn Sie mit Amazon EMR arbeiten, ist es möglicherweise nicht notwendig, ein Backup Ihres HDFS-Datenspeichers zu erstellen, solange Sie die Daten aus Amazon S3 in Amazon EMR wiederherstellen können.

Bei der Auswahl einer Backup-Strategie sollten Sie die für die Datenwiederherstellung benötigte Zeit berücksichtigen. Diese hängt von der Art des Backups (im Falle einer Backup-Strategie) oder von der Komplexität des Datenreproduktions-Mechanismus ab. Die benötigte Zeit sollte im RTO für die Workload liegen.

Implementierungsschritte

1. Identifizieren Sie alle Datenquellen für die Workload. Daten können über verschiedene Ressourcen wie Datenbanken, Volumes, Dateisysteme, Protokollierungssysteme und Objektspeicher gespeichert werden. Im Abschnitt Ressourcen finden Sie verwandte Dokumente zu den verschiedenen AWS-Services, mit denen Daten gespeichert werden, und zu den Backup-Möglichkeiten, die diese Services bieten.
2. Klassifizieren Sie Datenquellen basierend auf Kritikalität. Unterschiedliche Datensätze haben unterschiedliche Kritikalitäts-Niveaus für eine Workload und damit auch verschiedene Anforderungen an die Ausfallsicherheit. So können beispielsweise bestimmte kritische Daten einen RPO erfordern, der gegen Null geht, während bei anderen, weniger kritischen Daten, ein höherer RPO und somit ein gewisser Datenverlust toleriert werden kann. Ebenso können unterschiedliche Datensätze auch unterschiedliche RTO-Anforderungen haben.
3. Nutzen Sie AWS- oder Drittanbieter-Services, um Backups der Daten zu erstellen. AWS Backup ist ein verwalteter Service, der die Erstellung von Backups von verschiedenen Datenquellen auf AWS ermöglicht. AWS Elastic Disaster Recovery übernimmt die automatisierte sekundengenaue Replikation von Daten in einer AWS-Region. Die meisten AWS-Services verfügen zusätzlich über native Funktionen zur Erstellung von Backups. Der AWS Marketplace umfasst zahlreiche Lösungen, die diese Funktionen ebenfalls bieten. In den unten aufgeführten Ressourcen finden Sie Informationen darüber, wie Sie Backups von Daten aus verschiedenen AWS-Services erstellen können.
4. Für Daten, die nicht gesichert werden, sollten Sie einen Datenreproduktions-Mechanismus festlegen. Es gibt verschiedene Gründe dafür, Daten, die aus anderen Quellen reproduziert werden können, nicht zu sichern. Möglicherweise ergibt sich die Situation, dass es günstiger ist, Daten bei Bedarf aus Quellen zu reproduzieren als ein Backup zu erstellen, da mit der Speicherung von Backups gewisse Kosten verbunden sind. Ein weiterer Grund wäre, wenn das Wiederherstellen aus einem Backup länger dauert als die Reproduktion der Daten aus anderen Quellen, was zu einer Nichteinhaltung des RTO führen würde. In solchen Situationen sollten

Sie sich einen Kompromiss überlegen und einen gut definierten Prozess festlegen, wie Daten aus diesen Quellen reproduziert werden können, wenn eine Datenwiederherstellung erforderlich ist. Wenn Sie beispielsweise Daten zur Analyse aus Amazon S3 in ein Data Warehouse (wie Amazon Redshift) oder einen MapReduce-Cluster (wie Amazon EMR) geladen haben, kann es sich dabei z. B. um Daten handeln, die aus anderen Quellen reproduziert werden können. Solange die Ergebnisse dieser Analysen gespeichert werden oder reproduzierbar sind, besteht kein Risiko eines Datenverlusts durch einen Ausfall im Data Warehouse oder MapReduce-Cluster. Andere Daten, die aus Quellen reproduziert werden können, sind Cache-Inhalte (z. B. Amazon ElastiCache) oder RDS Read Replicas.

5. Legen Sie eine Kadenz für die Sicherung von Daten fest. Das Erstellen von Datenquellen ist ein periodischer Prozess und die Häufigkeit sollte vom RPO abhängen.

Aufwand für den Implementierungsplan: Mittel.

Ressourcen

Zugehörige bewährte Methoden:

REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten

REL13-BP02 Verwenden von definierten Wiederherstellungsstrategien, um die Wiederherstellungsziele zu erreichen

Zugehörige Dokumente:

- Was ist AWS Backup?
- Was ist AWS DataSync?
- Was ist Volume Gateway?
- APN-Partner: Partner, die Sie bei der Sicherung unterstützen können
- AWS Marketplace: Für die Sicherung geeignete Produkte
- Amazon-EBS-Snapshots
- Backing Up Amazon EFS
- Backing up Amazon FSx for Windows File Server
- Backup and Restore for ElastiCache for Redis
- Creating a DB Cluster Snapshot in Neptune
- Creating a DB Snapshot

- Creating an EventBridge Rule That Triggers on a Schedule
- Cross-Region Replication with Amazon S3
- EFS-zu-EFS AWS Backup
- Exportieren von Protokolldaten nach Amazon S3
- Object lifecycle management
- On-Demand-Backup und Wiederherstellung für DynamoDB
- Zeitpunktbezogene Wiederherstellung für DynamoDB
- Working with Amazon OpenSearch Service Index Snapshots
- Was ist AWS Elastic Disaster Recovery?

Zugehörige Videos:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account & Cross-Region Backup
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

REL09-BP02 Schützen und Verschlüsseln von Backups

Kontrollieren und erkennen Sie den Zugriff auf Backups durch eine Authentifizierung und Autorisierung. Vermeiden und erkennen Sie mittels Verschlüsselung Beeinträchtigungen der Datenintegrität von Backups.

Implementieren Sie Sicherheitskontrollen, um einen unbefugten Zugriff auf Backup-Daten zu verhindern. Verschlüsseln Sie Backups, um die Vertraulichkeit und Integrität Ihrer Daten zu schützen.

Typische Anti-Muster:

- Derselbe Zugriff auf die Sicherungen und die automatisierte Wiederherstellung wie auf die Daten.
- Keine Verschlüsselung der Sicherungen.
- Keine Implementierung von Unveränderlichkeit zum Schutz vor Löschung oder Manipulation.
- Verwendung derselben Sicherheitsdomäne für Produktions- und Backup-Systeme.
- Keine Überprüfung der Integrität der Backups durch regelmäßige Tests.

Vorteile der Nutzung dieser bewährten Methode:

- Die Absicherung Ihrer Backups verhindert eine Manipulation der Daten und die Verschlüsselung der Daten sorgt dafür, dass kein Zugriff auf die Daten möglich ist, wenn sie versehentlich offengelegt werden.
- Verbesserter Schutz vor Ransomware und anderen Cyberbedrohungen, die auf die Backup-Infrastruktur abzielen.
- Kürzere Wiederherstellungszeit nach einem Cybervorfall dank geprüfter Wiederherstellungsprozesse.
- Verbesserte Fähigkeit zur Wahrung der Geschäftskontinuität bei Sicherheitsvorfällen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Steuern und erkennen Sie den Zugriff auf Backups durch Authentifizierung und Autorisierung wie z. B. mit AWS Identity and Access Management (IAM). Vermeiden und erkennen Sie mittels Verschlüsselung Beeinträchtigungen der Datenintegrität von Backups.

Amazon S3 unterstützt mehrere Verschlüsselungsmethoden für Daten im Ruhezustand. Mithilfe der serverseitigen Verschlüsselung akzeptiert Amazon S3 Ihre Objekte als unverschlüsselte Daten und sorgt für ihre Verschlüsselung bei der Speicherung. Bei der clientseitigen Verschlüsselung ist Ihre Workload-Anwendung für die Verschlüsselung der Daten verantwortlich, bevor sie an Amazon S3 gesendet werden. Beide Methoden ermöglichen Ihnen, zum Erstellen und Speichern des Datenschlüssels AWS Key Management Service (AWS KMS) zu verwenden oder einen eigenen Schlüssel bereitzustellen, für den Sie verantwortlich sind. Bei AWS KMS können Sie mithilfe von IAM festlegen, wer auf Ihre Datenschlüssel und entschlüsselten Daten zugreifen kann.

Wenn Sie bei Amazon RDS Ihre Datenbanken verschlüsseln, werden Ihre Sicherungsdaten ebenfalls verschlüsselt. DynamoDB-Sicherungen werden immer verschlüsselt. Bei Verwendung von AWS Elastic Disaster Recovery werden alle Daten während der Übertragung und im Ruhezustand verschlüsselt. Mit Elastic Disaster Recovery können Daten im Ruhezustand entweder mit dem standardmäßigen -Amazon EBS-Volume-Verschlüsselungsschlüssel oder einem vom Kunden verwalteten Schlüssel verschlüsselt werden.

Überlegungen zur Cyberresilienz

Um Ihre Backups besser vor Cyberbedrohungen zu schützen, sollten Sie neben der Verschlüsselung auch die Einführung dieser zusätzlichen Kontrollen in Betracht ziehen:

- Implementieren Sie Unveränderlichkeit mithilfe von AWS Backup Vault Lock oder Amazon S3 Object Lock, um zu verhindern, dass Backup-Daten während ihres Aufbewahrungszeitraums verändert oder gelöscht werden. Auf diese Weise schützen Sie sie vor Ransomware und böswilliger Löschung.
- Sorgen Sie für eine logische Trennung zwischen Produktions- und Backup-Umgebungen mit einem logischen Air-Gapped Vault von AWS Backup für kritische Systeme. Diese Trennung hilft, eine gleichzeitige Kompromittierung beider Umgebungen zu verhindern.
- Überprüfen Sie die Integrität Ihrer Backups regelmäßig. Stellen Sie dazu mithilfe von AWS Backup-Wiederherstellungstests sicher, dass die Backups nicht beschädigt sind und nach einem Cybervorfall erfolgreich wiederhergestellt werden können.
- Nutzen Sie die Mehrparteien-Genehmigung von AWS Backup für kritische Wiederherstellungsvorgänge, um sicherzustellen, dass keine unbefugten oder böswilligen Wiederherstellungsversuche erfolgen können, da eine Autorisierung durch mehrere benannte Genehmigungsberechtigte erforderlich ist.

Implementierungsschritte

1. Verwenden Sie eine Verschlüsselung für jeden Datenspeicher. Wenn Ihre Quelldaten verschlüsselt sind, wird die Sicherung ebenfalls verschlüsselt.
   - Nutzen Sie die Verschlüsselung in Amazon RDS. Beim Erstellen einer RDS-Instance können Sie die Verschlüsselung im Ruhezustand mit AWS Key Management Service konfigurieren.
   - Nutzen Sie die Verschlüsselung von Amazon EBS-Volumes. Während der Erstellung von Volumes können Sie eine Standardverschlüsselung konfigurieren oder einen eindeutigen Schlüssel angeben.
   - Verwenden Sie die erforderliche Amazon DynamoDB-Verschlüsselung. DynamoDB verschlüsselt alle Daten im Ruhezustand. Sie können entweder einen AWS-eigenen AWS KMS-Schlüssel oder einen AWS-verwalteten KMS-Schlüssel verwenden und dabei einen Schlüssel angeben, der in Ihrem Konto gespeichert wird.
   - Verschlüsseln Sie Ihre in Amazon EFS gespeicherten Daten Konfigurieren Sie die Verschlüsselung beim Erstellen des Dateisystems.
   - Konfigurieren Sie die Verschlüsselung in den Quell- und Zielregionen. Sie können die Verschlüsselung im Ruhezustand in Amazon S3 mit Schlüsseln konfigurieren, die in KMS gespeichert sind. Die Schlüssel sind jedoch regionsspezifisch. Sie können die Zielschlüssel angeben, während Sie die Replikation konfigurieren.

- Entscheiden Sie sich für die Standardverschlüsselung oder die angepasste Amazon EBS-Verschlüsselung für Elastic Disaster Recovery. Mit dieser Option werden Ihre replizierten Daten im Ruhezustand auf den Staging-Area Subnetz-Datenträgern und den replizierten Datenträgern verschlüsselt.

2. Implementieren Sie Rechte mit geringsten Berechtigungen für den Zugriff auf Ihre Backups. Begrenzen Sie den Zugriff auf die Backups, Snapshots und Replikate anhand bewährter Methoden im Bereich Sicherheit.
3. Konfigurieren Sie Unveränderlichkeit für kritische Backups. Implementieren Sie für kritische Daten AWS Backup Vault Lock oder S3 Object Lock, um zu verhindern, dass diese Daten während des angegebenen Aufbewahrungszeitraums gelöscht oder verändert werden. Weitere Informationen zur Implementierung finden Sie unter AWS Backup Vault Lock.
4. Schaffen Sie eine logische Trennung für Backup-Umgebungen. Implementieren Sie einen logischen Air-Gapped Vault von AWS Backup für kritische Systeme, die einen verbesserten Schutz vor Cyberbedrohungen benötigen. Anleitungen zur Implementierung finden Sie unter Building cyber resiliency with AWS Backup logically air-gapped vault.
5. Implementieren Sie Prozesse zur Backup-Validierung. Konfigurieren Sie AWS Backup-Wiederherstellungstests, um regelmäßig sicherzustellen, dass die Backups nicht beschädigt sind und nach einem Cybervorfall erfolgreich wiederhergestellt werden können. Weitere Informationen finden Sie unter Validate recovery readiness with AWS Backup restore testing.
6. Konfigurieren Sie eine Mehrparteien-Genehmigung für Wiederherstellungsvorgänge vertraulicher Daten. Implementieren Sie für kritische Systeme die Mehrparteien-Genehmigung von AWS Backup, sodass die Wiederherstellung erst fortgesetzt werden kann, nachdem eine Genehmigung von mehreren benannten Genehmigungsberechtigten eingeholt wurde. Einzelheiten zur Implementierung finden Sie unter Improve recovery resilience with AWS Backup support for Multi-party approval.

Ressourcen

Zugehörige Dokumente:

- AWS Marketplace: Für die Sicherung geeignete Produkte
- Amazon-EBS-Verschlüsselung
- Amazon S3: Daten durch Verschlüsselung schützen

- Zusätzliche CRR-Konfiguration: Replizieren von Objekten, die mit serverseitiger Verschlüsselung (SSE) unter Verwendung von Verschlüsselungsschlüsseln erstellt wurden, die in gespeichert wurden AWS KMS
- DynamoDB-Verschlüsselung in Ruhezustand
- Verschlüsseln von Amazon RDS-Ressourcen
- Encrypting Data and Metadata in Amazon EFS
- Encryption for Backups in AWS
- Managing Encrypted Tables
- Säule „Sicherheit“ – AWS Well-Architected Framework
- Was ist AWS Elastic Disaster Recovery?
- FSISEC11: Wie schützen Sie sich vor Ransomware?
- Ransomware Risk Management on AWS Using the NIST Cyber Security Framework
- Building cyber resiliency with AWS Backup logically air-gapped vault
- Validate recovery readiness with AWS Backup restore testing
- Improve recovery resilience with AWS Backup support for Multi-party approval

Zugehörige Beispiele:

- Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3

REL09-BP03 Automatische Daten-Backups

Sie können die Backups so konfigurieren, dass sie automatisch nach einem Zeitplan, der auf dem Recovery Point Objective (RPO) basiert, oder bei Änderungen am Datensatz durchgeführt werden. Kritische Datensätze, bei denen Datenverlust vermieden werden sollte, müssen regelmäßig automatisch gesichert werden, wohingegen weniger kritische Daten, bei denen ein gewisser Verlust akzeptabel ist, weniger häufig gesichert werden können.

Gewünschtes Ergebnis: Ein automatisierter Prozess, der Backups von Datenquellen in einem festgelegten Rhythmus erstellt.

Typische Anti-Muster:

- Sicherungen werden manuell durchgeführt.

- Es werden Ressourcen mit Sicherungsfunktionen verwendet, die Sicherung wird aber nicht in die Automatisierung einbezogen.

Vorteile der Nutzung dieser bewährten Methode: Durch die Automatisierung von Backups wird sichergestellt, dass diese regelmäßig gemäß Ihrem RPO durchgeführt werden. Sie werden gewarnt, wenn sie nicht durchgeführt werden.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

AWS Backup kann zum Erstellen von automatisierten Daten-Backups verschiedener AWS-Datenquellen genutzt werden. Amazon RDS-Instances können fast kontinuierlich alle fünf Minuten gesichert werden und Amazon S3-Objekte können praktisch durchgehend alle 15 Minuten gesichert werden, was eine zeitpunktbezogene Wiederherstellung (PITR) an einem bestimmten Zeitpunkt im Backup-Verlauf ermöglicht. Andere AWS-Datenquellen wie Amazon EBS-Volumes, Amazon DynamoDB-Tabellen oder Amazon FSx-Dateisysteme kann AWS Backup stündlich ein automatisiertes Backup ausführen. Diese Services bieten außerdem native Backup-Funktionen. Zu den AWS-Services, die ein automatisiertes Backup mit zeitpunktbezogener Wiederherstellung anbieten, gehören Amazon DynamoDB, Amazon RDS und Amazon Keyspaces (für Apache Cassandra). Diese können bis zu einem bestimmten Zeitpunkt innerhalb der Backup-Historie wiederhergestellt werden. Die meisten anderen AWS-Datenspeicher-Services bieten die Möglichkeit, stündliche periodische Backups einzuplanen.

Amazon RDS und Amazon DynamoDB bieten ein kontinuierliches Backup mit zeitpunktbezogener Wiederherstellung. Die Amazon S3-Versionsverwaltung erfolgt nach der Aktivierung automatisch. Mit Amazon Data Lifecycle Manager können Sie das Erstellen, Kopieren und Löschen von Amazon EBS-Snapshots automatisieren. Außerdem können damit das Erstellen, das Kopieren, die Außerbetriebnehmen und die Abmeldung von Amazon EBS-gestützten Amazon Machine Images (AMIs) und den zugrunde liegenden Amazon EBS-Snapshots automatisiert werden.

AWS Elastic Disaster Recovery bietet eine kontinuierliche Replikation auf Blockebene von der Quellumgebung (On-Premises oder AWS) zur Ziel-Wiederherstellungsregion. Point-in-Time-Amazon EBS-Snapshots werden automatisch vom Service erstellt und verwaltet.

Für eine zentrale Ansicht Ihrer Sicherungsautomatisierung und des Verlaufs bietet AWS Backup eine vollständig verwaltete, richtlinienbasierte Sicherungslösung. Diese zentralisiert und automatisiert die Sicherung von Daten in mehreren AWS-Services in der Cloud sowie On-Premises mithilfe des AWS Storage Gateway.

Zusätzlich zur Versionsverwaltung bietet Amazon S3 eine Replikationsfunktion. Der gesamte S3-Bucket kann automatisch in einen anderen Bucket in einer anderen AWS-Region repliziert werden.

Implementierungsschritte

1. Identifizieren Sie Datenquellen, die derzeit manuell gesichert werden. Weitere Details erhalten Sie unter REL09-BP01 Ermitteln und Sichern aller zu sichernden Daten oder Reproduzieren der Daten aus Quellen.
2. Bestimmen Sie das RPO für den Workload. Weitere Details erhalten Sie unter REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten.
3. Nutzen Sie eine automatisierte Backup-Lösung oder einen verwalteten Service. AWS Backup ist ein vollständig verwalteter Service, der die Zentralisierung und Automatisierung der Datensicherung über AWS-Services, in der Cloud und On-Premises erleichtert. Mithilfe von Backup-Plänen in AWS Backup erstellen Sie Regeln, die die zu sichernden Ressourcen und die Häufigkeit, mit der diese Backups erstellt werden sollen, festlegen. Diese Häufigkeit sollte auf dem in Schritt 2 festgelegten RPO basieren. Eine praktische Anleitung für die Erstellung automatisierter Backups mit AWS Backup finden Sie unter Testing Backup and Restore of Data. Native Backup-Funktionen werden von den meisten AWS-Services, die Daten speichern, angeboten. So kann beispielsweise RDS für automatisierte Backups mit zeitpunktbezogener Wiederherstellung (PITR) genutzt werden.
4. Für Datenquellen, die nicht von einer automatisierten Backup-Lösung oder einem verwalteten Service unterstützt werden, wie z. B. On-Premises-Datenquellen oder Warteschlangen, sollten Sie eine zuverlässige Lösung eines Drittanbieters verwenden, um automatische Backups zu erstellen. Als Alternative können Sie die Automatisierung für diesen Vorgang mit der AWS CLI oder mit SDKs erstellen. Sie können AWS Lambda-Funktionen oder AWS Step Functions nutzen, um die Logik für die Erstellung eines Backups von Daten zu definieren und Amazon EventBridge einsetzen, um diese in einer Häufigkeit entsprechend Ihren RPOs aufzurufen.

Aufwand für den Implementierungsplan: Niedrig.

Ressourcen

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Sicherung unterstützen können
- AWS Marketplace: Für die Sicherung geeignete Produkte
- Creating an EventBridge Rule That Triggers on a Schedule

- Was ist AWS Backup?
- Was ist AWS Step Functions?
- Was ist AWS Elastic Disaster Recovery?

Zugehörige Videos:

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

REL09-BP04 Verifizieren der Sicherungsintegrität und -verfahren durch regelmäßiges Wiederherstellen der Daten

Überprüfen Sie mit einem Wiederherstellungstest, ob sich mit Ihren Sicherungsverfahren das Recovery Time Objective (RTO) und das Recovery Point Objective (RPO) einhalten lassen.

Gewünschtes Ergebnis: Daten aus Backups werden regelmäßig mithilfe klar definierter Mechanismen wiederhergestellt, um zu überprüfen, ob die Wiederherstellung innerhalb des festgelegten Recovery Time Objective (RTO) für die Workload möglich ist. Überprüfen Sie, dass die Wiederherstellung aus einem Backup in eine Ressource erfolgt, die die Originaldaten enthält und dass keine dieser Daten korrupt oder nicht zugänglich sind, sowie dass sich der Datenverlust im Rahmen des Recovery Point Objective (RPO) bewegt.

Typische Anti-Muster:

- Wiederherstellung eines Backups ohne Abfrage oder Abruf von Daten, um zu überprüfen, ob die Wiederherstellung funktionsfähig ist.
- Es wird angenommen, dass ein Backup existiert.
- Es wird angenommen, dass das Backup eines System voll funktionsfähig ist und Daten daraus wiederhergestellt werden können.
- Es wird angenommen, dass die Zeit für das Wiederherstellen von Daten aus einem Backup innerhalb des RTO für die Workload liegt.
- Es wird angenommen, dass die im Backup enthaltenen Daten in das RPO für die Workload fallen.
- Wiederherstellung bei Bedarf, ohne ein Runbook zu verwenden oder außerhalb eines etablierten automatisierten Verfahrens.

Vorteile der Nutzung dieser bewährten Methode: Durch das Testen der Wiederherstellung der Backups wird überprüft, ob Daten bei Bedarf wiederhergestellt werden können, ohne befürchten zu müssen, dass Daten fehlen oder beschädigt sein könnten, ob die Wiederherstellung innerhalb des RTO für die Workload möglich ist und ob Datenverluste innerhalb des RPO für die Workload fallen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Das Testen der Sicherungs- und Wiederherstellungsfunktionen stärkt das Vertrauen in die Fähigkeit zur Durchführung dieser Aktionen während eines Ausfalls. Stellen Sie regelmäßig Backups an einem neuen Speicherort wieder her und führen Sie Tests aus, um die Datenintegrität zu überprüfen. Einige übliche Tests sind die Überprüfung, ob alle Daten verfügbar, nicht beschädigt und zugreifbar sind und ob ein Datenverlust innerhalb des RPO für die Workload liegt. Solche Tests können dabei helfen, zu ermitteln, ob die Wiederherstellungsmechanismen schnell genug sind, um dem RTO der Workload gerecht zu werden.

Mit AWS können Sie eine Testumgebung einrichten und Ihre Sicherungen wiederherstellen, um RTO- und RPO-Funktionen zu bewerten und Tests für Dateninhalte und Integrität durchzuführen.

Darüber hinaus ermöglichen Amazon RDS und Amazon DynamoDB eine zeitpunktbezogene Wiederherstellung (PITR). Durch die kontinuierliche Sicherung können Sie Ihren Datensatz in den Zustand zurücksetzen, in dem er sich an einem bestimmten Datum und zu einer bestimmten Uhrzeit befand.

Testen Sie, ob alle Daten verfügbar, nicht beschädigt und zugreifbar sind und ob ein Datenverlust innerhalb des RPO für die Workload liegt. Solche Tests können dabei helfen, zu ermitteln, ob die Wiederherstellungsmechanismen schnell genug sind, um dem RTO der Workload gerecht zu werden.

AWS Elastic Disaster Recovery bietet eine kontinuierliche, zeitpunktbezogene Wiederherstellung von Snapshots von Amazon EBS-Volumes. Bei der Replikation von Quellservern werden die Point-in-Time-Zustände auf der Grundlage der konfigurierten Richtlinie im Laufe der Zeit aufgezeichnet. Elastic Disaster Recovery hilft Ihnen, die Integrität dieser Snapshots zu überprüfen, indem Sie Instances zu Test- und Übungszwecken starten, ohne den Datenverkehr weiterzuleiten.

Implementierungsschritte

1. Identifizieren Sie Datenquellen, die derzeit gesichert werden, und den Speicherort dieser Backups. Eine Anleitung zur Implementierung finden Sie unter REL09-BP01 Ermitteln und Sichern aller zu sichernden Daten oder Reproduzieren der Daten aus Quellen.

2. Legen Sie für jede Datenquelle Kriterien für die Datenvalidierung fest. Verschieden Datentypen können unterschiedliche Eigenschaften aufweisen und somit auch unterschiedliche Validierungsmechanismen erfordern. Überlegen Sie, wie diese Daten validiert werden können, bevor Sie sie in der Produktion einsetzen. Häufig werden für die Datenvalidierung Daten- und Sicherungseigenschaften wie Datentyp, Format, Prüfsumme, Größe oder eine Kombination dieser Eigenschaften mit einer benutzerdefinierten Validierungslogik verwendet. Ein Beispiel hierfür wäre der Vergleich der Prüfsummenwerte zwischen der wiederhergestellten Ressource und der Datenquelle zum Zeitpunkt der Erstellung des Backups.
3. Bestimmen Sie das RTO und RPO für die Wiederherstellung der Daten auf der Grundlage der Kritikalität der Daten. Eine Anleitung zur Implementierung finden Sie unter REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten.
4. Bewerten Sie Ihre Wiederherstellungsfunktion. Prüfen Sie Ihre Sicherungs- und Wiederherstellungsstrategie, um festzustellen, ob sie Ihr RTO und RPO erfüllen kann, und passen Sie die Strategie bei Bedarf an. Mithilfe von AWSResilience Hub können Sie eine Bewertung Ihrer Workload durchführen. Dabei wird Ihre Anwendungskonfiguration im Hinblick auf die Ausfallsicherheitsrichtlinien bewertet und Sie erfahren, ob Ihre RTO- und RPO-Ziele erfüllt werden können.
5. Führen Sie eine Testwiederherstellung mit derzeit etablierten Prozessen durch, die in der Produktion für die Datenwiederherstellung verwendet werden. Diese Prozesse hängen davon ab, wie die ursprüngliche Datenquelle gesichert wurde sowie vom Format und der Speicherung des Backups selbst oder davon, ob die Daten aus anderen Quellen reproduziert werden. Wenn Sie beispielsweise einen verwalteten Service wie AWS Backup verwenden, kann es sich einfach um das Wiederherstellen des Backups auf einer neuen Ressource handeln. Wenn Sie AWS Elastic Disaster Recovery verwendet haben, können Sie eine Wiederherstellungsübung starten.
6. Überprüfen Sie die Datenwiederherstellung von der wiederhergestellten Ressource anhand von Kriterien, die Sie zuvor für die Datenvalidierung festgelegt haben. Enthalten die wiederhergestellten Daten den neuesten Datensatz bzw. das neueste Element zum Zeitpunkt des Backups? Fallen diese Daten in das RPO für die Workload?
7. Messen Sie den Zeitaufwand für die Wiederherstellung und vergleichen Sie ihn mit Ihrem festgelegten RTO. Ist dieser Prozess Teil des RTO für die Workload? Vergleichen Sie beispielsweise den Zeitstempel des Starts des Wiederherstellungsprozesses und des Abschlusses der Wiederherstellungsbewertung, um zu ermitteln, wie lange dieser Prozess dauert. Alle AWS API-Aufrufe sind mit einem Zeitstempel versehen und diese Informationen sind in AWS CloudTrail verfügbar. Während diese Informationen Details dazu liefern können, wann der Wiederherstellungsprozess gestartet wurde, sollte der End-Zeitstempel für den Abschluss der

Validierung von der Validierungslogik aufgezeichnet werden. Wenn Sie einen automatisierten Prozess verwenden, können Services wie Amazon DynamoDB verwendet werden, um diese Informationen zu speichern. Darüber hinaus können viele AWS-Services ein Ereignisprotokoll bereitstellen, das mit einem Zeitstempel versehene Informationen dazu enthält, wann bestimmte Aktionen aufgetreten sind. In AWS Backup werden Sicherungs- und Wiederherstellungsaktionen als Aufträge bezeichnet. Diese Aufträge enthalten Zeitstempelinformationen als Teil der Metadaten, anhand derer die für die Wiederherstellung benötigte Zeit gemessen werden kann.

8. Informieren Sie die Stakeholder, wenn die Datenvalidierung fehlschlägt oder wenn die für die Wiederherstellung benötigte Zeit das für die Workload festgelegte RTO überschreitet. Bei der Implementierung von Automatisierung zu diesem Zweck (wie in dieser Übung) können Services wie Amazon Simple Notification Service (Amazon SNS) verwendet werden, um Push-Benachrichtigungen wie E-Mail- oder SMS-Nachrichten an Stakeholder zu senden. Diese Nachrichten können auch in Messaging-Anwendungen wie Amazon Chime, Slack oder Microsoft Teams veröffentlicht werden oder zum Erstellen von Aufgaben als OpsItems mit AWS Systems Manager OpsCenter verwendet werden.
9. Automatisieren Sie diesen Prozess so, dass er regelmäßig ausgeführt wird. Sie können beispielsweise Services wie AWS Lambda oder einen Zustandsautomaten in AWS Step Functions nutzen, um die Wiederherstellungsprozesse zu automatisieren. Außerdem können Sie Amazon EventBridge verwenden, um diesen automatisierten Workflow regelmäßig aufzurufen, wie im folgenden Architekturdiagramm abgebildet. Erfahren Sie, wie Sie die Validierung der Datenwiederherstellung mit AWS Backup automatisieren. Darüber hinaus bietet dieses Well-Architected-Lab praktische Erfahrungen mit einer Möglichkeit, mehrere der hier beschriebenen Schritte zu automatisieren.

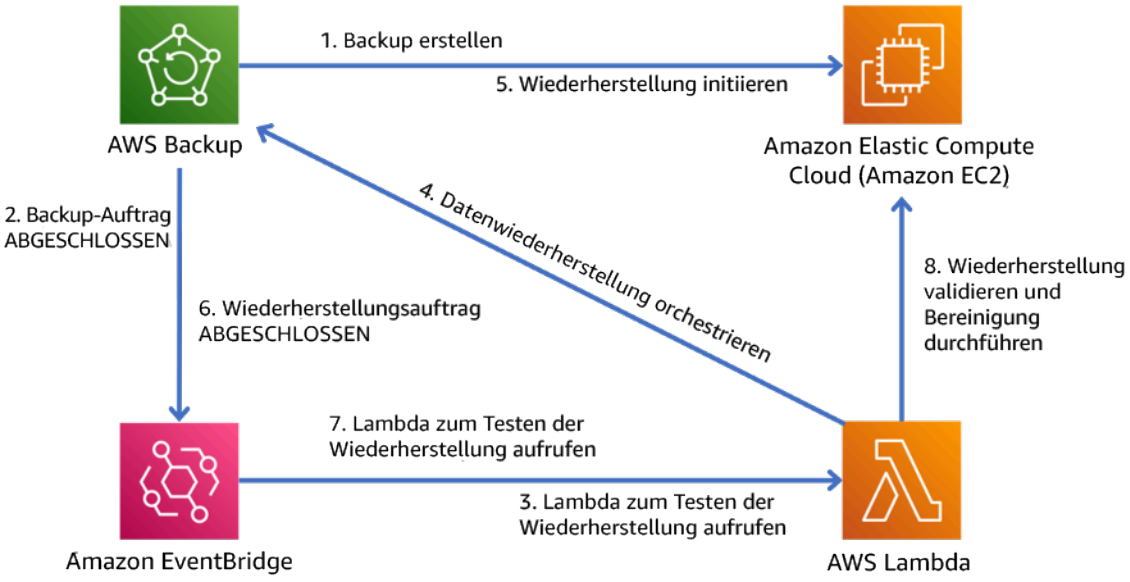


Abbildung 9. Ein automatisierter Sicherungs- und Wiederherstellungsprozess

Aufwand für den Implementierungsplan: Mittel bis hoch, abhängig von der Komplexität der Validierungskriterien.

Ressourcen

Zugehörige Dokumente:

- Validierung der Datenwiederherstellung mit AWS Backup automatisieren.
- APN-Partner: Partner, die Sie bei der Sicherung unterstützen können
- AWS Marketplace: Für die Sicherung geeignete Produkte
- Erstellen einer EventBridge-Regel, die nach einem Zeitplan ausgelöst wird
- On-Demand-Backup und Wiederherstellung für DynamoDB
- Was ist AWS Backup?
- Was ist AWS Step Functions?
- Was ist AWS Elastic Disaster Recovery
- AWS Elastic Disaster Recovery

Schützen von Workloads durch Fehlerisolierung

Die Fehlerisolierung begrenzt die Auswirkungen eines Komponenten- oder Systemausfalls auf eine definierte Grenze. Bei ordnungsgemäßer Isolierung sind Komponenten außerhalb der Grenze nicht

vom Ausfall betroffen. Wenn Sie Ihren Workload über mehrere Fehlerisolierungsgrenzen hinweg ausführen, kann er anfälliger für Ausfälle werden.

Best Practices

- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL10-BP03 Automatisierte Wiederherstellung für Komponenten, die auf einen einzelnen Standort beschränkt sind
- REL10-BP04 Verwenden von Bulkhead-Architekturen, um den Umfang von Beeinträchtigungen zu begrenzen

REL10-BP01 Bereitstellen des Workloads an mehreren Standorten

Verteilen Sie die Workload-Daten und -Ressourcen über mehrere Availability Zones oder ggf. über mehrere AWS-Regionen.

Ein grundlegendes Prinzip für das Servicedesign in AWS ist die Vermeidung von Single Points of Failure, einschließlich der zugrunde liegenden physischen Infrastruktur. AWS bietet Cloud-Computing-Ressourcen und -Services weltweit an mehreren geographischen Standorten, den sogenannten Regionen. Jede Region ist physisch und logisch unabhängig und besteht aus drei oder mehr Availability Zones (AZs). Availability Zones liegen geographisch nahe beieinander, sind jedoch physisch voneinander getrennt und isoliert. Wenn Sie Ihre Workloads auf Availability Zones und Regionen verteilen, minimieren Sie das Risiko von Bedrohungen wie Bränden, Überschwemmungen, wetterbedingten Katastrophen, Erdbeben und menschlichem Versagen.

Entwickeln Sie eine Standortstrategie, um eine hohe Verfügbarkeit zu gewährleisten, die für Ihre Workloads geeignet ist.

Gewünschtes Ergebnis: Produktionsworkloads werden auf mehrere Availability Zones (AZs) oder Regionen verteilt, um Fehlertoleranz und Hochverfügbarkeit zu erreichen.

Typische Anti-Muster:

- Ihr Produktionsworkload ist nur in einer einzelnen Availability Zone vorhanden.
- Sie implementieren eine Architektur mit mehreren Regionen, wenn eine Multi-AZ-Architektur die geschäftlichen Anforderungen erfüllen würde.
- Ihre Bereitstellungen oder Daten werden desynchronisiert, was zu Konfigurationsabweichungen oder unzureichend replizierten Daten führt.

- Sie berücksichtigen nicht die Abhängigkeiten zwischen Anwendungskomponenten, wenn sich die Anforderungen an Ausfallsicherheit und mehrere Standorte zwischen diesen Komponenten unterscheiden.

Vorteile der Nutzung dieser bewährten Methode:

- Ihre Workloads sind widerstandsfähiger gegen Vorfälle wie Strom- oder Umgebungssteuerungsausfälle, Naturkatastrophen, Upstream-Serviceausfälle oder Netzwerkprobleme, die sich auf eine AZ oder eine ganze Region auswirken.
- Sie können auf einen größeren Bestand an Amazon-EC2-Instances zugreifen und die Wahrscheinlichkeit von InsufficientCapacityExceptions (ICE) verringern, wenn Sie bestimmte EC2-Instance-Typen starten.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Hoch

Implementierungsleitfaden

Stellen Sie alle Produktionsworkloads in mindestens zwei Availability Zones (AZs) in einer Region bereit und führen Sie diese dort aus.

Verwenden mehrerer Availability Zones

Availability Zones sind Standorte für das Hosting von Ressourcen, die physisch voneinander getrennt sind, um korrelierte Ausfälle aufgrund von Risiken wie Bränden, Überschwemmungen und Tornados zu vermeiden. Jede Availability Zone verfügt über eine unabhängige physische Infrastruktur, einschließlich Stromversorgungsanschlüssen, Notstromquellen, mechanischen Services und Netzwerkkonnektivität. Dadurch bleiben Fehler in einer dieser Komponenten nur auf die betroffene Availability Zone beschränkt. Wenn beispielsweise aufgrund eines AZ-weiten Vorfalls EC2-Instances in der betroffenen Availability Zone nicht verfügbar sind, bleiben Ihre Instances in einer anderen Availability Zone weiterhin verfügbar.

Obwohl die Availability Zones in derselben AWS-Region physisch voneinander getrennt sind, sind sie einander nah genug, um Netzwerke mit hohem Durchsatz und niedriger Latenz (einstellige Millisekundenbeträge) bereitzustellen. Sie können Daten für die meisten Workloads synchron zwischen Availability Zones replizieren, ohne den Benutzerkomfort wesentlich zu beeinträchtigen. So können Sie Availability Zones in einer Aktiv/Aktiv- oder Aktiv/Standby-Konfiguration nutzen.

Die gesamte mit Ihren Workloads verbundene Computingleistung sollte auf mehrere Availability Zones verteilt werden. Dazu gehören Amazon-EC2-Instances, AWS Fargate-Aufgaben und VPC-

verbundene AWS Lambda-Funktionen. AWS-Computingservices, darunter EC2 Auto Scaling, Amazon Elastic Container Service (ECS) und Amazon Elastic Kubernetes Service (EKS), bieten Ihnen Möglichkeiten, Computingleistung in Availability Zones zu starten und zu verwalten. Konfigurieren Sie diese so, dass die Computingleistung bei Bedarf in einer anderen Availability Zone automatisch ersetzt wird, um die Verfügbarkeit aufrechtzuerhalten. Um Datenverkehr an verfügbare Availability Zones weiterzuleiten, platzieren Sie einen Load Balancer vor Ihrem Computer, z. B. einen Application Load Balancer oder Network Load Balancer. AWS Load Balancer können den Datenverkehr im Falle einer Beeinträchtigung der Availability Zone auf verfügbare Instances umleiten.

Sie sollten auch Daten für Ihre Workloads replizieren und sie in mehreren Availability Zones verfügbar machen. Einige AWS-verwaltete Datenservices wie Amazon S3 Amazon Elastic File Service (EFS), Amazon Aurora, Amazon DynamoDB, Amazon Simple Queue Service (SQS) und Amazon Kinesis Data Streams replizieren Daten standardmäßig in mehreren Availability Zones und bieten Widerstand gegen Beeinträchtigungen der Availability Zone. Bei anderen AWS-verwalteten Datenservices wie Amazon Relational Database Service (RDS), Amazon Redshift und Amazon ElastiCache müssen Sie die Multi-AZ-Replikation aktivieren. Nach der Aktivierung erkennen diese Services automatisch eine Beeinträchtigung der Availability Zone, leiten Anfragen an eine verfügbare Availability Zone weiter und replizieren Daten nach Bedarf nach der Wiederherstellung ohne Kundeneingriff erneut. Machen Sie sich mit dem Benutzerhandbuch für jeden von Ihnen verwendeten AWS-verwalteten Datenservice vertraut, um sich mit den Multi-AZ-Funktionen, Verhaltensweisen und Abläufen auszukennen.

Wenn Sie selbstverwalteten Speicher wie Amazon Elastic Block Store (EBS)-Volumes oder Amazon-EC2-Instance-Speicher verwenden, müssen Sie die Multi-AZ-Replikation selbst verwalten.

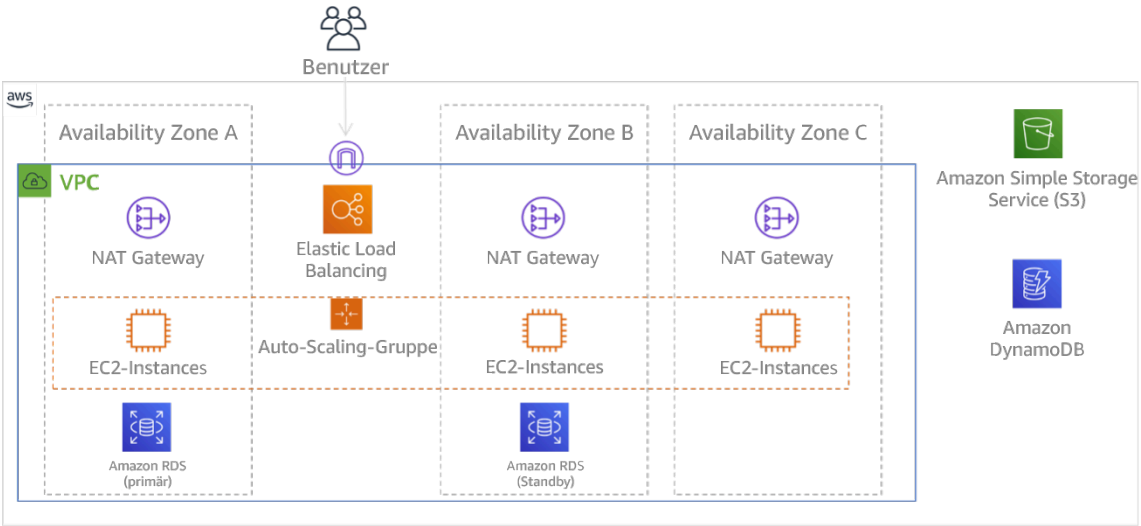


Abbildung 9: Mehrstufige Architektur, die in drei Availability Zones bereitgestellt wird. Amazon S3 und Amazon DynamoDB nutzen immer automatisch mehrere AZs. Auch der ELB wird in allen drei Zonen bereitgestellt.

Verwenden von mehreren AWS-Regionen

Wenn Sie über Workloads verfügen, die extreme Widerstandsfähigkeit erfordern (z. B. kritische Infrastrukturen, gesundheitsbezogene Anwendungen oder Services mit strengen kundenspezifischen oder gesetzlich vorgeschriebenen Verfügbarkeitsanforderungen), benötigen Sie möglicherweise zusätzliche Verfügbarkeit, die über das hinausgeht, was eine einzelne AWS-Region bieten kann. In diesem Fall sollten Sie Ihre Workloads auf mindestens zwei AWS-Regionen verteilen und dort ausführen (vorausgesetzt, dass Ihre Anforderungen an die Datenresidenz dies zulassen).

AWS-Regionen befinden sich in verschiedenen geographischen Regionen der Welt und auf mehreren Kontinenten. AWS-Regionen verfügen über eine noch stärkere physische Trennung und Isolierung als Availability Zones allein. AWS-Services, mit wenigen Ausnahmen, nutzen dieses Konzept, um völlig unabhängig zwischen verschiedenen Regionen zu operieren (auch bekannt als Regionalservices). Ein Ausfall eines Services in einer AWS-Region soll den Services in einer anderen Region nicht beeinträchtigen.

Wenn Sie Ihre Workloads in mehreren Regionen betreiben, sollten Sie zusätzliche Anforderungen berücksichtigen. Da Ressourcen in verschiedenen Regionen voneinander getrennt und unabhängig voneinander sind, müssen Sie die Komponenten Ihrer Workloads in jeder Region duplizieren. Dazu gehören neben Rechen- und Datenservices auch grundlegende Infrastrukturen wie VPCs.

HINWEIS: Wenn Sie ein multiregionales Design in Betracht ziehen, stellen Sie sicher, dass Ihre Workloads in einer einzigen Region ausgeführt werden können. Wenn Sie Abhängigkeiten zwischen Regionen erstellen, in denen eine Komponente in einer Region auf Services oder Komponenten in einer anderen Region angewiesen ist, könnt dies das Ausfallrisiko erhöhen und die Zuverlässigkeit erheblich beeinträchtigen.

Um multiregionale Bereitstellungen zu vereinfachen und die Konsistenz aufrechtzuerhalten, kann AWS CloudFormation StackSets Ihre gesamte AWS-Infrastruktur über mehrere Regionen hinweg replizieren. AWS CloudFormation kann auch Konfigurationsabweichungen erkennen und Sie informieren, wenn Ihre AWS-Ressourcen in einer Region nicht mehr synchron sind. Viele AWS-Services bieten die Replikation wichtiger Workload-Assets in mehreren Regionen. Beispielsweise kann EC2 Image Builder Ihre EC2-Machine-Images (AMIs) nach jedem Build in jeder Region, die Sie verwenden, veröffentlichen. Amazon Elastic Container Registry (ECR) kann Ihre Container-Images in Ihre ausgewählten Regionen replizieren.

Sie müssen Ihre Daten auch in jeder der von Ihnen ausgewählten Regionen replizieren. Viele AWS-verwaltete Datenservices bieten multiregionale Replikationsfunktionen, wie z. B. Amazon S3, Amazon DynamoDB, Amazon RDS, Amazon RDS, Amazon Aurora Amazon Aurora, Amazon Redshift, Amazon Elastiache und Amazon EFS. Globale Amazon DynamoDB-Tabellen akzeptieren Schreibvorgänge in jeder unterstützten Region und replizieren Daten zwischen allen Ihren anderen konfigurierten Regionen. Bei anderen Services müssen Sie eine primäre Region für Schreibvorgänge angeben, da andere Regionen schreibgeschützte Replikate enthalten. Informationen zu den einzelnen AWS-verwalteten Datenservices, die Ihre Workloads verwenden, finden Sie im zugehörigen Benutzer- und Entwicklerhandbuch, wo Sie mehr über die Funktionen und Einschränkungen für mehrere Regionen rfahren. Achten Sie besonders darauf, wohin Schreibvorgänge geleitet werden müssen, welche Transaktionsmöglichkeiten und Einschränkungen gelten, wie die Replikation durchgeführt wird, und wie die Synchronisation zwischen Regionen überwacht wird.

AWS bietet außerdem die Möglichkeit, den Anforderungsdatenverkehr mit großer Flexibilität an Ihre regionalen Bereitstellungen weiterzuleiten. Beispielsweise können Sie Ihre DNS-Datensätze mithilfe von Amazon Route 53 so konfigurieren, dass der Datenverkehr in die Region geleitet wird, die dem Benutzer am nächsten liegt. Alternativ können Sie Ihre DNS-Datensätze in einer Aktiv-/Standby-Konfiguration konfigurieren, in der Sie eine Region als primär festlegen und nur dann auf ein regionales Replikat zurückgreifen, wenn die primäre Region fehlerhaft wird. Sie können Route 53-Zustandsprüfungen so konfigurieren, dass sie fehlerhafte Endpunkte erkennen und einen automatischen Failover durchführen. Darüber hinaus können Sie Amazon Application Recovery Controller (ARC) verwenden, um eine hochverfügbare Routing-Steuerung für die manuelle Umleitung von Datenverkehr nach Bedarf bereitzustellen.

Auch wenn Sie sich dafür entscheiden, aus Gründen der Hochverfügbarkeit nicht in mehreren Regionen zu operieren, sollten Sie mehrere Regionen als Teil Ihrer Notfallwiederherstellungsstrategie (DR) in Betracht ziehen. Wenn möglich, replizieren Sie die Infrastrukturkomponenten und Daten Ihrer Workloads in einer Warm-Standby- oder Pilot-Light-Konfiguration in einer sekundären Region. Dabei replizieren Sie die Basisinfrastruktur aus der primären Region wie VPCs, Auto-Scaling-Gruppen, Container-Orchestratoren und andere Komponenten, konfigurieren aber die Komponenten variabler Größe in der Standby-Region (wie die Anzahl der EC2-Instances und Datenbankreplikate) so, dass sie eine minimal funktionsfähige Größe haben. Sie sorgen auch für eine kontinuierliche Datenreplikation von der primären Region zur Standby-Region. Wenn ein Vorfall eintritt, können Sie die Ressourcen in der Standby-Region skalieren oder vergrößern und sie dann zur primären Region heraufstufen.

Implementierungsschritte

1. Ermitteln Sie gemeinsam mit geschäftlichen Stakeholdern und Datenresidenzexperten die AWS-Regionen, die zum Hosten Ihrer Ressourcen und Daten verwendet werden können.
2. Bewerten Sie Ihre Workloads in Zusammenarbeit mit geschäftlichen und technischen Stakeholdern, um festzustellen, ob die Anforderungen an die Ausfallsicherheit durch einen Multi-AZ-Ansatz (einzelneAWS-Region) erfüllt werden können, oder ob ein multiregionaler Ansatz erforderlich ist (wenn mehrere Regionen zulässig sind). Die Verwendung mehrerer Regionen kann zu einer höheren Verfügbarkeit führen, jedoch auch zusätzliche Komplexität und Kosten mit sich bringen. Berücksichtigen Sie bei Ihrer Bewertung die folgenden Faktoren:
   a. Geschäftsziele und Kundenanforderungen: Welche Ausfallzeiten sind zulässig, wenn in einer Availability Zone oder einer Region ein Vorfall eintritt, der sich auf Workloads auswirkt? Bewerten Sie Ihre Ziele für den Wiederherstellungspunkt wie in REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten beschrieben.
   b. Anforderungen an die Notfallwiederherstellung (DR): Gegen welche Art von potenziellen Notfällen möchten Sie sich versichern? Ziehen Sie die Möglichkeit eines Datenverlusts oder einer langfristigen Nichtverfügbarkeit in unterschiedlichen Ausprägungen in Betracht, von einer einzelnen Availability Zone bis hin zu einer ganzen Region. Wenn Sie Daten und Ressourcen über Availability Zones hinweg replizieren und in einer einzelnen Availability Zone ein länger dauernder Ausfall auftritt, können Sie den Service in einer anderen Availability Zone wiederherstellen. Wenn Sie Daten und Ressourcen regionsübergreifend replizieren, können Sie den Service in einer anderen Region wiederherstellen.
3. Stellen Sie Ihre Computingressourcen in mehreren Availability Zones bereit.
   a. Erstellen Sie in Ihrer VPC mehrere Subnetze in verschiedenen Availability Zones. Konfigurieren Sie jedes Subnetz so, dass es groß genug ist, um die Ressourcen aufzunehmen, die zur Bewältigung des Workloads benötigt werden, auch bei einem Vorfall. Weitere Informationen finden Sie unter REL02-BP03 Berücksichtigen von Erweiterung und Verfügbarkeit bei der Zuweisung von IP-Adressen für Subnetze.
   b. Wenn Sie Amazon-EC2-Instances verwenden, verwenden Sie EC2 Auto Scaling, um Ihre Instances zu verwalten. Geben Sie die Subnetze an, die Sie im vorherigen Schritt ausgewählt haben, als Sie Ihre Auto-Scaling-Gruppen erstellt haben.
   c. Wenn Sie AWS Fargate Compute für Amazon ECS oder Amazon EKS verwenden, wählen Sie die Subnetze aus, die Sie im ersten Schritt ausgewählt haben, wenn Sie einen ECS-Service erstellen, eine ECS-Aufgabe starten oder ein Fargate-Profil für EKS erstellen.

   d. Wenn Sie AWS Lambda-Funktionen verwenden, die in Ihrer VPC ausgeführt werden müssen, wählen Sie die Subnetze aus, die Sie im ersten Schritt bei der Erstellung der Lambda-Funktion ausgewählt haben. AWS Lambda verwaltet die Verfügbarkeit für alle Funktionen, die keine VPC-Konfiguration haben, automatisch für Sie.

   e. Platzieren Sie Traffic Directors wie Load Balancer vor Ihren Computingressourcen. Wenn zonenübergreifendes Load Balancing aktiviert ist, erkennen AWSAp plication Load Balancer und Network Load Balancer, wenn Ziele wie EC2-Instances und Container aufgrund einer Beeinträchtigung der Availability Zone nicht erreichbar sind, und leiten den Datenverkehr zu Zielen in intakten Availability Zones um. Wenn Sie das zonenübergreifende Load Balancing deaktivieren, verwenden Sie Amazon Application Recovery Controller (ARC), um die Zonenverschiebungsfunktion bereitzustellen. Wenn Sie einen Load Balancer eines Drittanbieters verwenden oder Ihre eigenen Load Balancer implementiert haben, konfigurieren Sie diese mit mehreren Front-Ends in verschiedenen Availability Zones.

4. Replizieren Sie die Daten Ihrer Workloads in mehrere Availability Zones.

   a. Wenn Sie einen AWS-verwalteten Datenservice wie Amazon RDS, Amazon ElastiCache oder Amazon FSx verwenden, lesen Sie dessen Benutzerhandbuch, um mehr über seine Datenreplikations- und Ausfallsicherheitsfunktionen zu erfahren. Aktivieren Sie bei Bedarf AZ-übergreifende Replikation und Failover.

   b. Wenn Sie AWS-verwaltete Speicherservices wie Amazon S3, Amazon EFS und Amazon FSx verwenden, vermeiden Sie die Verwendung von Single-AZ- oder One-Zone-Konfigurationen für Daten, die eine hohe Dauerhaftigkeit erfordern. Verwenden Sie für diese Services eine Multi-AZ-Konfiguration. Prüfen Sie im Benutzerhandbuch des jeweiligen Services, ob die Multi-AZ-Replikation standardmäßig aktiviert ist, oder ob Sie sie aktivieren müssen.

   c. Wenn Sie eine selbstverwaltete Datenbank, eine Warteschlange oder einen anderen Speicherservice ausführen, organisieren Sie die Multi-AZ-Replikation gemäß den Anweisungen zu der Anwendung oder bewährten Methoden. Machen Sie sich mit den Failover-Verfahren für Ihre Anwendung vertraut.

5. Konfigurieren Sie Ihren DNS-Service so, dass er AZ-Beeinträchtigungen erkennt und den Datenverkehr in eine fehlerfreie Availability Zone umleitet. Amazon Route 53 kann dies automatisch tun, wenn es in Kombination mit Elastic Load Balancers verwendet wird. Route 53 kann auch mit Failover-Datensätzen konfiguriert werden, bei denen mithilfe von Integritätsprüfungen nur Anfragen mit fehlerfreien IP-Adressen beantwortet werden. Geben Sie für alle DNS-Datensätze, die für das Failover verwendet werden, einen kleinen TTL-Wert (Time to Live) an (z. B. 60 Sekunden oder weniger), um zu verhindern, dass das Zwischenspeichern von

Einträgen die Wiederherstellung behindert (Route 53-Aliaseinträge stellen die entsprechenden TTLs für Sie bereit).

Zusätzliche Schritte bei Verwendung mehrerer AWS-Regionen

1. Replizieren Sie den gesamten Betriebssystem- (OS) und Anwendungscode, der von Ihren Workloads verwendet wird, in den ausgewählten Regionen. Replizieren Sie bei Bedarf Amazon Machine Images (AMIs), die von Ihren EC2-Instances verwendet werden, mithilfe von Lösungen wie Amazon EC2 Image Builder. Replizieren Sie Container-Images, die in Registern gespeichert sind, mithilfe von Lösungen wie der regionsübergreifenden Replikation in Amazon ECR. Aktivieren Sie die regionale Replikation für alle Amazon-S3-Buckets, die zum Speichern von Anwendungsressourcen verwendet werden.
2. Stellen Sie Ihre Computingressourcen und Konfigurationsmetadaten (z. B. als im AWS Systems Manager Parameter Store gespeicherte Parameter) in mehreren Regionen bereit. Verwenden Sie dieselben Verfahren, die in den vorherigen Schritten beschrieben wurden, replizieren Sie jedoch die Konfiguration für jede Region, die Sie für Ihre Workloads verwenden. Verwenden Sie Infrastructure-as-Code-Lösungen wie AWS CloudFormation, um beispielsweise die Konfigurationen in allen Regionen einheitlich zu reproduzieren. Wenn Sie eine sekundäre Region in einer Pilot-Light-Konfiguration für die Notfallwiederherstellung verwenden, können Sie die Anzahl Ihrer Computingressourcen auf einen Mindestwert reduzieren, um Kosten zu sparen, was die Zeit bis zur Wiederherstellung entsprechend verlängert.
3. Replizieren Sie Ihre Daten aus Ihrer primären Region in Ihre sekundären Regionen.
    a. Globale Amazon-DynamoDB-Tabellen stellen globale Replikate Ihrer Daten bereit, in die aus jeder unterstützten Region geschrieben werden kann. Bei anderen AWS-verwalteten Datenservices wie Amazon RDS, Amazon Aurora und Amazon Elasticache legen Sie eine primäre (Lese-/Schreib-) Region und Replikatregionen (schreibgeschützt) fest. Einzelheiten zur regionalen Replikation finden Sie in den Benutzer- und Entwicklerhandbüchern der jeweiligen Services.
    b. Wenn Sie eine selbstverwaltete Datenbank verwenden, richten Sie die Replikation in mehreren Regionen gemäß den Anweisungen zur Anwendung oder zu den bewährten Methoden ein. Machen Sie sich mit den Failover-Verfahren für Ihre Anwendung vertraut.
    c. Wenn Ihre Workloads AWS EventBridge verwenden, müssen Sie möglicherweise ausgewählte Ereignisse aus Ihrer primären Region an Ihre sekundären Regionen weiterleiten. Geben Sie dazu Ereignisbusse in Ihren sekundären Regionen als Ziele für übereinstimmende Ereignisse in Ihrer Hauptregion an.

4. Überlegen Sie, ob und in welchem Umfang Sie in allen Regionen identische Verschlüsselungsschlüssel verwenden möchten. Ein typischer Ansatz, der Sicherheit und Benutzerfreundlichkeit in Einklang bringt, ist die Verwendung von regionsspezifischen Schlüsseln für für die Region lokale Daten und Authentifizierung und die Verwendung von Schlüsseln mit globalem Geltungsbereich für die Verschlüsselung von Daten, die zwischen verschiedenen Regionen repliziert werden. AWS Key Management Service (KMS) unterstützt Schlüssel für mehrere Regionen, um Schlüssel, die zwischen Regionen gemeinsam genutzt werden, sicher zu verteilen und zu schützen.
5. Ziehen Sie AWS Global Accelerator in Betracht, um die Verfügbarkeit Ihrer Anwendung zu verbessern, indem Sie den Datenverkehr in Regionen mit fehlerfreien Endpunkten umleiten.

Ressourcen

Zugehörige bewährte Methoden:

- REL02-BP03 Berücksichtigen von Erweiterung und Verfügbarkeit bei der Zuweisung von IP-Adressen für Subnetze
- REL11-BP05 Verhindern von bimodalem Verhalten mithilfe statischer Stabilität
- REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten

Zugehörige Dokumente:

- Globale AWS-Infrastruktur
- Whitepaper: Grenzen der AWS-Fehlerisolierung
- Ausfallsicherheit in Amazon EC2 Auto Scaling
- Amazon EC2 Auto Scaling: Beispiel: Aufteilen von Instances über mehrere Availability Zones
- So funktioniert EC2 Image Builder
- Wie Amazon ECS Aufgaben auf Container-Instances platziert (einschließlich Fargate)
- Ausfallsicherheit in AWS Lambda
- Amazon S3: Überblick über das Replizieren von Objekten
- Replikation privater Images in Amazon ECR
- Globale Tabellen: Multiregionale Replikation mit DynamoDB
- Amazon Elasticache for Redis OSS: Replikation über AWS-Regionen hinweg mit globalen Datenspeichern

- Ausfallsicherheit in Amazon RDS
- Verwenden von Amazon Aurora Global Databases
- AWS Global Accelerator – Entwicklerhandbuch
- Multiregionale Schlüssel in AWS KMS
- Amazon Route 53: Konfiguration des DNS-Failovers
- Amazon Application Recovery Controller (ARC) – Entwicklerhandbuch
- Senden und Empfangen von Amazon-EventBridge-Ereignissen zwischen AWS-Regionen
- Blogserie „Erstellen einer regionsübergreifenden Anwendung mit AWS-Services“
- Notfallwiederherstellung (DR)-Architektur in AWS, Teil I: Strategien für die Wiederherstellung in der Cloud
- Architektur für die Notfallwiederherstellung in AWS, Teil III: Pilot Light und Warm Standby

Zugehörige Videos:

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

REL10-BP03 Automatisierte Wiederherstellung für Komponenten, die auf einen einzelnen Standort beschränkt sind

Wenn Komponenten der Workload nur in einer einzigen Availability Zone oder in einem On-Premises-Rechenzentrum ausgeführt werden können, implementieren Sie die Möglichkeit, die Workload innerhalb Ihrer definierten Wiederherstellungsziele komplett neu aufzusetzen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Wenn die bewährte Methode zur Bereitstellung der Workload an mehreren Standorten aufgrund technologischer Einschränkungen nicht möglich ist, müssen Sie einen alternativen Pfad zur Ausfallsicherheit implementieren. Sie müssen die Möglichkeit automatisieren, die erforderliche Infrastruktur neu zu erstellen, Anwendungen neu bereitzustellen und die erforderlichen Daten für diese Fälle neu zu erstellen.

Amazon EMR startet beispielsweise alle Knoten für einen bestimmten Cluster in derselben Availability Zone, da die Ausführung eines Clusters in derselben Zone eine höhere Datenzugriffsrate bietet und

dadurch eine höhere Leistung für die Aufgabenbearbeitung bereitstellt. Wenn diese Komponente für die Ausfallsicherheit von Workloads erforderlich ist, müssen Sie die Möglichkeit haben, den Cluster und seine Daten erneut bereitzustellen. Für Amazon EMR sollten Sie nicht nur Multi-AZs verwenden, um für Redundanz zu sorgen. Sie können mehrere Knoten bereitstellen. Mit EMR File System (EMRFS) können Daten in EMR in Amazon S3 gespeichert und dann über mehrere Availability Zones oder AWS-Regionen repliziert werden.

Ähnlich wird Ihr Cluster bei Amazon Redshift standardmäßig in einer zufällig ausgewählten Availability Zone innerhalb der ausgewählten AWS-Region bereitgestellt. Alle Knoten des Clusters werden in derselben Zone bereitgestellt.

Für zustandsbehaftete serverbasierte Workloads, die in einem On-Premises-Rechenzentrum bereitgestellt werden, können Sie AWS Elastic Disaster Recovery verwenden, um Ihre Workloads in AWS zu schützen. Wenn Sie bereits in AWS gehostet sind, können Sie Elastic Disaster Recovery verwenden, um Ihre Workload in einer alternativen Availability Zone oder Region zu schützen. Elastic Disaster Recovery verwendet eine kontinuierliche Replikation auf Block-Ebene in eine schlanke Staging-Area, um eine schnelle, zuverlässige Wiederherstellung von On-Premises-Anwendungen und cloudbasierten Anwendungen zu gewährleisten.

Implementierungsschritte

1. Implementieren Sie die Selbstreparatur. Stellen Sie Ihre Instances oder Container nach Möglichkeit mit automatischer Skalierung bereit. Wenn dies nicht möglich ist, nutzen Sie für EC2-Instances die automatische Wiederherstellung oder implementieren Sie eine automatische Selbstreparatur basierend auf Amazon EC2- oder ECS-Container-Lebenszyklusereignissen.
   - Verwenden Sie Amazon EC2-Auto-Scaling-Gruppen für Instances und Container-Workloads, die keine IP-Adresse für eine einzelne Instance, keine private IP-Adresse, keine elastische IP-Adresse und keine Instance-Metadaten benötigen.
     - Die Benutzerdaten der Startvorlage können zur Implementierung einer Automatisierung verwendet werden, die die meisten Workloads automatisch reparieren kann.
   - Verwenden Sie die automatische Wiederherstellung von Amazon EC2-Instances für Workloads, die eine IP-Adresse für eine einzelne Instance, eine private IP-Adresse, eine elastische IP-Adresse und Instance-Metadaten benötigen.
     - Automatic Recovery sendet Benachrichtigungen zum Wiederherstellungsstatus an ein SNS-Thema, wenn der Instance-Fehler erkannt wird.

- Verwenden Sie Amazon EC2-Instance-Lebenszyklusereignisse bzw. Amazon ECS-Ereignisse für die Automatisierung der Selbstreparatur, wenn die automatische Skalierung oder EC2-Wiederherstellung nicht verwendet werden kann.
  - Verwenden Sie die Ereignisse, um die Automatisierung der Reparatur der Komponente entsprechend der erforderlichen Prozesslogik aufzurufen.
- Schützen Sie statusbehaftete Workloads, die auf einen einzelnen Standort beschränkt sind, mithilfe von AWS Elastic Disaster Recovery.

Ressourcen

Zugehörige Dokumente:

- Amazon ECS-Ereignisse
- Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling
- Wiederherstellen der Instance.
- Automatische Skalierung von Services
- Was ist Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP04 Verwenden von Bulkhead-Architekturen, um den Umfang von Beeinträchtigungen zu begrenzen

Implementieren Sie Bulkhead-Architekturen (zellenbasierte Architekturen), um die Auswirkungen von Fehlern innerhalb einer Workload auf eine begrenzte Anzahl von Komponenten zu beschränken.

Gewünschtes Ergebnis: Eine zellenbasierte Architektur verwendet mehrere isolierte Instances einer Workload, wobei jede Instance als Zelle bezeichnet wird. Jede Zelle ist unabhängig. Sie teilt ihren Status nicht mit anderen Zellen und bearbeitet eine Teilmenge der gesamten Workload-Anfragen. Dadurch werden die möglichen Auswirkungen eines Fehlers, z. B. eines fehlerhaften Software-Updates, auf eine einzelne Zelle und die von ihr verarbeiteten Anfragen reduziert. Wenn in einer Workload 10 Zellen für die Beantwortung von 100 Anfragen verwendet werden, sind bei einem Fehler 90 % der gesamten Anfragen nicht davon betroffen.

Typische Anti-Muster:

- Es wird ein unbegrenztes Wachstum der Zellen zugelassen.

- Code-Updates oder Bereitstellungen werden auf alle Zellen gleichzeitig angewandt.
- Status oder Komponenten werden von den Zellen geteilt (mit Ausnahme der Router-Schicht).
- Es werden komplexe Geschäfts- oder Routing-Logiken in die Routing-Schicht eingefügt.
- Es gibt keine Minimierung der zellenübergreifenden Interaktionen.

Vorteile der Nutzung dieser bewährten Methode: Bei zellenbasierten Architekturen sind viele gängige Fehlertypen in der Zelle selbst enthalten, was eine zusätzliche Fehlerisolierung ermöglicht. Diese Fehlergrenzen können die Widerstandsfähigkeit gegenüber Fehlertypen erhöhen, die sonst schwer einzudämmen sind, wie z. B. erfolglose Codebereitstellungen oder Anfragen, die beschädigt sind oder einen bestimmten Fehlermodus aufrufen (auch bekannt als Poison-Pill–Anfragen).

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Auf einem Schiff sorgen Schotten dafür, dass eine Beschädigung des Rumpfes auf einen Teil des Schiffes beschränkt bleibt. In komplexen Systemen wird dieses Muster oft kopiert, um eine Fehlerisolierung zu ermöglichen. Fehlerisolierte Grenzen beschränken die Auswirkungen eines Fehlers innerhalb einer Workload auf eine begrenzte Anzahl von Komponenten. Komponenten außerhalb der Grenze sind von dem Ausfall nicht betroffen. Durch die Verwendung mehrerer fehlerisolierter Grenzen können Sie die Auswirkungen auf Ihre Workload einschränken. Bei AWS können Kunden mehrere Availability Zones und Regionen verwenden, um eine Fehlerisolierung zu gewährleisten. Das Konzept der Fehlerisolierung lässt sich jedoch auch auf die Architektur Ihrer Workload ausweiten.

Die gesamte Workload wird durch einen Partitionsschlüssel in Zellen unterteilt. Dieser Schlüssel muss mit der Struktur des Service übereinstimmen oder mit der natürlichen Art und Weise, wie die Workload eines Service mit minimalen zellenübergreifenden Interaktionen unterteilt werden kann. Beispiele für Partitionsschlüssel sind die ID des Kunden, die ID der Ressource oder jeder andere Parameter, der in den meisten API-Aufrufen leicht zugänglich ist. Eine Schicht für das Routing von Zellen verteilt Anfragen auf der Grundlage des Partitionsschlüssels an einzelne Zellen und präsentiert den Kunden einen einzigen Endpunkt.

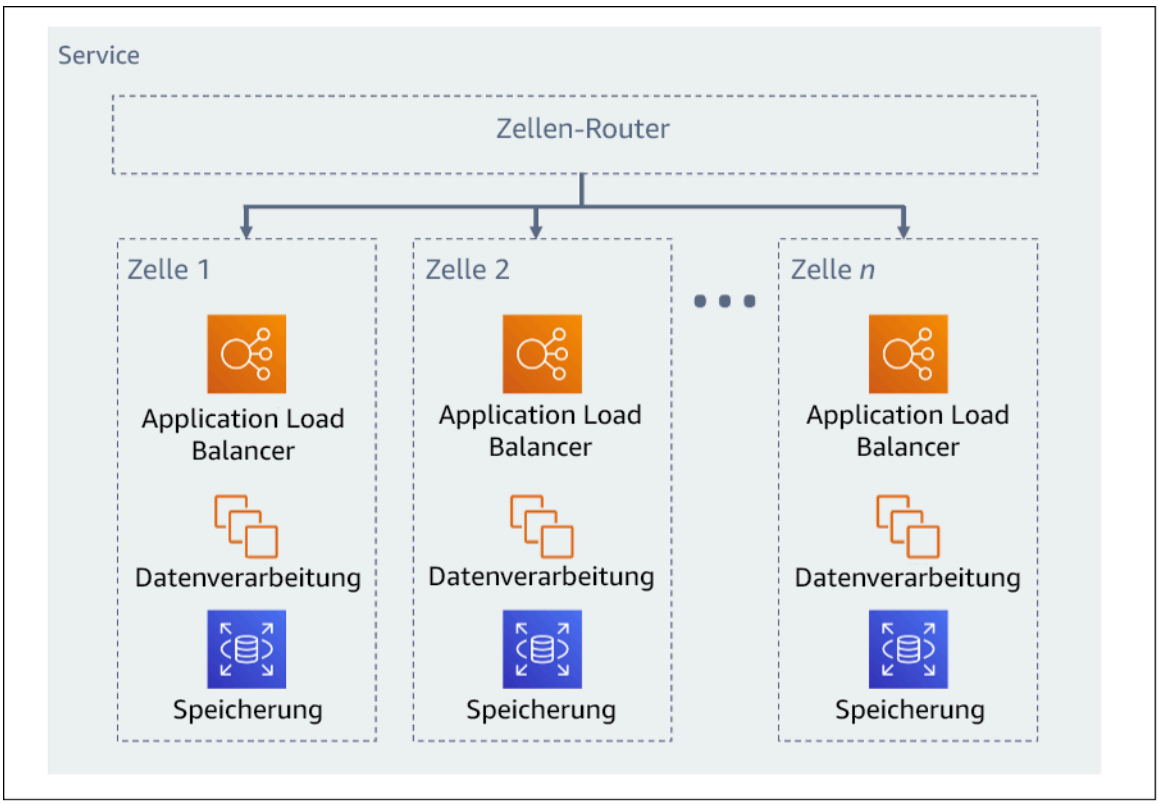


Abbildung 11: Zellenbasierte Architektur

Implementierungsschritte

Bei der Entwicklung einer zellenbasierten Architektur sind mehrere Designüberlegungen zu berücksichtigen:

1. Partitionsschlüssel: Bei der Auswahl des Partitionsschlüssels sollten besondere Überlegungen angestellt werden.
   - Er sollte mit der Struktur des Service übereinstimmen oder mit der natürlichen Art und Weise, wie die Workload eines Service mit minimalen zellenübergreifenden Interaktionen unterteilt werden kann. Beispiele sind `customer ID` oder `resource ID`.
   - Der Partitionsschlüssel muss in allen Anfragen verfügbar sein – entweder direkt oder in einer Weise, die sich durch andere Parameter leicht deterministisch ableiten lässt.
2. Persistente Zellenzuordnung: Upstream-Services sollten während des gesamten Lebenszyklus ihrer Ressourcen nur mit einer einzelnen Zelle interagieren.
   - Je nach Workload kann eine Strategie zur Migration von Zellen erforderlich sein, um Daten von einer Zelle in eine andere zu migrieren. Ein mögliches Szenario, in dem eine Migration von

Zellen erforderlich sein kann, ist, wenn ein bestimmter Benutzer oder eine bestimmte Ressource in Ihrer Workload zu groß wird und eine eigene Zelle benötigt.

- Zellen sollten keinen Status und keine Komponenten gemeinsam nutzen.
- Folglich sollten zellenübergreifende Interaktionen vermieden oder auf ein Minimum beschränkt werden, da diese Interaktionen Abhängigkeiten zwischen den Zellen schaffen und somit die Möglichkeiten zur Fehlerisolierung verringern.

3. Router-Schicht: Die Router-Schicht ist eine Komponente, die von Zellen gemeinsam genutzt wird, sodass nicht dieselbe Segmentierungsstrategie verfolgt werden kann wie bei Zellen.
   - Es wird empfohlen, dass die Routing-Schicht Anfragen auf einzelne Zellen verteilt, indem sie einen effizienten Algorithmus für die Zuordnung von Partitionen einsetzt – z. B. als die Kombination von kryptographischen Hash-Funktionen und einer modularen Arithmetik.
   - Um Auswirkungen auf mehrere Zellen zu vermeiden, muss die Routing-Schicht so einfach und horizontal skalierbar wie möglich bleiben, was den Verzicht auf eine komplexe Geschäftslogik innerhalb dieser Schicht erforderlich macht. Dies hat den zusätzlichen Nutzen, dass das erwartete Verhalten jederzeit leicht nachvollziehbar ist, was eine gründliche Testbarkeit ermöglicht. Wie Colm MacCárthaigh in Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee erklärt, sorgen einfache Designs und Muster mit konstanter Ausführung für zuverlässige Systeme und verringern die Widerstandsfähigkeit gegen Fragilität.
4. Zellgröße: Für Zellen sollte eine maximale Größe festgelegt sein, die sie nicht überschreiten dürfen.
   - Die maximale Größe sollte durch gründliche Tests ermittelt werden – bis Sollbruchstellen erreicht und sichere operative Margen etabliert sind. Weitere Details zur Implementierung von Testverfahren finden Sie unter REL07-BP04 Belastungstest Ihr Workload.
   - Die gesamte Workload sollte durch Hinzufügen zusätzlicher Zellen wachsen, sodass die Workload mit der steigenden Nachfrage skalieren kann.
5. Multi-AZ- oder multiregionale Strategien: Zum Schutz vor unterschiedlichen Ausfall-Domains sollten mehrere Resilienzebenen genutzt werden.
   - Für die Ausfallsicherheit sollten Sie einen Ansatz wählen, bei dem verschiedene Verteidigungsebenen aufgebaut werden. Eine Ebene schützt vor kleineren, häufiger auftretenden Unterbrechungen, indem eine hochverfügbare Architektur mit mehreren AZs erstellt wird. Eine weitere Verteidigungsebene schützt vor seltenen Ereignissen wie Naturkatastrophen mit großer Reichweite und Unterbrechungen auf Regionsebene. Für diese zweite Ebene muss die Architektur Ihrer Anwendung mehrere AWS-Regionen umfassen. Wenn Sie eine Multi-Region-Strategie für Ihre Workload implementieren, ist sie vor weitreichenden

Naturkatastrophen, die einen großen geografischen Bereich in einem Land betreffen, oder technischen Fehlern in einer ganzen Region geschützt. Beachten Sie dabei, dass das Implementieren einer Multi-Region-Architektur äußerst komplex sein kann und bei den meisten Workloads nicht erforderlich ist. Weitere Details erhalten Sie unter REL10-BP01 Bereitstellen des Workloads an mehreren Standorten.

6. Codebereitstellung: Eine Strategie zur gestaffelten Codebereitstellung sollte der gleichzeitigen Bereitstellung von Codeänderungen in allen Zellen vorgezogen werden.
   - Auf diese Weise werden mögliche Fehler in mehreren Zellen aufgrund einer fehlerhaften Bereitstellung oder menschlichen Versagens minimiert. Weitere Informationen finden Sie unter Automatisierung sicherer, vollautomatischer Bereitstellungen.

Ressourcen

Zugehörige bewährte Methoden:

- REL07-BP04 Belastungstest Ihr Workload
- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten

Zugehörige Dokumente:

- Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee
- AWS und Segmentierung
- Workload-Isolation mit Shuffle Sharding
- Automatisierung sicherer, vollautomatischer Bereitstellungen

Zugehörige Videos:

- AWS re:Invent 2018: Close Loops & Opening Minds: Wie man die Kontrolle über große und kleine Systeme übernimmt
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)
- Shuffle-Sharding: AWS re:Invent 2019: Einführung in die Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021 – Everything fails, all the time: Designing for resilience

Entwerfen von Workloads, die Komponentenausfälle verkraften

Workloads, für die eine hohe Verfügbarkeit und eine niedrige mittlere Reparaturzeit (MTTR) erforderlich ist, müssen auf Ausfallsicherheit ausgelegt sein.

Best Practices

- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler
- REL11-BP02 Failover zu fehlerfreien Ressourcen
- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen
- REL11-BP04 Nutzen der Datenebene und nicht der Steuerebene während der Wiederherstellung
- REL11-BP05 Verhindern von bimodalem Verhalten mithilfe statischer Stabilität
- REL11-BP06 Senden von Benachrichtigungen, wenn sich Ereignisse auf die Verfügbarkeit auswirken
- REL11-BP07 Architektur Ihres Produkts zur Erfüllung von Verfügbarkeitszielen und Uptime-SLAs (Service Level Agreements)

REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

Überwachen Sie den Zustand Ihrer Workload kontinuierlich, damit Sie und Ihre automatisierten Systeme auf Fehler oder Verschlechterungen aufmerksam werden, sobald diese auftreten. Überwachen Sie Key Performance Indicators (KPIs, wichtige Leistungskennzahlen) auf Grundlage des geschäftlichen Wertes.

Alle Wiederherstellungs- und Reparaturmechanismen müssen auf eine schnelle Erkennung von Problemen ausgelegt sein. Technische Fehler sollten zuerst erkannt werden, damit sie behoben werden können. Die Verfügbarkeit basiert jedoch auf der Fähigkeit Ihrer Workload, einen Unternehmenswert zu liefern. Daher müssen wichtige Leistungskennzahlen (KPIs), die dies messen, in Ihre Erkennungs- und Behebungsstrategie integriert sein.

Gewünschtes Ergebnis: Wesentliche Komponenten einer Workload werden unabhängig überwacht, um Fehler zu erkennen und anzuzeigen, wann und wo sie auftreten.

Typische Anti-Muster:

- Es sind keine Alarme konfiguriert, sodass Ausfälle ohne Benachrichtigung auftreten.
- Alarme sind vorhanden, aber mit Schwellenwerten, die keine ausreichende Zeit für die Reaktion bieten.

- Metriken werden nicht häufig genug erfasst, um das Recovery Time Objective (RTO) zu erreichen.
- Nur die kundenorientierten Schnittstellen der Workload werden aktiv überwacht.
- Es werden nur technische Metriken erfasst, keine Metriken für Geschäftsfunktionen.
- Es gibt keine Metriken, die die Benutzererfahrung der Workload messen.
- Es werden zu viele Überwachungen erstellt.

Vorteile der Nutzung dieser bewährten Methode: Mit einer angemessenen Überwachung auf allen Ebenen können Sie die Wiederherstellungszeit reduzieren, indem Sie die Zeit bis zur Erkennung verkürzen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Identifizieren Sie alle Workloads, die für die Überwachung überprüft werden sollen. Sobald Sie alle zu überwachenden Komponenten der Workload identifiziert haben, müssen Sie das Überwachungsintervall festlegen. Das Überwachungsintervall wirkt sich direkt darauf aus, wie schnell eine Wiederherstellung eingeleitet werden kann (abhängig davon, wie lange die Erkennung eines Fehlers dauert). Die Mittlere Zeit bis zur Erkennung ist die Zeitspanne zwischen dem Auftreten eines Fehlers und dem Beginn der Reparaturarbeiten. Die Liste der Services sollte umfassend und vollständig sein.

Die Überwachung muss alle Ebenen des Anwendungs-Stacks (inklusive Anwendung, Plattform, Infrastruktur und Netzwerk) abdecken.

Ihre Überwachungsstrategie sollte außerdem die Auswirkungen von grauen Fehlern berücksichtigen. Weitere Informationen zu grauen Fehlern finden Sie unter Graue Fehler im Whitepaper „Erweiterte Multi-AZ Resilience-Muster“.

Implementierungsschritte

- Das Überwachungsintervall hängt davon ab, wie schnell Wiederherstellungen durchgeführt werden müssen. Die Wiederherstellungszeit hängt davon ab, wie viel Zeit für eine Wiederherstellung benötigt wird. Daher müssen Sie die Häufigkeit der Erfassung bestimmen, indem Sie diese Zeit und das RTO einkalkulieren.
- Konfigurieren Sie eine detaillierte Überwachung für Komponenten und verwaltete Services.

  - Legen Sie fest, ob eine detaillierte Überwachung für EC2-Instances und Auto Scaling erforderlich ist. Eine detaillierte Überwachung liefert Metriken in einminütigen Intervallen, die Standardüberwachung liefert Metriken in fünfminütigen Intervallen.
  - Legen Sie fest, ob eine erweiterte Überwachung für RDS notwendig ist. Die erweiterte Überwachung verwendet einen Agenten auf RDS-Instances, um nützliche Informationen über verschiedene Prozesse oder Threads zu erhalten.
  - Ermitteln Sie die Überwachungsanforderungen kritischer Serverless-Komponenten für Lambda, API Gateway, Amazon EKS, Amazon ECS und alle Arten von Load Balancern.
  - Ermitteln Sie die Überwachungsanforderungen der Speicherkomponenten für Amazon S3, Amazon FSx, Amazon EFS und Amazon EBS.
- Erstellen Sie benutzerdefinierte Metriken, um Leistungskennzahlen (KPIs) zu messen. Workloads implementieren wichtige geschäftliche Funktionen, die als KPIs verwendet werden sollten, um zu erkennen, wann ein indirektes Problem auftritt.
- Überwachen Sie das Benutzererlebnis mithilfe von Benutzer-Canarys auf Fehler. Synthetische Transaktionstests (auch bekannt als Canary-Tests, aber nicht zu verwechseln mit Canary-Bereitstellungen), die das Kundenverhalten simulieren können, gehören zu den wichtigsten Testprozessen. Führen Sie diese Tests für Ihre Workload-Endpunkte konstant von verschiedenen Remote-Standorten aus.
- Erstellen Sie benutzerdefinierte Metriken zur Verfolgung des Benutzererlebnisses. Wenn Sie das Kundenerlebnis instrumentieren können, können Sie die Verschlechterung des Kundenerlebnisses feststellen.
- Richten Sie Alarme ein, um zu erkennen, wenn ein Teil Ihrer Workload nicht ordnungsgemäß funktioniert, und um anzuzeigen, wann die Ressourcen automatisch skaliert werden müssen. Alarme können visuell auf Dashboards angezeigt werden, Warnungen über Amazon SNS oder E-Mail versenden und mit Auto Scaling zusammenarbeiten, um Workload-Ressourcen hoch- oder herunterskalieren zu können.
- Erstellen Sie Dashboards, um Ihre Metriken zu visualisieren. Dashboards können verwendet werden, um Trends, Ausreißer und andere Indikatoren für potenzielle Probleme zu visualisieren, und auf Probleme hinweisen, die Sie untersuchen sollten.
- Erstellen Sie eine verteilte Ablaufverfolgungsüberwachung für Ihre Services. Mit der verteilten Überwachung können Sie nachvollziehen, wie Ihre Anwendung und die ihr zugrunde liegenden Services arbeiten, um die Ursache von Leistungsproblemen und Fehlern zu identifizieren und zu beheben.

- Erstellen Sie Dashboards und Datenerfassung für Überwachungssysteme (mit CloudWatch oder X-Ray) in einer separaten Region und einem separaten Konto.
- Bleiben Sie mit AWS Health über Serviceeinbußen auf dem Laufenden. Erstellen Sie maßgeschneiderte AWS Health-Ereignisbenachrichtigungen für E-Mail- und Chat-Kanäle über AWS-Benutzerbenachrichtigungen, und integrieren Sie sie programmgesteuert in Ihre Überwachungs- und Warnmeldungstools über Amazon EventBridge.

Ressourcen

Zugehörige bewährte Methoden:

- Definition der Verfügbarkeit
- REL11-BP06 Senden von Benachrichtigungen, wenn sich Ereignisse auf die Verfügbarkeit auswirken

Zugehörige Dokumente:

- Amazon CloudWatch Synthetics unterstützt Sie bei der Erstellung von Benutzer-Canarys
- Aktivieren oder Deaktivieren der detaillierten Überwachung für Ihre Instance
- Verbesserte Überwachung
- Überwachung Ihrer Auto-Scaling-Gruppen und -Instances mithilfe von Amazon CloudWatch
- Veröffentlichen von benutzerdefinierten Metriken
- Verwenden von Amazon CloudWatch Alarms
- Verwenden von CloudWatch Dashboards
- Verwenden von regionen- und kontoübergreifenden Amazon CloudWatch-Dashboards
- Verwenden der regionen- und kontoübergreifenden X-Ray-Nachverfolgung
- Verstehen der Verfügbarkeit

Zugehörige Videos:

- Beheben von grauen Fehlern

Zugehörige Beispiele:

- Workshop zur Beobachtbarkeit: X-Ray erkunden

Zugehörige Tools:

- CloudWatch
- CloudWatch X-Ray

REL11-BP02 Failover zu fehlerfreien Ressourcen

Wenn ein Fehler bei einer Ressource auftritt, sollten intakte Ressourcen weiterhin Anfragen bedienen. Stellen Sie sicher, dass Sie bei Standortbeeinträchtigungen (z. B. Availability Zone oder AWS-Region) über Systeme verfügen, die einen Failover auf intakte Ressourcen an nicht beeinträchtigten Standorten ermöglichen.

Wenn Sie einen Service entwerfen, verteilen Sie die Last auf Ressourcen, Availability Zones oder Regionen. So kann der Fehler einer einzelnen Ressource oder eine Beeinträchtigung durch die Verlagerung des Datenverkehrs auf die verbleibenden intakten Ressourcen aufgefangen werden. Überlegen Sie, wie Services im Falle eines Fehlers erkannt und geroutet werden.

Entwerfen Sie Ihre Services mit Blick auf die Fehlerbehebung. Bei AWS konzipieren wir Services mit dem Ziel, die Wiederherstellungszeit nach Ausfällen und die Auswirkungen auf Daten zu minimieren. Unsere Services verwenden primär Datenspeicher, die Anfragen erst akzeptieren, nachdem sie dauerhaft auf mehreren Replikaten in einer Region gespeichert wurden. Sie sind so aufgebaut, dass sie eine zellenbasierte Isolation und die Fehlerisolierung von Availability Zones nutzen. In unseren betrieblichen Abläufen setzen wir sehr stark auf Automatisierung. Außerdem optimieren wir unsere Funktionalität für Ersetzungsvorgänge und Neustarts, um nach Unterbrechungen eine schnelle Wiederherstellung zu ermöglichen.

Die Muster und Entwürfe, die den Failover ermöglichen, variieren für jeden AWS-Plattform-Service. Viele native verwaltete Services von AWS nutzen von Haus aus mehrere Availability Zones (wie Lambda oder API Gateway). Andere AWS-Services (wie EC2 und EKS) erfordern spezielle bewährte Methoden, um einen Failover von Ressourcen oder Datenspeichern über AZs hinweg zu unterstützen.

Es sollte eine Überwachung eingerichtet werden, um zu überprüfen, ob die Failover-Ressource in Ordnung ist, den Fortschritt der Failover-Ressourcen zu verfolgen und die Wiederherstellung von Geschäftsprozessen zu überwachen.

Gewünschtes Ergebnis: Die Systeme sind in der Lage, automatisch oder manuell neue Ressourcen zu verwenden, um sich von Störungen zu erholen.

Typische Anti-Muster:

- Die Planung für Fehler ist nicht Teil der Planungs- und Designphase.
- RTO und RPO sind nicht festgelegt.
- Unzureichende Überwachung, um ausfallende Ressourcen zu erkennen.
- Ordnungsgemäße Isolierung von fehlerhaften Domains.
- Multi-Region-Failover wird nicht berücksichtigt.
- Die Erkennung von Fehlern ist bei der Entscheidung für einen Failover zu empfindlich oder zu aggressiv.
- Failover-Design wird nicht getestet oder validiert.
- Durchführen automatischer Reparaturen ohne die Benachrichtigung, dass eine Reparatur erforderlich war.
- Fehlender Ausgleichszeitraum, um einen zu frühen Failover zu vermeiden.

Vorteile der Nutzung dieser bewährten Methode: Sie können widerstandsfähigere Systeme aufbauen, die auch bei Fehlern zuverlässig bleiben, indem sie ordnungsgemäß reduziert werden und sich schnell erholen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

AWS-Services wie Elastic Load Balancing und Amazon EC2 Auto Scaling helfen dabei, die Last auf Ressourcen und Availability Zones zu verteilen. Daher können der Ausfall einer einzelnen Ressource (wie etwa einer EC2-Instance) oder die Beeinträchtigung einer Availability Zone gemindert werden, indem Datenverkehr verlagert wird, um Ressourcen fehlerfrei zu halten.

Bei Workloads, die mehrere Regionen umfassen, sind Designs etwas komplizierter. Mit regionenübergreifenden Lesereplikaten können Sie beispielsweise Ihre Daten für mehrere AWS-Regionen bereitstellen. Der Failover ist jedoch immer noch erforderlich, um das Lesereplikat zum primären Endpunkt zu machen und den Datenverkehr auf den neuen Endpunkt zu lenken. Amazon Route 53, Amazon Application Recovery Controller (ARC), Amazon CloudFront und AWS Global Accelerator können dabei helfen, den Datenverkehr über AWS-Regionen hinweg weiterzuleiten.

AWS-Services wie Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge und Amazon DynamoDB

werden von AWS automatisch in mehreren Availability Zones bereitgestellt. Im Falle eines Fehlers leiten diese AWS-Services den Datenverkehr automatisch an intakte Standorte um. Die Daten werden redundant in mehreren Availability Zones gespeichert und bleiben verfügbar.

Für Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon EKS und Amazon ECS ist Multi-AZ eine Konfigurationsoption. AWS kann den Datenverkehr an die fehlerfreie Instance weiterleiten, wenn ein Failover eingeleitet wird. Diese Failover-Aktion kann von AWS oder auf Wunsch des Kunden durchgeführt werden.

Wählen Sie für Amazon-EC2-Instances, Amazon Redshift, Amazon-ECS-Aufgaben oder Amazon-EKS-Pods die Availability Zones für die Bereitstellung aus. Bei einigen Designs bietet Elastic Load Balancing die Lösung, um Instances in fehlerhaften Zonen zu erkennen und den Datenverkehr an die fehlerfreien Zonen weiterzuleiten. Elastic Load Balancing kann den Datenverkehr auch an Komponenten in Ihrem On-Premises-Rechenzentrum weiterleiten.

Für den Failover von Datenverkehr aus mehreren Regionen kann das Rerouting mit Amazon Route 53, Route 53 ARC, AWS Global Accelerator, Route 53 Private DNS für VPCs oder CloudFront eine Möglichkeit bieten, Internetdomains zu definieren und Routing-Richtlinien einschließlich Zustandsprüfungen zuzuweisen, um den Datenverkehr in intakte Regionen zu leiten. AWS Global Accelerator stellt statische IP-Adressen bereit, die als fester Einstiegspunkt für Ihre Anwendung fungieren und dann zu Endpunkten Ihrer Wahl in AWS-Regionen weitergeleitet werden, wobei das globale Netzwerk von AWS anstelle des Internets für eine bessere Leistung und Zuverlässigkeit genutzt wird.

Implementierungsschritte

- Erstellen Sie Failover-Designs für alle entsprechenden Anwendungen und Services. Isolieren Sie jede Komponente der Architektur und erstellen Sie Failover-Designs, die das RTO und RPO für jede Komponente erfüllen.
- Konfigurieren Sie weniger anspruchsvolle Umgebungen (wie Entwicklungs- oder Testumgebungen) mit allen Services, die für einen Failover-Plan erforderlich sind. Stellen Sie die Lösungen mit Infrastructure as Code (IaC) bereit, um die Reproduzierbarkeit sicherzustellen.
- Konfigurieren Sie einen Wiederherstellungsstandort, z. B. eine zweite Region, um die Failover-Designs zu implementieren und zu testen. Falls erforderlich, können die Ressourcen für die Tests vorübergehend konfiguriert werden, um die zusätzlichen Kosten zu begrenzen.
- Bestimmen Sie, welche Failover-Pläne durch AWS automatisiert sind, welche durch einen DevOps-Prozess automatisiert werden können und welche möglicherweise manuell sind. Dokumentieren und messen Sie die RTO- und RPO-Zeiten der einzelnen Services.

- Erstellen Sie ein Failover-Playbook, das alle Schritte zum Failover jeder Ressource, Anwendung und jedes Services enthält.
- Erstellen Sie ein Failback-Playbook, das alle Schritte zum Failback (mit Zeitangabe) für jede Ressource, jede Anwendung und jeden Service enthält.
- Erstellen Sie einen Plan, um das Playbook zu initiieren und zu proben. Verwenden Sie Simulationen und Chaostests, um die Schritte des Playbooks und die Automatisierung zu testen.
- Stellen Sie sicher, dass Sie bei einer Beeinträchtigung des Standorts (z. B. Availability Zone oder AWS-Region) über Systeme verfügen, die einen Failover auf intakte Ressourcen an nicht beeinträchtigten Standorten ermöglichen. Überprüfen Sie Kontingente, die automatische Skalierung und laufende Ressourcen vor dem Failover-Test.

Ressourcen

Zugehörige bewährte Methoden für Well-Architected:

- REL13 – Planen für DR
- REL10 – Schützen von Workloads durch Fehlerisolierung

Zugehörige Dokumente:

- Einstellen von RTO- und RPO-Zielen
- Failover mithilfe von gewichtetem Route 53-Routing
- Notfallwiederherstellung mit Amazon Application Recovery Controller
- EC2 mit automatischer Skalierung
- EC2-Bereitstellungen – Multi-AZ
- ECS-Bereitstellungen – Multi-AZ
- Wechseln des Datenverkehrs mithilfe von Amazon Application Recovery Controller
- Lambda mit einem Application Load Balancer und Failover
- ACM-Replikation und -Failover
- Parameter Store-Replikation und -Failover
- Regionsübergreifende ECR-Replikation und Failover
- Konfigurieren der regionsübergreifenden Replikation von Secrets Manager
- Aktivieren der regionsübergreifenden Replikation für EFS und Failover

- Regionsübergreifende EFS-Replikation und Failover
- Netzwerk-Failover
- S3-Endpunkt-Failover mit MRAP
- Erstellen einer regionsübergreifenden Replikation für S3
- Anleitung für regionsübergreifendes Failover und Graceful Failback auf AWS
- Failover mit Global Accelerator über mehrere Regionen
- Failover mit DRS

Zugehörige Beispiele:

- Notfallwiederherstellung in AWS
- Elastische Notfallwiederherstellung in AWS

REL11-BP03 Automatisieren der Reparatur auf allen Ebenen

Verwenden Sie bei Erkennung eines Fehlers automatisierte Funktionen, um Maßnahmen zur Behebung durchzuführen. Beeinträchtigungen können automatisch durch interne Service-Mechanismen behoben werden. Es kann aber auch erforderlich sein, Ressourcen neu zu starten oder Abhilfemaßnahmen durchzuführen.

Für selbstverwaltete Anwendungen und regionenübergreifende Korrekturen können Wiederherstellungskonzepte und automatisierte Korrekturprozesse aus bestehenden bewährten Methoden verwendet werden.

Die Möglichkeit, eine Ressource neu zu starten oder zu entfernen, ist ein wichtiges Instrument zur Behebung von Fehlern. Eine bewährte Methode besteht darin, Services nach Möglichkeit zustandslos zu betreiben. Dies verhindert den Datenverlust oder den Verlust der Verfügbarkeit bei einem Neustart der Ressource. In der Cloud können Sie (und sollten Sie üblicherweise) die gesamte Ressource (z. B. eine Datenverarbeitungs-Instance oder eine Serverless-Funktion) im Rahmen des Neustarts ersetzen. Der Neustart selbst ist eine einfache und zuverlässige Methode zur Wiederherstellung nach einem Ausfall. Bei Workloads treten viele verschiedene Arten von Fehlern auf. Fehler können bei Hardware, Software, Kommunikation und Betrieb auftreten.

Der Neustart oder Wiederholungsversuch gilt auch für Netzwerkanfragen. Nutzen Sie denselben Wiederherstellungsansatz für eine Netzwerk-Zeitüberschreitung und einen Abhängigkeitsfehler,

bei dem die Abhängigkeit einen Fehler ausgibt. Beide Ereignisse wirken sich in ähnlicher Weise auf das System aus. Statt also zu versuchen, aus den einzelnen Ereignissen einen „Sonderfall“ zu konstruieren, sollten Sie eine ähnliche Strategie anwenden und versuchen, ein exponentielles Backoff mit Jitter durchzuführen. Die Fähigkeit zum Neustart ist eine Funktion, die in wiederherstellungsorientierten Datenverarbeitungs- und Hochverfügbarkeits-Cluster-Architekturen empfohlen wird.

Gewünschtes Ergebnis: Automatisierte Aktionen werden durchgeführt, um die Erkennung eines Fehlers zu beheben.

Typische Anti-Muster:

- Bereitstellung von Ressourcen ohne automatische Skalierung.
- Einzelne Bereitstellung von Anwendungen in Instances oder Containern.
- Bereitstellen von Anwendungen, die nicht ohne automatische Wiederherstellung an mehreren Standorten bereitgestellt werden können.
- Manuelle Reparatur von Anwendungen, die sich mit Auto Scaling und einer automatischen Wiederherstellung nicht reparieren lassen.
- Keine Automatisierung beim Failover von Datenbanken.
- Keine automatisierten Methoden zur Umleitung des Datenverkehrs auf neue Endpunkte.
- Keine Speicherreplikation.

Vorteile der Nutzung dieser bewährten Methode: Eine automatisierte Korrektur kann die mittlere Zeit bis zur Wiederherstellung verkürzen und Ihre Verfügbarkeit verbessern.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Designs für Amazon EKS oder andere Kubernetes-Services sollten sowohl minimale und maximale Replikate oder zustandsbehaftete Sets als auch die minimale Größenanpassung von Clustern und Knotengruppen umfassen. Diese Mechanismen sorgen für ein Minimum an kontinuierlich verfügbaren Verarbeitungsressourcen und beheben gleichzeitig automatisch alle Fehler über die Steuerebene von Kubernetes.

Entwurfsmuster, auf die über einen Load Balancer mit Datenverarbeitungs-Clustern zugegriffen wird, sollten Auto-Scaling-Gruppen nutzen. Elastic Load Balancing (ELB) verteilt den eingehenden

Datenverkehr von Anwendungen automatisch auf mehrere Ziele und virtuelle Appliances in einer oder mehreren Availability Zones (AZs).

Bei Cluster-Datenverarbeitungs-Instances, die kein Load Balancing nutzen, sollte die Größe für den Verlust von mindestens einem Knoten ausgelegt sein. Auf diese Weise kann der Service mit möglicherweise reduzierter Kapazität weiterlaufen, während er einen neuen Knoten wiederherstellt. Beispielservices sind Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK und Amazon OpenSearch Service. Viele dieser Services können mit zusätzlichen Features zur Selbstheilung ausgestattet werden. Einige Cluster-Technologien müssen beim Verlust eines Knotens einen Alarm generieren, der einen automatisierten oder manuellen Workflow zur Wiederherstellung eines neuen Knotens auslöst. Dieser Workflow kann mit AWS Systems Manager automatisiert werden, um Probleme schnell zu beheben.

Mit Amazon EventBridge können Ereignisse wie CloudWatch-Alarme oder Statusänderungen in anderen AWS-Services überwacht und gefiltert werden. Auf der Grundlage von Ereignisinformationen kann es dann AWS Lambda, Systems Manager Automation oder andere Ziele aufrufen, um eine angepasste Wiederherstellungslogik für Ihre Workload auszuführen. Amazon EC2 Auto Scaling kann dafür konfiguriert werden, den Zustand der EC2-Instance zu prüfen. Wenn sich die Instance nicht im ausgeführten Status befindet oder der Systemstatus beeinträchtigt ist, betrachtet Amazon EC2 Auto Scaling die Instance als fehlerhaft und startet eine Ersatz-Instance. Bei Large-Scale-Ersetzungen (z. B. dem Verlust einer ganzen Availability Zone) ist für eine Hochverfügbarkeit die statische Stabilität vorzuziehen.

Implementierungsschritte

- Verwenden Sie Auto-Scaling-Gruppen, um Tiers in einem Workload bereitzustellen. Auto Scaling kann zustandslose Anwendungen selbst reparieren und Kapazitäten hinzufügen oder entfernen.
- Verwenden Sie Load Balancing für die zuvor genannten Datenverarbeitungs-Instances und wählen Sie den entsprechenden Load Balancer-Typ aus.
- Erwägen Sie die Reparatur für Amazon RDS. Konfigurieren Sie bei Standby-Instances das automatische Failover zur Standby-Instance. Bei Amazon RDS-Lesereplikaten ist ein automatisierter Workflow erforderlich, um ein Lesereplikat zur primären Instance zu machen.
- Implementieren Sie eine automatische Wiederherstellung für EC2-Instances, in denen Anwendungen bereitgestellt werden, die nicht an mehreren Standorten bereitgestellt werden können, und die einen Neustart nach Ausfällen tolerieren. Mithilfe der automatischen Wiederherstellung kann ausgefallene Hardware ersetzt und die Instance neu gestartet werden, wenn die Anwendung sich nicht an mehreren Standorten bereitstellen lässt. Die Metadaten der Instance und die zugehörigen IP-Adressen werden beibehalten, ebenso wie die EBS-Volumes und

Bindungspunkte für Amazon Elastic File Systems oder Dateisysteme für Lustre und Windows. Mit AWS OpsWorks können Sie die automatische Wiederherstellung von EC2-Instances auf Layer-Ebene konfigurieren.

- Implementieren Sie die automatische Wiederherstellung mit AWS Step Functions und AWS Lambda, wenn Sie keine automatische Skalierung oder automatische Wiederherstellung verwenden können oder wenn die automatische Wiederherstellung fehlschlägt. Wenn Sie keine automatische Skalierung verwenden können und die automatische Wiederherstellung entweder nicht genutzt werden kann oder fehlschlägt, können Sie die Reparatur mithilfe von AWS Step Functions und AWS Lambda automatisieren.
- Mit Amazon EventBridge können Ereignisse wie CloudWatch-Alarme oder Statusänderungen in anderen AWS-Services überwacht und gefiltert werden. Auf der Grundlage von Ereignisinformationen kann es dann AWS Lambda (oder andere Ziele) aufrufen, um eine angepasste Wiederherstellungslogik für Ihre Workload auszuführen.

Ressourcen

Zugehörige bewährte Methoden:

- Definition der Verfügbarkeit
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

Zugehörige Dokumente:

- Wie AWS Auto Scaling funktioniert
- Automatische Wiederherstellung in Amazon EC2
- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- Was ist Amazon FSx für Lustre?
- Was ist Amazon FSx für Windows File Server?
- AWS OpsWorks: Verwenden von Auto Healing zum Austausch fehlgeschlagener Instances
- Was ist AWS Step Functions?
- Was ist AWS Lambda?
- Was ist Amazon EventBridge?
- Verwenden von Amazon CloudWatch Alarms

- Amazon RDS-Failover
- SSM – Systems Manager Automation
- Bewährte Methoden für eine widerstandsfähige Architektur

Zugehörige Videos:

- Automatische Bereitstellung und Skalierung des OpenSearch-Services
- Automatisches Amazon RDS-Failover

Zugehörige Beispiele:

- Amazon RDS-Failover-Workshop

Zugehörige Tools:

- CloudWatch
- CloudWatch X-Ray

REL11-BP04 Nutzen der Datenebene und nicht der Steuerebene während der Wiederherstellung

Steuerebenen stellen die administrativen APIs zum Erstellen, Lesen und Schreiben, Aktualisieren, Löschen und Auflisten (CRUDL) von Ressourcen bereit, während Datenebenen den normalen Datenverkehr des Services abwickeln. Konzentrieren Sie sich bei der Implementierung von Wiederherstellungs- oder Abhilfemaßnahmen für Ereignisse, die sich möglicherweise auf die Ausfallsicherheit auswirken, auf eine minimale Anzahl von Operationen auf der Steuerebene, um den Service wiederherzustellen, zu skalieren, zu reparieren oder einen Failover durchzuführen. Aktionen auf der Datenebene sollten während dieser Beeinträchtigungen Vorrang vor allen anderen Aktivitäten haben.

Die folgenden Aktionen gehören beispielsweise alle zur Steuerebene: Starten einer neuen Datenverarbeitungs-Instance, Erstellen von Block-Speicher und Beschreiben von Warteschlangen-Services. Wenn Sie Datenverarbeitungs-Instances starten, muss die Steuerebene mehrere Aufgaben erfüllen, z. B. einen physischen Host mit Kapazität finden, Netzwerkschnittstellen zuweisen, lokale Block-Speicher-Volumes vorbereiten, Anmeldeinformationen generieren und Sicherheitsregeln hinzufügen. Steuerebenen neigen zu einer komplizierten Orchestrierung.

Gewünschtes Ergebnis: Wenn bei einer Ressource eine Störung auftritt, ist das System in der Lage, diese automatisch oder manuell zu beheben, indem es den Datenverkehr von gestörten auf intakte Ressourcen umleitet.

Typische Anti-Muster:

- Abhängigkeit von der Änderung von DNS-Einträgen, um den Datenverkehr umzuleiten.
- Abhängigkeit von Skalierungsoperationen auf Steuerebene, um beeinträchtigte Komponenten aufgrund einer unzureichenden Bereitstellung von Ressourcen zu ersetzen.
- Abhängigkeit von umfangreichen Aktionen auf der Steuerebene, in die mehrere Services und APIs involviert sind, um Störungen jeglicher Art zu beheben.

Vorteile der Nutzung dieser bewährten Methode: Eine höhere Erfolgsquote bei der automatisierten Behebung kann Ihre mittlere Zeit bis zur Wiederherstellung verkürzen und die Verfügbarkeit des Workloads verbessern.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel: Bei bestimmten Arten von Service-Einschränkungen sind Steuerebenen betroffen. Die Abhängigkeit von einer umfassenden Nutzung der Steuerebene für die Behebung kann die Wiederherstellungszeit (RTO) und die mittlere Zeit bis zur Wiederherstellung (MTTR) erhöhen.

Implementierungsleitfaden

Um die Aktionen auf der Datenebene zu begrenzen, bewerten Sie für jeden Service, welche Aktionen zur Wiederherstellung des Services erforderlich sind.

Nutzen Sie Amazon Application Recovery Controller, um den DNS-Datenverkehr zu verschieben. Diese Features überwachen kontinuierlich die Fähigkeit Ihrer Anwendung, nach Fehlern wiederhergestellt zu werden, und ermöglichen es Ihnen, die Wiederherstellung Ihrer Anwendung über mehrere AWS-Regionen, Availability Zones und On-Premises zu steuern.

Route 53-Routingrichtlinien verwenden die Steuerebene. Verlassen Sie sich also bei der Wiederherstellung nicht auf diese Ebene. Die Route 53-Datenebenen beantworten DNS-Abfragen und führen Zustandsprüfungen durch und werten diese aus. Sie werden weltweit vertrieben und sind für ein Service Level Agreement (SLA) mit 100-prozentiger Verfügbarkeit konzipiert.

Die Route 53-Verwaltungs-APIs und -Konsolen, über die Sie Route 53-Ressourcen erstellen, aktualisieren und löschen, arbeiten auf Steuerebenen, die so konzipiert sind, dass die starke Konsistenz und Stabilität, die Sie beim Verwalten von DNS benötigen, Priorität haben. Zu diesem

Zweck befinden sich die Steuerebenen in einer einzelnen Region, USA Ost (Nord-Virginia). Beide Systeme sind zwar äußerst zuverlässig, aber die Steuerebenen sind nicht in der SLA enthalten. In seltenen Fällen kann es vorkommen, dass das ausfallsichere Design der Datenebene es ermöglicht, die Verfügbarkeit aufrechtzuerhalten, während die Steuerebene dies nicht tut. Verwenden Sie für die Notfallwiederherstellung und Failover-Mechanismen Datenebenen-Funktionen, um die bestmögliche Zuverlässigkeit bereitzustellen.

Entwerfen Sie Ihre Computing-Infrastruktur so, dass sie statisch stabil ist, um zu vermeiden, dass die Steuerebene während eines Vorfalls verwendet wird. Wenn Sie beispielsweise Amazon-EC2-Instances verwenden, vermeiden Sie es, neue Instances manuell bereitzustellen oder Auto-Scaling-Gruppen anzuweisen, als Reaktion darauf Instances hinzuzufügen. Für ein Höchstmaß an Ausfallsicherheit stellen Sie ausreichende Kapazitäten in dem für den Failover verwendeten Cluster bereit. Wenn diese Kapazität begrenzt werden muss, legen Sie Drosselungen für das gesamte System fest, um den Gesamtdatenverkehr an die beschränkte Ressourcenmenge sicher zu begrenzen.

Bei Services wie Amazon DynamoDB, Amazon API Gateway, Load Balancern und AWS Lambda Serverless wird die Datenebene für diese Services genutzt. Die Erstellung neuer Funktionen, Load Balancers, API-Gateways oder DynamoDB-Tabellen ist jedoch eine Aktion auf der Steuerebene und sollte vor der Störung als Vorbereitung auf ein Ereignis und zum Üben von Failover-Aktionen durchgeführt werden. Für Amazon RDS ermöglichen Aktionen auf der Datenebene den Zugriff auf Daten.

Weitere Informationen über Datenebenen, Steuerebenen und wie AWS Services aufbaut, um Hochverfügbarkeitsziele zu erfüllen, finden Sie unter Statische Stabilität mithilfe von Availability Zones.

Erfahren Sie, welche Operationen auf der Datenebene und welche Operationen auf der Steuerebene ausgeführt werden.

Implementierungsschritte

Bewerten Sie für jede Workload, die nach einem Störfall wiederhergestellt werden muss, das Failover-Runbook, das Hochverfügbarkeitsdesign, das Auto Healing Design oder den Plan zur Wiederherstellung von HA-Ressourcen. Identifizieren Sie jede Aktion, die als Aktion auf der Steuerebene in Frage kommt.

Ziehen Sie in Erwägung, eine Aktion auf der Steuerebene in eine Aktion auf der Datenebene umzuwandeln:

- Auto Scaling (Steuerebene) im Vergleich zu vorab skalierten Amazon-EC2-Ressourcen (Datenebene)
- Instance-Skalierung mit Amazon EC2 (Steuerebene) im Vergleich zu AWS Lambda-Skalierung (Datenebene)
- Bewerten Sie alle Entwürfe unter Verwendung von Kubernetes und der Art der Aktionen auf der Steuerebene. Das Hinzufügen von Pods ist eine Aktion auf der Datenebene von Kubernetes. Aktionen sollten sich auf das Hinzufügen von Pods und nicht von Knoten beschränken. Die Verwendung von überdimensionierten Knoten ist die bevorzugte Methode zur Begrenzung von Aktionen auf der Steuerebene.

Ziehen Sie alternative Ansätze in Betracht, bei denen Aktionen auf der Datenebene dieselbe Maßnahme bewirken können.

- Änderung des Route-53-Datensatzes (Steuerebene) im Vergleich zu Amazon Application Recovery Controller (Datenebene)
- Route 53-Zustandsprüfungen für weitere automatisierte Aktualisierungen

Ziehen Sie einige Services in einer sekundären Region in Betracht, wenn der Service geschäftskritisch ist, um mehr Aktionen auf der Steuerebene und Datenebene in einer nicht betroffenen Region zu ermöglichen.

- Amazon EC2 Auto Scaling oder Amazon EKS in einer primären Region im Vergleich zu Amazon EC2 Auto Scaling oder Amazon EKS in einer sekundären Region und Weiterleitung des Datenverkehrs in eine sekundäre Region (Aktion auf der Steuerebene)
- Ein Lesereplikat in der sekundären primären Region erstellen oder Versuchen derselben Aktion in der primären Region (Aktion auf der Steuerebene)

Ressourcen

Zugehörige bewährte Methoden:

- Definition der Verfügbarkeit
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Automatisierung der Fehlertoleranz unterstützen können
- AWS Marketplace: Zur Erzielung von Fehlertoleranz geeignete Produkte
- Amazon Builders' Library: Vermeiden von Überlastungen verteilter Systeme durch Übernahme der Steuerung durch den kleineren Service
- Amazon DynamoDB API (Steuerebene und Datenebene)
- AWS Lambda-Aufteilungen (aufgeteilt in die Steuerebene und die Datenebene)
- AWS Elemental MediaStore Datenebene
- Entwickeln hoch resilienter Anwendungen mit Amazon Application Recovery Controller, Teil 1: Stack für eine einzelne Region
- Entwickeln hoch resilienter Anwendungen mit Amazon Application Recovery Controller, Teil 2: Stack für mehrere Regionen
- Erstellen von Mechanismen für die Notfallwiederherstellung mit Amazon Route 53
- Was ist Amazon Application Recovery Controller
- Kubernetes-Steuerebene und -Datenebene

Zugehörige Videos:

- Zurück zu den Basics – Verwendung statischer Stabilität
- Aufbau belastbarer Workloads an mehreren Standorten mit globalen AWS-Services

Zugehörige Beispiele:

- Einführung zu Amazon Application Recovery Controller
- Amazon Builders' Library: Vermeiden von Überlastungen verteilter Systeme durch Übernahme der Steuerung durch den kleineren Service
- Entwickeln hoch resilienter Anwendungen mit Amazon Application Recovery Controller, Teil 1: Stack für eine einzelne Region
- Entwickeln hoch resilienter Anwendungen mit Amazon Application Recovery Controller, Teil 2: Stack für mehrere Regionen
- Statische Stabilität mithilfe von Availability Zones

Zugehörige Tools:

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 Verhindern von bimodalem Verhalten mithilfe statischer Stabilität

Workloads sollten statisch stabil sein und nur in einem einzigen Normalmodus ausgeführt werden. Bimodales Verhalten liegt vor, wenn sich die Workload im Normalmodus und im Fehlermodus unterschiedlich verhält.

Sie können beispielsweise versuchen, nach einem Ausfall der Availability Zone eine Wiederherstellung durchzuführen, indem Sie neue Instances in einer anderen Availability Zone starten. Dies kann zu einer bimodalen Reaktion während eines Ausfallmodus führen. Stattdessen sollten Sie Workloads erstellen, die statisch stabil sind und nur in einem Modus betrieben werden. In diesem Beispiel hätten diese Instances vor dem Ausfall in der zweiten Availability Zone bereitgestellt werden sollen. Dieses statische Stabilitätsdesign verifiziert, dass die Workload nur in einem einzigen Modus ausgeführt wird.

Gewünschtes Ergebnis: Workloads zeigen im Normalmodus und im Fehlermodus kein bimodales Verhalten.

Typische Anti-Muster:

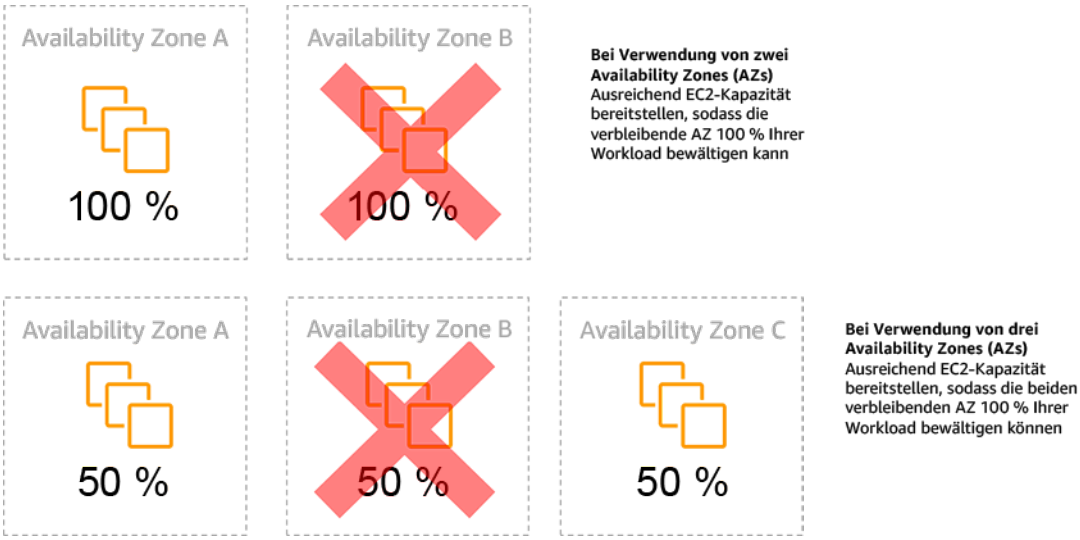
- Es wird davon ausgegangen, dass Ressourcen unabhängig vom Umfang des Fehlers immer bereitgestellt werden können.
- Während eines Fehlers wird versucht, dynamisch Ressourcen zu erwerben.
- Es werden keine ausreichenden Ressourcen für Zonen oder Regionen bereitgestellt, bis ein Fehler auftritt.
- Statische stabile Designs werden nur für Rechenressourcen in Erwägung gezogen.

Vorteile der Nutzung dieser bewährten Methode: Workloads, die mit statisch stabilen Designs ausgeführt werden, sind in der Lage, bei normalen Ereignissen und bei Ausfällen vorhersehbare Ergebnisse erzielen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Bimodales Verhalten bedeutet, dass eine Workload im normalen Modus und im Fehlermodus unterschiedliche Verhaltensweisen zeigt (z. B. Verlassen auf den Start neuer Instances bei Ausfall einer Availability Zone). Ein Beispiel für bimodales Verhalten ist, wenn stabile Amazon EC2-Designs genügend Instances in jeder Availability Zone bereitstellen, so dass die Verarbeitung der Workload auch beim Entfernen einer Availability Zone gewährleistet ist. Elastic Load Balancing oder Amazon Route 53 Health würde prüfen, ob eine Last von den beeinträchtigten Instances weg verlagert werden kann. Nachdem der Datenverkehr verlagert wurde, können Sie AWS Auto Scaling verwenden, um Instances in der ausgefallenen Zone asynchron zu ersetzen und sie in den fehlerfreien Zonen zu starten. Statische Stabilität für die Bereitstellung von Rechenleistung (z. B. EC2-Instances oder -Container) führt zu höchster Zuverlässigkeit.



Statische Stabilität von EC2-Instances über Availability Zones hinweg

Dies muss gegen die Kosten für dieses Modell und den geschäftlichen Nutzen der Aufrechterhaltung der Workload in allen Ausfallsituationen abgewogen werden. Es ist kostengünstiger, weniger Rechenkapazität bereitzustellen und bei einem Ausfall neue Instances zu starten. Bei großen Ausfällen (z. B. bei Beeinträchtigung einer Availability Zone oder Region) ist dieser Ansatz jedoch weniger effektiv, da er sowohl auf einer Betriebsebene als auch auf der Verfügbarkeit ausreichender Ressourcen in den nicht betroffenen Zonen oder Regionen beruht.

Ihre Lösung sollte die Anforderungen an die Zuverlässigkeit und Kosten für Ihre Workload gegeneinander abwägen. Ansätze mit statischer Stabilität gelten für eine Vielzahl von Architekturen, darunter Datenverarbeitungs-Instances, die über Availability Zones verteilt sind, Designs mit Lesereplikaten für Datenbanken, Kubernetes (Amazon EKS)-Clusterdesigns und Failover-Architekturen für mehrere Regionen.

Es ist auch möglich, ein statisch stabileres Design zu implementieren, indem mehr Ressourcen in jeder Zone verwendet werden. Wenn Sie eine größere Anzahl von Zonen hinzufügen, verringert sich die Menge der zusätzlichen Rechenleistung, die Sie für die statische Stabilität benötigen.

Ein weiteres Beispiel für bimodales Verhalten ist eine Netzwerk-Zeitüberschreitung, die dazu führen kann, dass ein System versucht, den Konfigurationsstatus des gesamten Systems zu aktualisieren. Dies kann zur unerwarteten Auslastung einer anderen Komponente führen, die daraufhin ausfallen könnte, was möglicherweise weitere unerwartete Konsequenzen nach sich zieht. Diese negative Feedback-Schleife wirkt sich auf die Verfügbarkeit Ihrer Workload aus. Deshalb sollten Sie stattdessen Systeme erstellen, die statisch stabil sind und nur in einem Modus betrieben werden. Ein statisch stabiles Design arbeitet konstant und aktualisiert den Konfigurationsstatus in regelmäßigen Abständen. Wenn ein Aufruf fehlschlägt, verwendet der Workload den zuvor zwischengespeicherten Wert und löst einen Alarm aus.

Ein weiteres Beispiel für bimodales Verhalten: Sie lassen zu, dass Clients im Fehlerfall den Workload-Cache umgehen. Dies scheint eine Lösung zu sein, die Clientanforderungen erfüllt, sie kann aber die Belastung Ihrer Workload erheblich ändern und führt wahrscheinlich zu Fehlern.

Bewerten Sie kritische Workloads, um festzustellen, für welche Workloads diese Art von Resilienzdesign erforderlich ist. Für diejenigen, die als kritisch eingestuft werden, muss jede Anwendungskomponente überprüft werden. Beispiele für Services, für die statische Stabilitätsbewertungen erforderlich sind:

- Datenverarbeitung: Amazon EC2, EKS-EC2, ECS-EC2, EMR-EC2
- Datenbanken: Amazon Redshift, Amazon RDS, Amazon Aurora
- Speicher: Amazon S3 (einzelne Zone), Amazon EFS (Mounts), Amazon FSx (Mounts)
- Load Balancer: bei bestimmten Designs

Implementierungsschritte

- Erstellen Sie Systeme, die statisch stabil sind und nur in einem einzigen Modus ausgeführt werden. Stellen Sie in diesem Fall in jeder Availability Zone oder Region genügend Instances bereit, um die Workload-Kapazität zu bewältigen, falls eine Availability Zone oder Region entfernt würde. Eine Vielzahl von Services kann für das Routing zu intakten Ressourcen verwendet werden, z. B.:
  - Regionsübergreifendes DNS-Routing
  - MRAP Amazon S3-Routing mit mehreren Regionen
  - AWS Global Accelerator

  - Amazon Application Recovery Controller
- Konfigurieren Sie Datenbank-Lesereplikate, um den Verlust einer einzelnen primären Instance oder eines Lesereplikats zu berücksichtigen. Wenn der Datenverkehr von Lesereplikaten bedient wird, sollte die Menge in jeder Availability Zone und jeder Region dem Gesamtbedarf im Fall eines Zonen- oder Regionsausfalls entsprechen.
- Konfigurieren Sie kritische Daten in einem Amazon S3-Speicher, der so konzipiert ist, dass er für die gespeicherten Daten beim Ausfall einer Availability Zone statisch stabil ist. Wenn die Amazon S3 One Zone-IA-Speicherklasse verwendet wird, sollte diese nicht als statisch stabil angesehen werden, da der Ausfall dieser Zone den Zugriff auf die zugehörigen gespeicherten Daten minimiert.
- Load Balancer sind manchmal falsch oder so konfiguriert, dass sie eine bestimmte Availability Zone bedienen. In diesem Fall könnte das statisch stabile Design darin bestehen, eine Workload über mehrere AZs in einem komplexeren Design zu verteilen. Das ursprüngliche Design kann aus Sicherheits-, Latenz- oder Kostengründen verwendet werden, um den Verkehr zwischen den Zonen zu reduzieren.

Ressourcen

Zugehörige bewährte Methoden für Well-Architected:

- Definition der Verfügbarkeit
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler
- REL11-BP04 Nutzen der Datenebene und nicht der Steuerebene während der Wiederherstellung

Zugehörige Dokumente:

- Minimierung der Abhängigkeiten bei der Planung der Notfallwiederherstellung
- Die Amazon Builders' Library: Statische Stabilität mithilfe von Availability Zones
- Grenzen für die Fehlerisolierung
- Statische Stabilität mithilfe von Availability Zones
- Mehrzonen-RDS
- Minimierung der Abhängigkeiten bei der Planung der Notfallwiederherstellung
- Regionsübergreifendes DNS-Routing
- MRAP Amazon S3-Routing mit mehreren Regionen

- AWS Global Accelerator
- Amazon Application Recovery Controller
- Amazon S3 (einzelne Zone)
- Zonenübergreifendes Load Balancing

Zugehörige Videos:

- Statische Stabilität in AWS: AWS re:Invent 2019: Einführung in die Amazon Builders' Library (DOP328)

REL11-BP06 Senden von Benachrichtigungen, wenn sich Ereignisse auf die Verfügbarkeit auswirken

Benachrichtigungen werden nach Erkennung von Schwellenwertüberschreitungen gesendet, auch wenn das durch das Ereignis verursachte Problem automatisch behoben wurde.

Auto Healing sorgt dafür, dass Ihre Workload zuverlässig ist. Allerdings können dadurch auch zugrunde liegende Probleme verschleiert werden, die behoben werden müssen. Implementieren Sie geeignete Überwachungsfunktionen und Ereignisse, damit Sie Problemmuster erkennen können, einschließlich solcher, die durch Auto Healing behoben werden. Auf diese Weise können Sie die Fehlerursachen beheben.

Resiliente Systeme sind so konzipiert, dass Verschlechterungsereignisse sofort an die entsprechenden Teams gemeldet werden. Diese Benachrichtigungen sollten über einen oder mehrere Kommunikationskanäle gesendet werden.

Gewünschtes Ergebnis: Bei Überschreitung von Schwellenwerten wie Fehlerraten, Latenz oder anderen kritischen Leistungsindikatoren (KPIs) werden sofort Benachrichtigungen an die Betriebsteams gesendet, sodass diese Probleme so schnell wie möglich behoben und Auswirkungen auf die Benutzer vermieden oder minimiert werden.

Typische Anti-Muster:

- Es werden zu viele Alarme gesendet.
- Es werden Alarme gesendet, die keine Maßnahmen erfordern.
- Die Schwellenwerte für den Alarm sind zu hoch (überempfindlich) oder zu niedrig (nicht empfindlich genug).

- Es werden keine Alarme für externe Abhängigkeiten gesendet.
- Graue Fehler werden bei der Planung von Überwachung und Alarmen nicht berücksichtigt.
- Es werden automatische Reparaturen ausgeführt, ohne das entsprechende Team darüber zu benachrichtigen, dass eine Reparatur erforderlich war.

Vorteile der Nutzung dieser bewährten Methode: Durch Benachrichtigungen über die Wiederherstellung werden Betriebs- und Geschäftsteams über Service-Einschränkungen informiert, sodass sie sofort reagieren können, um sowohl die mittlere Zeit zur Erkennung (Mean Time to Detect, MTTD) als auch die mittlere Wiederherstellungszeit (Mean Time to Repair, MTTR) zu minimieren. Benachrichtigungen zu Wiederherstellungen stellen sicher, dass Sie selten auftretende Probleme nicht ignorieren.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel. Wenn keine geeigneten Überwachungsfunktionen und Mechanismen zur Benachrichtigung bei Ereignissen implementiert werden, kann dies dazu führen, dass Problemmuster nicht erkannt werden, einschließlich solcher, die durch Auto Healing behoben werden. Ein Team wird nur dann auf eine Verschlechterung des Systems aufmerksam gemacht, wenn Benutzer den Kundendienst kontaktieren oder der Fehler zufällig bemerkt wird.

Implementierungsleitfaden

Bei der Definition einer Überwachungsstrategie ist ein ausgelöster Alarm ein häufiges Ereignis. Dieses Ereignis würde wahrscheinlich eine Kennung für den Alarm enthalten, den Alarmstatus (z. B. `IN ALARM` oder `OK`) und Einzelheiten darüber, was ihn ausgelöst hat. In vielen Fällen sollte ein Alarmereignis erkannt und eine E-Mail-Benachrichtigung gesendet werden. Dies ist ein Beispiel für eine Aktion bei einem Alarm. Die Alarmbenachrichtigung ist für die Beobachtbarkeit von entscheidender Bedeutung, da hiermit die richtigen Personen darüber informiert werden, dass ein Problem vorliegt. Wenn die Aktionen bei Ereignissen in Ihrer Lösung für die Beobachtbarkeit ausgereift sind, kann das Problem automatisch behoben werden, ohne dass menschliches Eingreifen erforderlich ist.

Sobald Alarme zur KPI-Überwachung eingerichtet wurden, sollten die entsprechenden Teams Warnmeldungen erhalten, wenn Schwellenwerte überschritten werden. Diese Warnungen können auch verwendet werden, um automatisierte Prozesse auszulösen, die versuchen, die Verschlechterung zu beheben.

Für eine komplexere Schwellenwertüberwachung sollten zusammengesetzte Alarme in Betracht gezogen werden. Zusammengesetzte Alarme verwenden eine Reihe von Alarmen zur KPI-

Überwachung, um eine Warnung auf Grundlage der Geschäftslogik zu erstellen. CloudWatch-Alarme können so konfiguriert werden, dass E-Mails gesendet oder Vorfälle mithilfe der Amazon SNS-Integration oder Amazon EventBridge in Drittanbietersystemen zur Nachverfolgung von Vorfällen protokolliert werden.

Implementierungsschritte

Erstellen Sie verschiedene Arten von Alarmen, je nachdem, wie Workloads überwacht werden, z. B.:

- Anwendungsalarme werden verwendet, um zu erkennen, wenn ein Teil der Workload nicht ordnungsgemäß funktioniert.
- Infrastrukturalarme geben an, wann Ressourcen skaliert werden müssen. Alarme können visuell auf Dashboards angezeigt werden, Warnungen über Amazon SNS oder E-Mail versenden und mit Auto Scaling zusammenarbeiten, um Workload-Ressourcen hoch- oder herunterskalieren zu können.
- Einfache statische Alarme können erstellt werden, um zu überwachen, wann eine Metrik für eine bestimmte Anzahl von Bewertungszeiträumen einen statischen Schwellenwert überschreitet.
- Zusammengesetzte Alarme können komplexe Alarme aus mehreren Quellen berücksichtigen.
- Nachdem der Alarm erstellt wurde, erstellen Sie entsprechende Benachrichtigungsereignisse. Sie können eine Amazon SNS-API direkt aufrufen, um Benachrichtigungen zu senden und Automatisierung zur Problembehebung oder Kommunikation zu verknüpfen.
- Bleiben Sie mit AWS Health über Serviceeinbußen auf dem Laufenden. Erstellen Sie maßgeschneiderte AWS Health-Ereignisbenachrichtigungen für E-Mail- und Chat-Kanäle über AWS-Benutzerbenachrichtigungen, und integrieren Sie sie programmgesteuert in Ihre Überwachungs- und Warnmeldungstools über Amazon EventBridge.

Ressourcen

Zugehörige bewährte Methoden für Well-Architected:

- Definition der Verfügbarkeit

Zugehörige Dokumente:

- Erstellen eines CloudWatch-Alarms basierend auf einem statischen Schwellenwert
- Was ist Amazon EventBridge?

- Was ist Amazon Simple Notification Service?
- Veröffentlichen von benutzerdefinierten Metriken
- Verwenden von Amazon CloudWatch Alarms
- Einrichten von zusammengesetzten CloudWatch-Alarmen
- Neuheiten im Bereich AWS-Beobachtbarkeit bei der re:Invent 2022

Zugehörige Tools:

- CloudWatch
- CloudWatch X-Ray

REL11-BP07 Architektur Ihres Produkts zur Erfüllung von Verfügbarkeitszielen und Uptime-SLAs (Service Level Agreements)

Erstellen Sie Ihr Produkt so, dass Verfügbarkeit und Betriebszeiten laut SLAs (Service Level Agreements) erfüllt werden. Wenn Sie Verfügbarkeitsziele oder Uptime-SLAs veröffentlichen oder privat vereinbaren, stellen Sie sicher, dass Ihre Architektur und Ihre operativen Prozesse so konzipiert sind, dass sie diese unterstützen.

Gewünschtes Ergebnis: Jede Anwendung hat ein definiertes Verfügbarkeitsziel und ein SLA für Leistungskennzahlen, die überwacht und verwaltet werden können, um die Geschäftsergebnisse zu erreichen.

Typische Anti-Muster:

- Entwurf und Bereitstellung von Workloads ohne Festlegung von SLAs.
- SLA-Metriken werden ohne Begründung oder geschäftliche Anforderungen zu hoch angesetzt.
- SLAs werden ohne Berücksichtigung von Abhängigkeiten und den ihnen zugrunde liegenden SLAs festgelegt.
- Anwendungsdesigns werden ohne Berücksichtigung des Modells der geteilten Verantwortung für die Ausfallsicherheit erstellt.

Vorteile der Nutzung dieser bewährten Methode: Die Entwicklung von Anwendungen auf der Grundlage wichtiger Ausfallsicherheitsziele hilft Ihnen dabei, Ihre Geschäftsziele und

Kundenerwartungen zu erfüllen. Diese Ziele sind die Grundlage für die Entwicklung von Anwendungen, bei der verschiedene Technologien bewertet und verschiedene Kompromisse in Betracht gezogen werden.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Bei der Entwicklung von Anwendungen müssen Sie eine Reihe von Anforderungen berücksichtigen, die sich aus geschäftlichen, operativen und finanziellen Zielen ergeben. Im Rahmen der operativen Anforderungen müssen für Workloads spezifische Metriken für die Ausfallsicherheit festgelegt werden, damit sie angemessen überwacht und unterstützt werden können. Die Metriken für die Ausfallsicherheit sollten nicht nach der Bereitstellung der Workload festgelegt oder ermittelt werden. Sie sollten in der Entwurfsphase festgelegt werden und als Leitlinien für verschiedene Entscheidungen und Abwägungen dienen.

- Jede Workload sollte ihre eigenen Metriken für die Ausfallsicherheit haben. Diese Metriken können sich von anderen geschäftlichen Anwendungen unterscheiden.
- Die Reduzierung von Abhängigkeiten kann sich positiv auf die Verfügbarkeit auswirken. Jede Workload sollte ihre Abhängigkeiten und deren SLAs berücksichtigen. Wählen Sie im Allgemeinen Abhängigkeiten mit Verfügbarkeitszielen aus, die den Zielen Ihrer Workload entsprechen oder höher sind.
- Ziehen Sie eine lose Verkoppelung in Betracht, damit Ihre Workload trotz der Beeinträchtigung durch Abhängigkeiten korrekt arbeiten kann, sofern dies möglich ist.
- Reduzieren Sie die Abhängigkeiten auf der Steuerebene, insbesondere während der Wiederherstellung oder einer Beeinträchtigung. Evaluieren Sie Designs, die für geschäftskritische Workloads statisch stabil sind. Nutzen Sie den sparsamen Umgang mit Ressourcen, um die Verfügbarkeit dieser Abhängigkeiten in einer Workload zu erhöhen.
- Die Beobachtbarkeit und die Instrumentierung sind entscheidend für das Erreichen von SLAs. Sie reduzieren die Mean Time to Detection (MTTD) und die Mean Time to Repair (MTTR).
- Weniger häufige Störungen (längere MTBF), kürzere Fehlererkennungszeiten (kürzere MTTD) und kürzere Reparaturzeiten (kürzere MTTR) sind die drei Faktoren, die zur Verbesserung der Verfügbarkeit in verteilten Systemen eingesetzt werden.
- Das Festlegen und Einhalten von Metriken für die Ausfallsicherheit einer Workload ist eine der Grundlagen für jedes effektive Design. Diese Designs müssen Kompromisse in Bezug auf Designkomplexität, Service-Abhängigkeiten, Leistung, Skalierung und Kosten berücksichtigen.

Implementierungsschritte

- Überprüfen und dokumentieren Sie das Workload-Design unter Berücksichtigung der folgenden Fragen:
  - Wo werden die Steuerebenen in der Workload verwendet?
  - Wie implementiert die Workload die Ausfallsicherheit?
  - Wie sehen die Designmuster für die Skalierung, automatische Skalierung, Redundanz und hochverfügbare Komponenten aus?
  - Welche Anforderungen gibt es an die Datenkonsistenz und -verfügbarkeit?
  - Gibt es Überlegungen zur sparsamen Nutzung von Ressourcen oder zur statischen Stabilität von Ressourcen?
  - Welche Abhängigkeiten bestehen zwischen den Services?
- Definieren Sie in Zusammenarbeit mit den Stakeholdern SLA-Metriken auf der Grundlage der Workload-Architektur. Berücksichtigen Sie die SLAs aller Abhängigkeiten, die die Workload nutzt.
- Sobald das SLA-Ziel festgelegt ist, optimieren Sie die Architektur, um die SLA zu erfüllen.
- Sobald das Design festgelegt ist, das die SLA erfüllt, implementieren Sie operative Änderungen, Prozessautomatisierungen und Runbooks, die ebenfalls auf die Reduzierung von MTTD und MTTR ausgerichtet sind.
- Sobald die Bereitstellung erfolgt ist, überwachen Sie die SLA und erstatten Sie darüber Bericht.

Ressourcen

Zugehörige bewährte Methoden:

- REL03-BP01 Wählen Sie, wie Sie Ihre Arbeitslast segmentieren möchten
- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler
- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering
- REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten
- Grundlegendes zum Workload-Status

Zugehörige Dokumente:

- Verfügbarkeit mit Redundanz
- Säule der Zuverlässigkeit – Verfügbarkeit
- Messung der Verfügbarkeit
- Grenzen für die AWS-Fehlerisolierung
- Modell der geteilten Verantwortung für Ausfallsicherheit
- Statische Stabilität mithilfe von Availability Zones
- AWS Service Level Agreements (SLAs)
- Empfehlungen zur zellenbasierten Architektur in AWS
- AWS-Infrastruktur
- Whitepaper „Erweiterte Multi-AZ-Resilience-Muster“

Zugehörige Services:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

Testen der Zuverlässigkeit

Nachdem Sie Ihre Workload so konzipiert haben, dass sie den Belastungen der Produktion standhält, sind Tests die einzige Möglichkeit, sie auf die erwartete Funktionalität und Ausfallsicherheit hin zu testen.

Testen Sie, ob Ihre Workload funktionale und nicht funktionale Anforderungen erfüllt, da Fehler oder Leistungsengpässe die Zuverlässigkeit Ihrer Workload beeinträchtigen können. Ermitteln Sie anhand von Tests der Ausfallsicherheit Ihrer Workload latente Bugs, die erst während der Produktion auftreten. Führen Sie diese Tests regelmäßig durch.

Best Practices

- REL12-BP01 Untersuchen von Fehlern mit Playbooks
- REL12-BP02 Durchführen von Analysen nach Vorfällen
- REL12-BP03 Testen von Skalierbarkeits- und Leistungsanforderungen
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering
- REL12-BP05 Regelmäßiges Durchführen von Gamedays

REL12-BP01 Untersuchen von Fehlern mit Playbooks

Gestatten Sie konsistente und schnelle Antworten auf noch unbekannte Fehlerszenarien, indem Sie den Untersuchungsprozess in Playbooks dokumentieren. Playbooks sind vordefinierte Abläufe zum Identifizieren der Faktoren, die zu einem Fehlerszenario beitragen. Die Ergebnisse aus jedem Prozessschritt sind die Grundlage für die nächsten Schritte. Nach diesem Muster wird vorgegangen, bis das Problem identifiziert oder eskaliert wird.

Das Playbook ist eine proaktive Planung, die für effektive Reaktionen erforderlich ist. Wenn nicht vom Playbook abgedeckte Fehlerszenarien in der Produktion auftreten, beheben Sie zunächst das Problem. Analysieren Sie danach die unternommenen Schritte und verwenden Sie diese, um einen neuen Eintrag im Playbook hinzuzufügen.

Beachten Sie, dass Playbooks als Reaktion auf bestimmte Vorfälle verwendet werden, während Runbooks verwendet werden, um bestimmte Ergebnisse zu erzielen. Häufig werden Runbooks für Routineaktivitäten verwendet, Playbooks hingegen, um auf außergewöhnliche Ereignisse zu reagieren.

Typische Anti-Muster:

- Planen der Bereitstellung einer Workload, ohne die Prozesse zur Diagnose von Problemen oder Reaktion auf Vorfälle zu kennen.
- Ungeplante Entscheidungen darüber, in welchen Systemen bei der Untersuchung von Ereignissen Protokolle und Metriken erfasst werden sollen.
- Metriken und Ereignisse werden nicht lange genug aufbewahrt, um die Daten abrufen zu können.

Vorteile der Nutzung dieser bewährten Methode: Durch die Erfassung von Playbooks wird sichergestellt, dass Prozesse konsistent befolgt werden können. Ihre Playbooks werden als Code festgehalten, um die Entstehung von Fehlern durch manuelle Aktivitäten zu reduzieren. Durch die Automatisierung von Playbooks kann schneller auf Ereignisse reagiert werden, weil Teammitglieder nicht eingreifen müssen oder ihnen vor dem Eingreifen zusätzliche Informationen zur Verfügung gestellt werden.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Ermitteln Sie Probleme mithilfe von Playbooks. Playbooks sind dokumentierte Prozesse für die Untersuchung von Problemen. Durch die Dokumentation der Prozesse in Playbooks werden die

Voraussetzungen für eine einheitliche und schnelle Reaktion auf Fehlerszenarien geschaffen. Playbooks müssen die Informationen und Anleitungen enthalten, die eine entsprechend qualifizierte Person zum Zusammentragen sachdienlicher Informationen, zum Identifizieren möglicher Fehlerursachen, zum Isolieren von Fehlern und zum Bestimmen beitragender Faktoren (zum Analysieren nach einem Vorfall) benötigt.

- Implementieren Sie Playbooks als Code. Führen Sie Ihre Operationen als Code aus, indem Sie Skripts für Ihre Playbooks erstellen, um Konsistenz sicherzustellen und Fehler zu reduzieren, die durch manuelle Prozesse verursacht werden. Playbooks können aus mehreren Skripts bestehen, die die verschiedenen Schritte darstellen, die erforderlich sein können, um die zu einem Problem beitragenden Faktoren zu identifizieren. Runbook-Aktivitäten können aufgerufen oder im Rahmen von Playbook-Aktivitäten ausgeführt werden. Sie können auch als Antwort auf identifizierte Ereignisse die Ausführung eines Playbooks auslösen.
  - Automatisieren Ihrer betrieblichen Playbooks mit AWS Systems Manager
  - AWS Systems Manager Run Command
  - AWS Systems Manager Automation
  - Was ist AWS Lambda?
  - Was ist Amazon EventBridge?
  - Verwenden von Amazon CloudWatch Alarms

Ressourcen

Zugehörige Dokumente:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- Automatisieren Ihrer betrieblichen Playbooks mit AWS Systems Manager
- Verwenden von Amazon CloudWatch Alarms
- Verwenden von Canaries (Amazon CloudWatch Synthetics)
- Was ist Amazon EventBridge?
- Was ist AWS Lambda?

Zugehörige Beispiele:

- Automating operations with Playbooks and Runbooks

REL12-BP02 Durchführen von Analysen nach Vorfällen

Überprüfen Sie die Ereignisse mit Auswirkungen auf Kunden und bestimmen Sie die beitragenden Faktoren und Präventivmaßnahmen. Entwickeln Sie anhand dieser Informationen Abhilfemaßnahmen, um Wiederholungen einzuschränken oder zu verhindern. Entwickeln Sie Verfahren für schnelle und effektive Reaktionen. Informieren Sie nach Bedarf auf zielgruppengerechte Weise über beitragende Faktoren und Korrekturmaßnahmen. Legen Sie eine Kommunikationsmethode fest, um andere bei Bedarf über die Ursachen zu informieren.

Bewerten Sie, warum bestehende Tests das Problem nicht gefunden haben. Fügen Sie Tests für diesen Fall hinzu, wenn noch keine Tests vorhanden sind.

Gewünschtes Ergebnis: Ihre Teams haben einen konsistenten und vereinbarten Ansatz für den Umgang mit Analysen nach einem Vorfall. Ein Mechanismus ist der Correction of Error (CoE)-Prozess. Der COE-Prozess hilft Ihren Teams, die Ursachen für Vorfälle zu identifizieren, zu verstehen und zu beseitigen. Gleichzeitig werden Mechanismen und Leitlinien entwickelt, um die Wahrscheinlichkeit zu verringern, dass sich ein solcher Vorfall wiederholt.

Typische Anti-Muster:

- Beitragende Faktoren werden ermittelt, es wird jedoch nicht weiter nach anderen potenziellen Problemen und Lösungsansätzen gesucht.
- Es werden nur menschliche Fehlerursachen ermittelt, es wird aber keine Schulung oder Automatisierung bereitgestellt, die menschliche Fehler verhindern könnte.
- Der Fokus liegt auf Schuldzuweisungen, anstatt die Ursache zu verstehen, wodurch eine Kultur der Angst entsteht und eine offene Kommunikation behindert wird.
- Es wird versäumt, Erkenntnisse weiterzugeben, wodurch die Ergebnisse der Ereignisanalyse in einer kleinen Gruppe bleiben und andere nicht von den gewonnenen Erkenntnissen profitieren können.
- Es gibt keine Mechanismen zur Erfassung des institutionellen Wissens, wodurch wertvolle Erkenntnisse verloren gehen, da die gewonnenen Erkenntnisse nicht in Form von aktualisierten bewährten Methoden festgehalten werden und es zu wiederholten Vorfällen mit derselben oder einer ähnlichen Ursache kommt.

Vorteile der Nutzung dieser bewährten Methode: Durch Analysen von Vorfällen und das Teilen von Ergebnissen können die Risiken für andere Workloads mit den gleichen beitragenden

Faktoren die Risiken verringert werden. Außerdem können Abhilfemaßnahmen oder automatisierte Wiederherstellungen implementiert werden, bevor es zu einem Vorfall kommt.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Durch gute Analysen nach Vorfällen lassen sich allgemeine Lösungen für Probleme mit Architekturmustern ermitteln, die Sie bereits an anderer Stelle in den Systemen anwenden.

Ein Grundpfeiler des COE-Prozesses ist die Dokumentation und Behandlung von Problemen. Es wird empfohlen, ein standardisiertes Verfahren zur Dokumentation kritischer Ursachen festzulegen und sicherzustellen, dass diese überprüft und behoben werden. Weisen Sie die Verantwortung für den Analyseprozess nach einem Vorfall eindeutig zu. Benennen Sie ein verantwortliches Team oder eine Person, die die Untersuchungen von Vorfällen und die Folgemaßnahmen beaufsichtigt.

Fördern Sie eine Kultur, die sich auf Lernen und Verbesserung konzentriert, anstatt Schuldzuweisungen vorzunehmen. Betonen Sie, dass das Ziel darin besteht, zukünftige Vorfälle zu verhindern, und nicht darin, Einzelpersonen zu strafen.

Entwickeln Sie klar definierte Verfahren für die Durchführung von Analysen nach einem Vorfall. Diese Verfahren sollten die zu ergreifenden Schritte, die zu sammelnden Informationen und die Schlüsselfragen, die während der Analyse zu behandeln sind, darlegen. Untersuchen Sie Vorfälle gründlich und gehen Sie dabei über die unmittelbaren Ursachen hinaus, um die Grundursachen und die beitragenden Faktoren zu ermitteln. Verwenden Sie Techniken wie die 5-Why-Methode, um sich eingehend mit den zugrundeliegenden Problemen zu befassen.

Führen Sie eine Sammlung von Erkenntnissen, die Sie aus der Analyse von Vorfällen gewonnen haben. Dieses institutionelle Wissen kann als Referenz für zukünftige Vorfälle und Präventionsmaßnahmen dienen. Tauschen Sie die Ergebnisse und Erkenntnisse aus den Analysen nach dem Vorfall aus und erwägen Sie, offene Besprechungen nach dem Vorfall abzuhalten, um die gewonnenen Erkenntnisse zu diskutieren.

Implementierungsschritte

- Achten Sie bei der Analyse nach einem Vorfall darauf, dass der Prozess frei von Schuldzuweisungen ist. Dies ermöglicht es den an dem Vorfall beteiligten Personen, die vorgeschlagenen Korrekturmaßnahmen sachlich zu beurteilen und fördert eine ehrliche Selbsteinschätzung und die Zusammenarbeit zwischen den Teams.

- Definieren Sie eine standardisierte Methode zur Dokumentation kritischer Probleme. Ein solches Dokument könnte beispielsweise folgendermaßen strukturiert sein:
  - Was ist passiert?
  - Welche Auswirkungen gab es auf Kunden und Ihr Unternehmen?
  - Was war die Ursache?
  - Welche Daten haben Sie, um dies zu unterstützten?
    - Zum Beispiel Metriken und Grafiken
  - Welches waren die kritischen Auswirkungen auf die Säulen, insbesondere in puncto Sicherheit?
    - Beim Entwerfen von Workloads sollten Sie je nach Geschäftskontext zwischen den einzelnen Säulen abwägen. Diese Geschäftsentscheidungen können Ihre technischen Prioritäten beeinflussen. Sie können optimieren, um Kosten zulasten der Zuverlässigkeit in Entwicklungsumgebungen zu senken, oder Sie können bei unternehmenskritischen Lösungen die Zuverlässigkeit mit höheren Kosten optimieren. Sicherheit ist immer oberstes Gebot, da Sie Ihre Kunden schützen müssen.
  - Welche Erkenntnisse haben Sie gewonnen?
  - Welche Maßnahmen ergreifen Sie?
    - Aktionselemente
    - Verwandte Elemente
- Erstellen Sie klar definierte Standardverfahren für die Durchführung von Analysen nach einem Vorfall.
- Richten Sie ein standardisiertes Verfahren zur Meldung von Vorfällen ein. Dokumentieren Sie alle Vorfälle ausführlich, einschließlich des ersten Vorfallberichts, der Protokolle, der Kommunikation und der während des Vorfalls getroffenen Maßnahmen.
- Denken Sie daran, dass ein Vorfall nicht unbedingt einen Ausfall zur Folge haben muss. Es könnte sich um einen Beinahe-Unfall handeln oder um ein System, das auf unerwartete Weise funktioniert und dennoch seine Geschäftsfunktion erfüllt.
- Verbessern Sie Ihren Analyseprozess nach einem Vorfall kontinuierlich auf Grundlage von Rückmeldungen und gewonnenen Erkenntnissen.
- Halten Sie die wichtigsten Erkenntnisse in einem Wissensmanagementsystem fest und überlegen Sie, welche Muster in Entwicklerhandbücher oder Checklisten vor der Bereitstellung aufgenommen werden sollten.

Ressourcen

Zugehörige Dokumente:

- Warum sollten Sie eine Fehlerkorrektur (Correction of Error,CoE) entwickeln

Zugehörige Videos:

- Der Ansatz von Amazon: erfolgreiches Scheitern
- AWS re:Invent 2021 – Die Amazon Builders' Library: Betriebliche Exzellenz von Amazon

REL12-BP03 Testen von Skalierbarkeits- und Leistungsanforderungen

Nutzen Sie Techniken wie Lasttests, um zu prüfen, ob die Workload die Skalierungs- und Leistungsanforderungen erfüllt.

In der Cloud können Sie bei Bedarf eine Testumgebung in Produktionsgröße für Ihren Workload erstellen. Anstatt sich auf eine herunterskalierte Testumgebung zu verlassen und dadurch ungenaue Vorhersagen des Produktionsverhaltens zu riskieren, können Sie in der Cloud eine Testumgebung bereitzustellen, die Ihrer erwarteten Produktionsumgebung sehr nahe kommt. Diese Umgebung ermöglicht Ihnen Tests in einer genaueren Simulation der realen Bedingungen, denen Ihre Anwendung ausgesetzt ist.

Neben der Durchführung von Leistungstests ist es wichtig, sicherzustellen, dass Ihre Basisressourcen, Skalierungseinstellungen, Service Quotas und die Ausfallsicherheit unter Auslastung wie erwartet funktionieren. Dieser ganzheitliche Ansatz stellt sicher, dass Ihre Anwendung auch unter den anspruchsvollsten Bedingungen zuverlässig skaliert und nach Bedarf ausgeführt werden kann.

Gewünschtes Ergebnis: Ihr Workload behält sein erwartetes Verhalten auch bei Spitzenlast bei. Sie kümmern sich proaktiv um leistungsbezogene Probleme, die auftreten können, wenn die Anwendung wächst und sich weiterentwickelt.

Typische Anti-Muster:

- Sie verwenden Testumgebungen, die nicht genau mit der Produktionsumgebung übereinstimmen.
- Sie behandeln Lasttests als separate, einmalige Aktivität und nicht als integrierten Bestandteil der Continuous Integration (CI)-Pipeline für die Bereitstellung.

- Sie definieren keine klaren und messbaren Leistungsanforderungen wie beispielsweise Ziele in Bezug auf Reaktionszeit, Durchsatz und Skalierbarkeit.
- Sie nutzen Szenarien mit unrealistischer oder unzureichender Last für Ihre Tests und versäumen es, Spitzenlasten, plötzliche Spitzen und anhaltend hohe Lasten zu testen.
- Sie unterziehen den Workload keinem Belastungstest, indem Sie die erwarteten Lastgrenzen überschreiten.
- Sie verwenden unzureichende oder unangemessene Tools für Lasttests und die Erstellung von Leistungsprofilen.
- Sie verfügen nicht über umfassende Überwachungs- und Warnsysteme, um Leistungsmetriken zu verfolgen und Anomalien zu erkennen.

Vorteile der Nutzung dieser bewährten Methode:

- Mithilfe von Lasttests können Sie potenzielle Leistungsengpässe in Ihrem System vor dem Übergang in die Produktion erkennen. Wenn Sie den Datenverkehr und die Workloads auf Produktionsebene simulieren, können Sie Bereiche identifizieren, in denen Ihr System möglicherweise Schwierigkeiten hat, die Last zu bewältigen, beispielsweise langsame Reaktionszeiten, Ressourcenengpässe oder Systemausfälle.
- Wenn Sie Ihr System unter verschiedenen Lastbedingungen testen, können Sie die Ressourcenanforderungen zur Unterstützung Ihres Workloads besser verstehen. Mithilfe dieser Informationen können Sie fundierte Entscheidungen über die Ressourcenzuweisung treffen und eine übermäßige oder zu geringe Bereitstellung von Ressourcen verhindern.
- Um potenzielle Fehlerquellen zu identifizieren, können Sie beobachten, wie sich Ihr Workload in Situationen mit hoher Last verhält. Diese Informationen helfen Ihnen, die Zuverlässigkeit und Ausfallsicherheit Ihres Workloads zu verbessern, indem Sie nach Bedarf Fehlertoleranzmechanismen, Failover-Strategien und Redundanzmaßnahmen implementieren.
- Sie erkennen und beheben Leistungsprobleme frühzeitig und können so die kostspieligen Folgen von Systemausfällen, langsamen Reaktionszeiten und unzufriedenen Benutzern vermeiden.
- Während der Tests erfasste detaillierte Leistungsdaten und Profilinformationen können Ihnen helfen, leistungsbezogene Probleme zu beheben, die in der Produktion auftreten können. Dies kann eine schnellere Reaktion auf Vorfälle und eine zügigere Behebung von Problemen ermöglichen, wodurch die Auswirkungen auf die Benutzer und den Betrieb Ihres Unternehmens verringert werden.

- In bestimmten Branchen können proaktive Leistungstests dazu beitragen, dass Ihr Workload die Compliance-Standards erfüllt, wodurch das Risiko von Strafen oder rechtlichen Problemen verringert wird.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Der erste Schritt besteht darin, eine umfassende Teststrategie zu definieren, die alle Aspekte der Skalierungs- und Leistungsanforderungen abdeckt. Definieren Sie zunächst klar die Servicelevel-Ziele (SLOs, Service-Level Objectives) Ihres Workloads auf der Grundlage Ihrer Geschäftsanforderungen, also beispielsweise Durchsatz, Latenzhistogramm und Fehlerrate. Entwerfen Sie dann eine Reihe von Tests, mit denen verschiedene Lastszenarien simuliert werden können, von durchschnittlicher Auslastung bis hin zu plötzlichen Spitzen und anhaltenden Spitzenlasten. Stellen Sie sicher, dass das Verhalten des Workloads Ihren SLOs entspricht. Diese Tests sollten automatisiert und in Ihre Pipeline für kontinuierliche Integration und Bereitstellung integriert werden, um Leistungsrückgänge frühzeitig im Entwicklungsprozess zu erkennen.

Um die Skalierung und Leistung effektiv testen zu können, sollten Sie in die richtigen Tools und eine geeignete Infrastruktur investieren. Hierzu gehören Tools für Lasttests, mit denen ein realistischer Benutzerverkehr generiert werden kann, Tools für die Erstellung von Leistungsprofilen zur Ermittlung von Engpässen und Überwachungslösungen zur Verfolgung wichtiger Metriken. Wichtig ist es sicherzustellen, dass Ihre Testumgebungen in Bezug auf Infrastruktur und Umgebungsbedingungen möglichst genau mit der Produktionsumgebung übereinstimmen, um möglichst genaue Testergebnisse zu erzielen. Verwenden Sie Infrastructure-as-Code- und containerbasierte Anwendungen, um die zuverlässige Replikation und Skalierung produktionsähnlicher Setups zu vereinfachen.

Skalierungs- und Leistungstests sind ein fortlaufender Prozess, keine einmalige Aktivität. Implementieren Sie eine umfassende Überwachung und Warnmeldungen, um die Leistung der Anwendung in der Produktion zu verfolgen, und verwenden Sie diese Daten, um Ihre Teststrategien und Optimierungsbemühungen kontinuierlich zu verbessern. Analysieren Sie Leistungsdaten regelmäßig, um neu auftretende Probleme zu identifizieren, neue Skalierungsstrategien zu testen, Optimierungen vorzunehmen und so die Effizienz und Zuverlässigkeit der Anwendung zu verbessern. Wenn Sie einen iterativen Ansatz verfolgen und kontinuierlich aus Produktionsdaten lernen, können Sie überprüfen, ob sich Ihre Anwendung an variable Benutzeranforderungen anpassen und im Laufe der Zeit Ausfallsicherheit und eine optimale Leistung aufrechterhalten kann.

Implementierungsschritte

1. Legen Sie klare und messbare Leistungsanforderungen fest, beispielsweise Ziele in Bezug auf Reaktionszeit, Durchsatz und Skalierbarkeit. Diese Anforderungen sollten auf den Nutzungsmustern Ihres Workloads, den Erwartungen der Benutzer und Ihren Geschäftsanforderungen basieren.
2. Wählen und konfigurieren Sie ein Tool für Lasttests, das die Lastmuster und das Benutzerverhalten in Ihrer Produktionsumgebung genau nachahmen kann.
3. Richten Sie eine Testumgebung ein, die der Produktionsumgebung, einschließlich Infrastruktur und Umgebungsbedingungen, sehr nahe kommt, um die Genauigkeit Ihrer Testergebnisse zu verbessern.
4. Erstellen Sie eine Testsuite, die eine Vielzahl von Szenarien abdeckt, von durchschnittlichen Nutzungsmustern über Spitzenlasten und kurzfristige Spitzen bis hin zu anhaltend hohen Lasten. Integrieren Sie die Tests in Ihre Pipelines für die kontinuierliche Integration und Bereitstellung, um Leistungsrückgänge schon früh im Entwicklungsprozess zu erkennen.
5. Führen Sie Lasttests durch, um den tatsächlichen Benutzerverkehr zu simulieren und zu verstehen, wie sich Ihre Anwendung unter verschiedenen Lastbedingungen verhält. Überschreiten Sie die erwartete Last, um Ihre Anwendung einem Stresstest zu unterziehen, und beobachten Sie das Verhalten der Anwendung. Prüfen Sie beispielsweise, ob es zu einer Verschlechterung der Reaktionszeit, zu einer Ressourcenüberlastung oder zu Systemausfällen kommt. Auf diese Weise können Sie die Belastungsgrenze Ihrer Anwendung ermitteln und fundierte Skalierungsstrategien entwickeln. Bewerten Sie die Skalierbarkeit Ihres Workloads, indem Sie die Last schrittweise erhöhen, und messen Sie die Auswirkungen auf die Leistung, um Skalierungsgrenzen zu ermitteln und zukünftige Kapazitätsanforderungen zu planen.
6. Implementieren Sie eine umfassende Überwachung und Warnmeldungen, um Leistungsmetriken zu verfolgen, Anomalien zu erkennen und Skalierungsaktionen oder Benachrichtigungen einzuleiten, wenn Schwellenwerte überschritten werden.
7. Überwachen und analysieren Sie die Leistungsdaten kontinuierlich, um Bereiche mit Verbesserungspotenzial zu identifizieren. Verbessern Sie Ihre Teststrategien und Optimierungsbemühungen durch Wiederholung.

Ressourcen

Zugehörige bewährte Methoden:

- REL01-BP04 Überwachen und Verwalten von Kontingenten

- REL06-BP01 Überwachen aller Komponenten des Workloads (Generierung)
- REL06-BP03 Senden von Benachrichtigungen (Verarbeitung und Benachrichtigung in Echtzeit)

Zugehörige Dokumente:

- Anwendungen für Lasttests
- Distributed Load Testing auf AWS
- Application Performance Monitoring
- Amazon-EC2-Testrichtlinie

Zugehörige Beispiele:

- Verteilte Lasttests in AWS (GitHub)

Zugehörige Tools:

- Amazon CodeGuru Profiler
- Amazon CloudWatch RUM
- Apache JMeter
- K6
- Vegeta
- Hey
- ab
- wrk
- Distributed Load Testing auf AWS

REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering

Führen Sie regelmäßig Chaos-Experimente in oder nahe an Produktionsumgebungen aus, um zu verstehen, wie Ihr System auf ungünstige Bedingungen reagiert.

Gewünschtes Ergebnis:

Die Ausfallsicherheit der Workload wird regelmäßig durch die Anwendung von Chaos-Engineering in Form von Fehlerinjektionsexperimenten oder einer Injektion unerwarteter Last überprüft. Dazu

kommen Tests der Ausfallsicherheit, um das bekannte erwartete Verhalten der Workload während eines Ereignisses zu validieren. Kombinieren Sie Chaos-Engineering mit Tests der Ausfallsicherheit, um sicher zu sein, dass Ihre Workload Komponentenausfällen standhalten und sich von unerwarteten Unterbrechungen erholen kann – mit minimalen oder gar keinen Auswirkungen.

Typische Anti-Muster:

- Auslegung der Systeme auf Ausfallsicherheit, aber keine Überprüfung, wie die Workload als Ganzes funktioniert, wenn Fehler auftreten.
- Keine Experimente unter echten Bedingungen und der erwarteten Last.
- Keine Behandlung der Experimente als Code und fehlendes Aufrechterhalten während des Entwicklungszyklus.
- Keine Durchführung von Chaos-Experimenten als Teil Ihrer CI/CD-Pipeline und außerhalb von Bereitstellungen.
- Keine Nutzung früherer Analysen nach Vorfällen bei der Entscheidung über die Fehler, mit denen experimentiert werden soll.

Vorteile der Nutzung dieser bewährten Methode: Durch die Injektion von Fehlern zur Überprüfung der Resilienz Ihrer Workload gewinnen Sie die nötige Zuversicht, dass die Wiederherstellungsverfahren Ihres resilienten Entwurfs im Fall eines realen Fehlers funktionieren.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Das Chaos-Engineering bietet Ihren Teams die nötigen Chancen, um auf kontrollierte Weise kontinuierlich reale Störungen (Simulationen) auf Serviceanbieter-, Infrastruktur-, Workload- und Komponentenebene zu injizieren – mit nur minimalen oder gar keinen Auswirkungen auf Ihre Kunden. Ihre Teams können so aus Fehlern lernen und die Resilienz Ihrer Workloads beobachten, messen und verbessern. Darüber hinaus können sie überprüfen, ob Warnungen ausgelöst werden und die Teams über Ereignisse benachrichtigt werden.

Bei kontinuierlicher Ausführung kann das Chaos-Engineering Mängel in Ihren Workloads aufzeigen, die sich negativ auf Verfügbarkeit und Ausführung auswirken könnten, wenn sie nicht behoben werden.

> **Note**
>
> Beim Chaos-Engineering geht es um das Experimentieren mit einem System, um sich davon zu überzeugen, dass das System in der Produktion auch außergewöhnlichen Bedingungen standhalten kann. – Grundlagen des Chaos-Engineering

Wenn ein System diesen Disruptionen standhalten kann, sollte das Chaos-Experiment weiter als automatisierter Regressionstest ausgeführt werden. In dieser Form sollten Chaos-Experimente als Teil Ihres Systementwicklungszyklus (Systems Development Lifecycle, SDLC) und Ihrer CI/CD-Pipeline ausgeführt werden.

Um sicherzustellen, dass Ihre Workload resilient gegenüber dem Ausfall von Komponenten ist, sollten Sie im Rahmen Ihrer Experimente Ereignisse aus der Praxis injizieren. Sie könnten beispielsweise mit dem Verlust von Amazon EC2-Instances oder einem Failover der primären Amazon RDS-Datenbank-Instance experimentieren und so verifizieren, dass Ihre Workload nicht beeinträchtigt wird (oder nur minimal beeinträchtigt wird). Mit einer Kombination von Komponentenfehlern könnten Sie Ereignisse simulieren, die von einer Disruption in einer Availability Zone verursacht werden könnten.

Hinsichtlich Fehlern auf Anwendungsebene (z. B. Abstürzen) könnten Sie mit Stressfaktoren wie Speicher- und CPU-Auslastung beginnen.

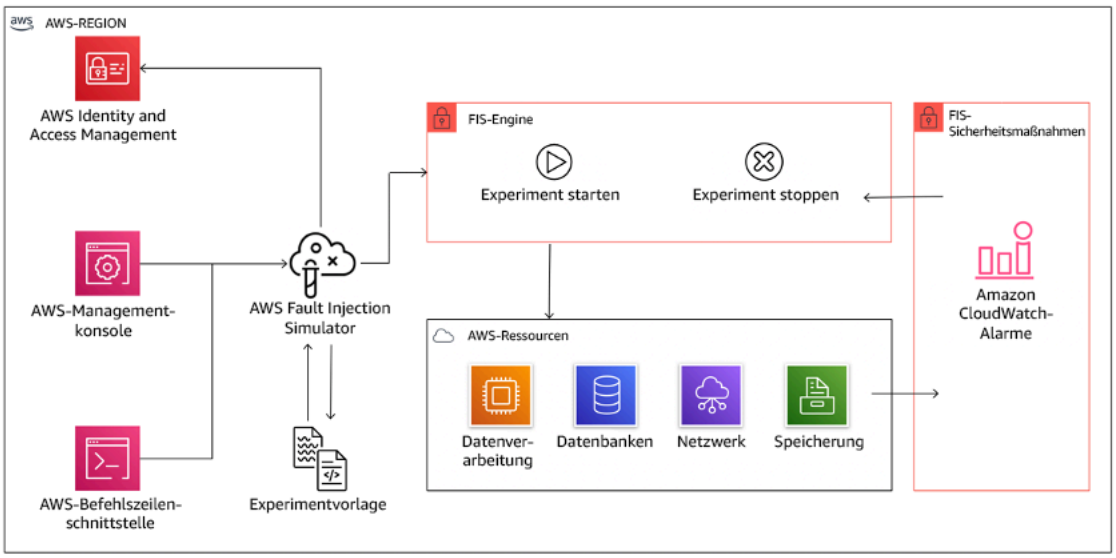
Zur Validierung von Fallback- oder Failover-Mechanismen für externe Abhängigkeiten, die bei zeitweisen Netzwerkdisruptionen ausgelöst werden, sollten Ihre Komponenten diese Ereignisse durch das Blockieren des Zugriffs auf externe Anbieter über einen bestimmten Zeitraum simulieren, der von wenigen Sekunden bis zu mehreren Stunden dauern kann.

Andere Degradierungsmodi führen möglicherweise zu einer reduzierten Funktionalität und zu verzögerten Reaktionen, was eine Disruption Ihrer Services verursachen kann. Bekannte Quellen für diese Degradierung sind eine erhöhte Latenz bei kritischen Services und eine unzuverlässige Netzwerkkommunikation (Verlust von Paketen). Experimente mit diesen Fehlern, darunter Netzwerkeffekten wie Latenz, Nachrichtenverlust und DNS-Ausfällen, könnten die fehlende Fähigkeit zur Auflösung eines Namens, zum Erreichen des DNS-Service oder zur Herstellung von Verbindungen zu abhängigen Services umfassen.

Chaos-Engineering-Tools:

AWS Fault Injection Service (AWS FIS) ist ein vollständig verwalteter Service für die Injektion von Fehlern, den Sie innerhalb oder außerhalb Ihrer CD-Pipeline verwenden können, um mit

diesen Fehlern zu experimentieren. AWS FIS ist eine gute Wahl für Gamedays, die dem Chaos-Engineering gewidmet sind. Es unterstützt die gleichzeitige Einführung von Fehlern für verschiedenen Ressourcentypen, darunter Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS) und Amazon RDS. Zu diesen Fehlern gehören die Beendigung von Ressourcen, die Erzwingung von Failovern, die Auslastung von CPU oder Arbeitsspeicher, Drosselung, Latenz und Paketverluste. Da dieser Service in Amazon CloudWatch Alarms integriert ist, können Sie Stoppbedingungen als Integritätsschutz einrichten, um Experimente rückgängig zu machen, wenn sie unerwartete Auswirkungen haben.



AWS Fault Injection Service lässt sich in AWS-Ressourcen integrieren, um Ihnen die Ausführung von Fehlerinjektionsexperimenten für Ihre Workloads zu ermöglichen.

Es gibt auch verschiedene Drittanbieteroptionen für Fehlerinjektionsexperimente. Dazu gehören Open-Source-Tools wie Chaos Toolkit, Chaos Mesh und Litmus Chaos sowie kommerzielle Tools wie Gremlin. Um den Umfang der Fehler zu erweitern, die in AWS injiziert werden können, ist AWS FIS in Chaos Mesh und Litmus Chaos integrierbar, sodass Sie die Workflows zur Fehlerinjektion zwischen mehreren Tools koordinieren können. Sie können beispielsweise einen Stresstest für die CPU eines Pods mit Chaos-Mesh- oder Litmus-Fehlern ausführen und gleichzeitig einen zufällig ausgewählten Prozentsatz von Cluster-Knoten mit AWS FIS-Fehleraktionen beenden.

Implementierungsschritte

1. Ermitteln Sie die Fehler, mit denen experimentiert werden soll.

   Bewerten Sie das Design Ihrer Workload in Bezug auf die Resilienz. Solche Designs (die unter Verwendung der bewährten Methoden des Well-Architected Framework erstellt wurden)

berücksichtigen Risiken, die auf kritischen Abhängigkeiten, vergangenen Ereignissen, bekannten Problemen und Compliance-Anforderungen basieren. Listen Sie die einzelnen Elemente des Designs auf, die Resilienz zeigen sollen, und die Fehler, denen es standhalten soll. Weitere Informationen zur Erstellung solcher Listen finden Sie im Whitepaper zur Überprüfung der betrieblichen Bereitschaft, in dem Sie auch erfahren, wie Sie einen Prozess erstellen können, um das Wiederauftreten früherer Vorfälle zu verhindern. Der Prozess für die Analyse von Fehlerarten und ihren Auswirkungen (Failure Modes and Effects Analysis, FMEA) stellt Ihnen ein Framework für Fehleranalysen auf Komponentenebene und die Analyse der Auswirkungen dieser Fehler auf Ihre Workload bereit. FMEA wird von Adrian Cockcroft unter Failure Modes and Continuous Resilience näher beschrieben.

2. Weisen Sie jedem Fehler eine Priorität zu.

   Beginnen Sie mit einer groben Kategorisierung wie „Hoch“, „Mittel“ oder „Niedrig“. Berücksichtigen Sie bei der Festlegung der Priorität die Häufigkeit des Fehlers und die Auswirkungen des Fehlers auf die Workload insgesamt.

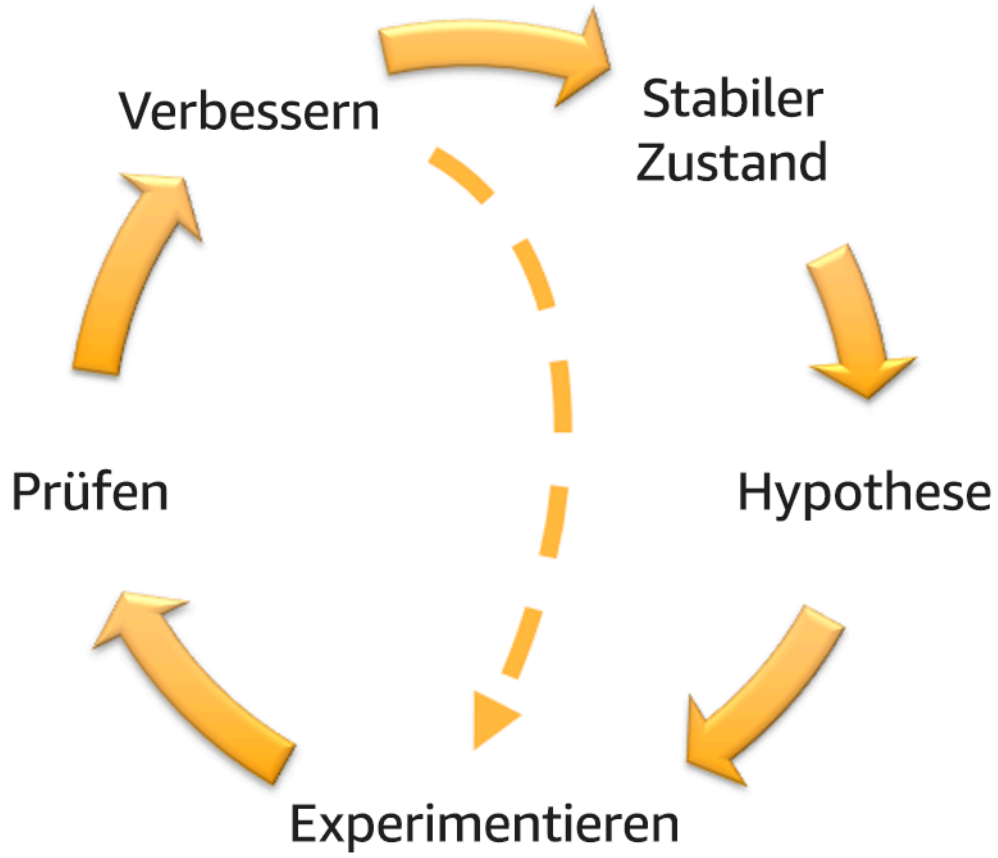
   Analysieren Sie hinsichtlich der Häufigkeit eines bestimmten Fehlers frühere Daten für die betreffende Workload, wenn verfügbar. Wenn keine Daten verfügbar sind, verwenden Sie Daten zu anderen Workloads, die in einer ähnlichen Umgebung ausgeführt werden.

   Bei der Betrachtung der Auswirkungen eines bestimmten Fehlers gilt, dass die Auswirkungen im Allgemeinen umso größer sind, je größer der vom Fehler betroffene Bereich ist. Sie sollten auch das Design und den Zweck der Workload berücksichtigen. Beispielsweise ist für eine Workload, die Daten transformiert und analysiert, der Zugriff auf die Quelldatenspeicher von kritischer Bedeutung. In diesem Fall würden Sie Experimente im Zusammenhang mit Zugriffsfehlern, Zugriffsdrosselungen und Latenzen priorisieren.

   Nach Vorfällen durchgeführte Analysen stellen eine gute Datenquelle dar, um Häufigkeit und Auswirkungen von Fehlerarten besser zu verstehen.

   Legen Sie anhand der zugewiesenen Priorität die Fehler fest, mit denen zuerst experimentiert werden soll, und die Reihenfolge, in der neue Fehlerinjektionsexperimente entwickelt werden sollen.

3. Für jedes von Ihnen ausgeführte Experiment sollten Sie sich am Schwungrad für Chaos-Engineering und kontinuierliche Resilienz in der folgenden Abbildung orientieren.



Schwungrad für Chaos-Engineering und kontinuierliche Resilienz unter Verwendung der wissenschaftlichen Methode von Adrian Hornsby.

a. Definieren Sie den Steady-State als die messbare Ausgabe einer Workload, die ein normales Verhalten zeigt.

   Ihre Workload befindet sich im Steady-State, wenn sie zuverlässig und wie erwartet ausgeführt wird. Daher sollten Sie die Integrität Ihrer Workload überprüfen, bevor Sie den Steady-State definieren. Steady-State bedeutet nicht notwendigerweise, dass sich ein Fehler nicht auf die Workload auswirkt, da ein bestimmter Prozentsatz an Fehlern innerhalb akzeptabler Grenzen liegen könnte. Der Steady-State ist die Basislinie, die Sie während des Experiments beobachten. Diese wird Anomalien aufweisen, wenn Ihre Hypothese, die Sie im nächsten Schritt definieren, nicht die erwarteten Ergebnisse zeigt.

Der Steady-State eines Zahlungssystems kann beispielsweise als die Verarbeitung von 300 TPS mit einer Erfolgsrate von 99 % und einer Roundtrip-Zeit von 500 ms definiert sein.

b. Formulieren Sie eine Hypothese dazu, wie die Workload auf den Fehler reagieren wird.

Eine gute Hypothese basiert darauf, wie die Workload den Fehler voraussichtlich bewältigt, um den Steady-State zu wahren. Die Hypothese besagt, dass bei einem Fehler eines spezifischen Typs das System oder die Workload weiter im Steady-State bleibt, da die Workload mit bestimmten Resilienzmerkmalen entworfen wurde. Der spezifische Fehlertyp und die Fehlerbewältigung sollten in der Hypothese angegeben werden.

Sie können für die Hypothese die folgende Vorlage verwenden (andere Formulierungen sind jedoch auch akzeptabel):

> **Note**
>
> Wenn ein *`bestimmter Fehler`* auftritt, beschreibt die *`Workload-Name`*-Workload *`Maßnahmen zur Minderung der Auswirkungen`*, um die *`Auswirkungen auf geschäftliche oder technische Kennzahlen`* beizubehalten.

Beispiel:

- Wenn 20 % der Knoten in der Amazon EKS-Knotengruppe ausfallen, wird die Transaction Create API das 99. Perzentil der Anforderungen weiter in weniger als 100 ms erfüllen (Steady-State). Die Amazon EKS-Knoten werden innerhalb von fünf Minuten wiederhergestellt und die Pods werden geplant und verarbeiten Datenverkehr innerhalb von acht Minuten nach der Einleitung des Experiments. Warnungen werden innerhalb von drei Minuten ausgelöst.
- Wenn eine einzelne Amazon EC2-Instance ausfällt, veranlasst die Elastic Load Balancing-Zustandsprüfung des Bestellsystems Elastic Load Balancing, Anforderungen ausschließlich an die noch intakten Instances zu senden, während Amazon EC2 Auto Scaling die ausgefallene Instance ersetzt. Dabei kommt es zu einer Steigerung der serverseitigen Fehler (5xx) um weniger als 0,01 % (Steady-State).
- Wenn die primäre Amazon RDS-Datenbank-Instance ausfällt, führt die Workload für die Erfassung von Lieferkettendaten einen Failover aus und stellt eine Verbindung zur Amazon RDS-Standby-Datenbank-Instance her, sodass es für weniger als 1 Minute zu Lese- oder Schreibfehlern für die Datenbank kommt (Steady-State).

c. Führen Sie das Experiment aus, indem Sie den Fehler injizieren.

   Ein Experiment sollte grundsätzlich nicht zu einem Ausfall führen und von der Workload toleriert werden. Wenn Sie wissen, dass die Workload ausfallen wird, sollten Sie das Experiment nicht durchführen. Das Chaos-Engineering sollte verwendet werden, um bekannt-unbekannte oder unbekannt-unbekannte Ereignisse zu untersuchen. Bekannt-unbekannte Ereignisse sind Dinge, derer Sie sich bewusst sind, die Sie aber nicht vollständig verstehen, und unbekannt-unbekannte Ereignisse sind Dinge, derer Sie sich nicht bewusst sind und die Sie auch nicht vollständig verstehen. Wenn Sie Experimente für eine Workload ausführen, von der Sie wissen, dass sie fehlerhaft ist, werden Sie keine neuen Erkenntnisse gewinnen. Ihr Experiment sollte sorgfältig geplant sein, einen klaren Wirkungsumfang besitzen und einen Rollback-Mechanismus besitzen, der bei unerwarteten Störungen angewendet werden kann. Wenn eine sorgfältige Überprüfung zeigt, dass Ihre Workload das Experiment überstehen sollte, können Sie das Experiment starten. Für die Injektion von Fehlern gibt es verschiedene Optionen. Für Workloads in AWS bietet AWS FIS viele vordefinierte Fehlersimulationen, die als Aktionen bezeichnet werden. Sie können auch angepasste Aktionen für AWS FIS definieren, die mithilfe von AWS Systems Manager-Dokumenten ausgeführt werden.

   Wir raten davon ab, angepasste Skripts für Chaos-Experimente zu verwenden, es sei denn, die Skripts können den aktuellen Zustand der Workload erkennen, können Protokolle ausgeben und stellen Rollback-Mechanismen und Stoppbedingungen bereit, soweit möglich.

   Ein effektives Framework oder Toolset, das Chaos-Engineering unterstützt, sollte den aktuellen Status des Experiments nachverfolgen, Protokolle ausgeben und Rollback-Mechanismen bereitstellen, um eine kontrollierte Durchführung des Experiments zu unterstützen. Beginnen Sie mit einem verbreitet verwendeten Service wie AWS FIS, der Ihnen die Ausführung von Experimenten mit einem klar definierten Umfang ermöglicht und Sicherheitsmechanismen bereitstellt, um ein Experiment rückgängig machen zu können, wenn es zu unerwarteten Störungen führt. Weitere Informationen zu einer größeren Vielfalt von Experimenten mit AWS FIS finden Sie auch im Lab Resiliente und Well-Architected-Apps mit Chaos-Engineering. Außerdem analysiert AWS Resilience Hub Ihre Workload und erstellt Experimente, die Sie in AWS FIS implementieren und ausführen können.

> **Note**
>
> Sie sollten den Umfang und die Auswirkungen jedes Experiments genau verstehen. Wir empfehlen, Fehler zunächst in einer Nichtproduktionsumgebung zu simulieren, bevor sie in der Produktion ausgeführt werden.

Experimente sollten in der Produktion unter realer Last ausgeführt werden, wobei Canary-Bereitstellungen verwendet werden sollten, die sowohl eine Steuerung als auch eine experimentelle Systembereitstellung ermöglichen, sofern dies möglich ist. Die Ausführung von Experimenten außerhalb von Spitzenzeiten stellt ein empfehlenswertes Verfahren dar, um potenzielle Auswirkungen zu reduzieren, wenn ein Experiment zum ersten Mal in der Produktion durchgeführt wird. Wenn die Verwendung von tatsächlichem Kunden-Traffic ein zu großes Risiko darstellt, können Sie unter Verwendung der Kontroll- und Experimentbereitstellungen Experimente mit synthetischem Datenverkehr in der Produktionsinfrastruktur durchführen. Wenn ein Experiment nicht in der Produktion ausgeführt werden kann, führen Sie es in einer Präproduktionsumgebung aus, die der Produktionsumgebung so nahe wie möglich ist.

Sie müssen einen Integritätsschutz einrichten und überwachen, um sicherzustellen, dass sich das Experiment nicht jenseits akzeptabler Grenzen auf den Datenverkehr in der Produktionsumgebung oder andere Systeme auswirkt. Richten Sie Stoppbedingungen ein, um ein Experiment anhalten zu können, wenn es in einer Integritätsschutz-Metrik einen von Ihnen definierten Schwellenwert erreicht. Diese Metriken sollten die Metrik für den Steady-State der Workload und die Metrik für die Komponenten einschließen, in die Sie den Fehler injizieren. Die synthetische Überwachung (auch als Benutzer-Canary bezeichnet) gehört zu den Metriken, die Sie in der Regel als Benutzer-Proxy einschließen sollten. Stoppbedingungen für AWS FIS werden als Teil der Experimentvorlage unterstützt. Es sind bis zu fünf Stoppbedingungen pro Vorlage möglich.

Zu den Grundsätzen des Chaos-Engineering gehört die Minimierung von Umfang und Auswirkungen des Experiments:

Auch wenn einige kurzfristige negative Auswirkungen zulässig sein sollten, ist der Chaos-Engineer dafür verantwortlich, die Auswirkungen der Experimente zu minimieren und einzudämmen.

Eine Methode für die Überprüfung des Umfangs und der möglichen Auswirkungen besteht darin, das Experiment statt in der Produktionsumgebung zunächst in einer Nichtproduktionsumgebung durchzuführen. Dabei wird überprüft, ob die Schwellenwerte für Stoppbedingungen während des Experiments wie vorgesehen aktiviert werden und ob das Experiment beobachtet werden kann, um Ausnahmen abzufangen.

Wenn Sie Fehlerinjektionsexperimente durchführen, müssen alle verantwortlichen Beteiligten gut informiert sein. Teilen Sie den betroffenen Teams mit, wann die Experimente durchgeführt werden und was zu erwarten ist. Dies können Operations-Teams, die für die Servicezuverlässigkeit verantwortlichen Teams und der Kundensupport sein. Stellen Sie diesen Teams Kommunikationstools bereit, damit sie das Team, das das Experiment durchführt, über nachteilige Auswirkungen informieren können.

Sie müssen nach dem Experiment die Workload und die zugrunde liegenden Systeme wieder in den ursprünglichen, gut funktionierenden Zustand zurückversetzen. Häufig führt das resiliente Design der betreffenden Workload eine Selbstreparatur durch. Einige Fehlerdesigns oder fehlgeschlagenen Experimente können Ihre Workload jedoch in einem nicht erwarteten Fehlerzustand zurücklassen. Nach dem Ende des Experiments müssen Sie dies erkennen und die Workload und die Systeme wiederherstellen können. Mit AWS FIS können Sie eine Rollback-Konfiguration innerhalb der Aktionsparameter einrichten (auch als „Post-Aktion“ bezeichnet). Eine Post-Aktion führt das Ziel in den Zustand zurück, in dem es sich vor Ausführung der Aktion befunden hat. Ob automatisiert (bei Verwendung von AWS FIS) oder manuell – diese Post-Aktionen sollten Teil eines Playbooks sein, das die Erkennung und Behandlung von Fehlern und Ausfällen beschreibt.

d. Prüfen Sie die Hypothese.

Unter Grundlagen des Chaos-Engineering wird die folgende Anleitung für die Verifizierung des Steady-State Ihrer Workload bereitgestellt:

Konzentrieren Sie sich auf die messbare Ausgabe des Systems und nicht auf seine internen Attribute. Messungen dieser Ausgabe über einen kurzen Zeitraum stellen einen Proxy für den Steady-State des Systems dar. Der Gesamtdurchsatz, die Fehlerraten und die Latenz-Perzentile des Systems könnten Metriken sein, die das Steady-State-Verhalten beschreiben. Durch die Konzentration auf die Verhaltensmuster des Systems während Experimenten überprüft das Chaos-Engineering, ob das System funktioniert, statt zu versuchen, die Art der Funktion zu validieren.

In unseren beiden Beispielen oben verwenden wir die Steady-State-Metrik einer Erhöhung von weniger als 0,01 % bei serverseitigen Fehlern (5xx) und von weniger als einer Minute, in der Datenbankschreib- und Lesefehler auftreten.

Die 5xx-Fehler stellen eine gute Metrik dar, da sie die Folge des Fehlermodus sind, dem ein Client der Workload direkt unterliegen wird. Die Messung der Datenbankfehler ist als direkte Folge des Fehlers gut als Metrik geeignet, sollte jedoch durch eine Messung der Client-Auswirkungen ergänzt werden, beispielsweise in Form von fehlgeschlagenen Kundenanfragen oder Fehlern im Client. Zusätzlich sollten Sie für alle APIs oder URIs, auf die der Client Ihrer Workload direkt zugreift, eine synthetische Überwachung einrichten (auch als Benutzer-Canary bezeichnet).

e. Verbessern Sie das Workload-Design hinsichtlich der Resilienz.

Wenn der Steady-State nicht bewahrt wurde, untersuchen Sie, wie das Workload-Design verbessert werden könnte, um den Fehler zu bewältigen. Wenden Sie dabei die bewährten Methoden der AWS Well-Architected-Säule „Zuverlässigkeit“ an. Zusätzliche Anleitungen und Ressourcen finden Sie in der AWS Builder's Library. Dort finden Sie unter anderem Artikel darüber, wie Sie Ihre Zustandsprüfungen verbessern oder Wiederholungen mit Backoff in Ihrem Anwendungscode verwenden können.

Führen Sie das Experiment nach der Implementierung dieser Änderungen erneut durch (angezeigt durch die gepunktete Linie im Flywheel für das Chaos-Engineering), um ihre Effektivität zu ermitteln. Wenn der Verifizierungsschritt zeigt, dass die Hypothese zutrifft, befindet sich die Workload im Steady-State und der Zyklus wird fortgesetzt.

4. Führen Sie regelmäßig Experimente durch.

Ein Chaos-Experiment ist ein Zyklus. Daher sollten Experimente regelmäßig als Teil des Chaos-Engineering durchgeführt werden. Wenn die Hypothese eines Experiments auf eine Workload zutrifft, sollte das Experiment automatisiert werden, um innerhalb Ihrer CI/CD-Pipeline kontinuierlich als Regression ausgeführt zu werden. Wie das geht, erfahren Sie in diesem Blogbeitrag zur Durchführung von AWS FIS-Experimenten mit AWS CodePipeline, In diesem Lab zu wiederkehrenden AWS FIS-Experimenten in einer CI/CD-Pipeline können Sie praktische Erfahrungen sammeln.

Fehlerinjektionsexperimente sind auch Bestandteil von Gamedays (siehe REL12-BP05 Regelmäßiges Durchführen von Gamedays). Bei Gamedays wird ein Fehler oder Ereignis simuliert, um Systeme, Prozesse und die Reaktionen von Teams zu testen. Dabei sollen die

auszuführenden Aktionen vom Team wie im Fall eines außergewöhnlichen Ereignisses tatsächlich ausgeführt werden.

5. Erfassen und speichern Sie die Ergebnisse der Experimente.

   Die Ergebnisse von Fehlerinjektionsexperimenten müssen erfasst und gespeichert werden. Erfassen Sie dabei alle notwendigen Daten (z. B. Zeit, Workload und Bedingungen), um die Ergebnisse und Trends von Experimenten später analysieren zu können. Beispiele für erfasste Ergebnisse können Screenshots von Dashboards, CSV-Versionen der Metrikdatenbank oder manuell eingegebene Aufzeichnungen von Ereignissen und Beobachtungen während des Experiments sein. Die Protokollierung von Experimenten mit AWS FIS kann Teil dieser Datenerfassung sein.

Ressourcen

Zugehörige bewährte Methoden:

- REL08-BP03 Integrieren von Ausfallsicherheitstests in die Bereitstellung
- REL13-BP03 Testen der Implementierung der Notfallwiederherstellung zur Validierung

Zugehörige Dokumente:

- Was ist AWS Fault Injection Service?
- Was ist AWS Resilience Hub?
- Grundlagen des Chaos Engineering
- Chaos-Engineering: Planung Ihres ersten Experiments
- Resilience Engineering: Aus Fehlern lernen
- Chaos-Engineering-Geschichten
- Vermeiden von Fallback in verteilten Systemen
- Canary-Bereitstellung für Chaos-Experimente

Zugehörige Videos:

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

Zugehörige Tools:

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

REL12-BP05 Regelmäßiges Durchführen von Gamedays

Führen Sie Gamedays durch, um Ihre Verfahren zur Reaktion auf Ereignisse und Beeinträchtigungen, die sich auf die Workloads auswirken, regelmäßig zu trainieren. Beziehen Sie dieselben Teams ein, die für die Bearbeitung von Produktionsszenarien verantwortlich wären. Diese Übungen helfen, Maßnahmen durchzusetzen, um eine Beeinträchtigung der Benutzer durch Ereignisse in der Produktionsumgebung zu verhindern. Wenn Sie Ihre Reaktionsverfahren unter realistischen Bedingungen üben, können Sie etwaige Lücken oder Schwächen erkennen und beheben, bevor ein tatsächliches Ereignis eintritt.

Bei Gamedays werden Ereignisse in produktionsähnlichen Umgebungen simuliert, um Systeme, Prozesse und die Reaktionen des Teams zu testen. Dabei sollen die Aktionen ausgeführt werden, die das Team im Falle eines tatsächlichen Eintretens des Ereignisses ausführen würde. So können Sie nachvollziehen, wo nachgebessert werden kann, und üben, mit Ereignissen und Beeinträchtigungen umzugehen. Gamedays sollten regelmäßig ausgeführt werden, damit Ihr Team tief verwurzelte Reaktionsgewohnheiten aufbaut.

Gamedays bereiten die Teams darauf vor, Produktionsereignisse mit größerer Sicherheit zu bewältigen. Geübte Teams sind besser in der Lage, verschiedene Szenarien schnell zu erkennen und darauf zu reagieren. Dies führt zu einer deutlich besseren Bereitschaft und Resilienz.

Gewünschtes Ergebnis: Sie veranstalten die Gamedays für Resilienz immer wieder zu geplanten Zeitpunkten. Diese Gamedays werden als normaler und erwarteter Teil der Geschäftstätigkeit angesehen. Ihr Unternehmen hat eine Kultur der Bereitschaft entwickelt, und wenn Produktionsprobleme auftreten, sind Ihre Teams gut darauf vorbereitet, effektiv zu reagieren, die Probleme effizient zu lösen und die Auswirkungen auf die Kunden zu minimieren.

Typische Anti-Muster:

- Sie dokumentieren die eigenen Verfahren, trainieren diese aber nie.

- Entscheidungsträger werden bei den Tests nicht mit einbezogen.
- Sie veranstalten einen Gameday, informieren aber nicht alle relevanten Stakeholder.
- Sie konzentrieren sich ausschließlich auf technische Fehler, beziehen aber geschäftliche Stakeholder nicht mit ein.
- Sie lassen die bei den Gamedays gewonnenen Erkenntnisse nicht in Ihre Wiederherstellungsprozesse einfließen.
- Sie geben den Teams die Schuld für Fehler oder Bugs.

Vorteile der Nutzung dieser bewährten Methode:

- Verbesserung der Reaktionsfähigkeit: An Gamedays üben die Teams ihre Aufgaben und testen ihre Kommunikationsmechanismen bei simulierten Ereignissen. So kommt es zu einer besser koordinierten, effizienteren Reaktion in Produktionssituationen.
- Identifizierung und Behebung von Abhängigkeiten: Komplexe Umgebungen beinhalten oft vielschichtige Abhängigkeiten zwischen verschiedenen Systemen, Services und Komponenten. Gamedays können Ihnen dabei helfen, diese Abhängigkeiten zu identifizieren und zu beheben und zu überprüfen, ob Ihre kritischen Systeme und Services ordnungsgemäß durch Ihre Runbook-Verfahren abgedeckt sind und zeitnah hochskaliert oder wiederhergestellt werden können.
- Förderung einer Kultur der Resilienz: Gamedays können dazu beitragen, eine Haltung der Resilienz innerhalb eines Unternehmens zu fördern. Wenn Sie funktionsübergreifende Teams und Stakeholder einbeziehen, fördern diese Übungen das Bewusstsein, die Zusammenarbeit und ein gemeinsames Verständnis für die Bedeutung von Resilienz im gesamten Unternehmen.
- Kontinuierliche Verbesserung und Anpassung: Regelmäßige Gamedays helfen Ihnen, Ihre Resilienzstrategien kontinuierlich zu bewerten und anzupassen, sodass sie auch bei sich ändernden Umständen relevant und wirksam bleiben.
- Stärkung des Vertrauens in das System: Erfolgreiche Gamedays tragen dazu bei, Vertrauen in die Fähigkeit des Systems aufzubauen, Störungen standzuhalten und sich von ihnen zu erholen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

Sobald Sie die erforderlichen Resilienzmaßnahmen entwickelt und umgesetzt haben, führen Sie einen Gameday durch, um zu überprüfen, ob in der Produktion alles wie geplant funktioniert. An einem Gameday, insbesondere am ersten Gameday, sollten alle Teammitglieder beteiligt sein und

alle Stakeholder und Teilnehmer sollten vorab über das Datum, die Uhrzeit und die simulierten Szenarien informiert werden.

Während des Gamedays simulieren die beteiligten Teams verschiedene Ereignisse und mögliche Szenarien gemäß den vorgeschriebenen Verfahren. Die Teilnehmer beobachten und bewerten die Auswirkungen dieser simulierten Ereignisse genau. Wenn das System wie vorgesehen funktioniert, sollten die automatischen Erkennungs-, Skalierungs- und Selbstreparaturmechanismen aktiviert werden und das Ereignis sollte nur geringe bis keine Auswirkungen auf die Benutzer haben. Wenn das Team negative Auswirkungen feststellt, setzt es den Test zurück und behebt die festgestellten Probleme, entweder auf automatisiertem Wege oder durch manuelles Eingreifen, wie in den entsprechenden Runbooks dokumentiert.

Um die Resilienz kontinuierlich zu verbessern, ist es wichtig, die gewonnenen Erkenntnisse zu dokumentieren und zu berücksichtigen. Bei diesem Prozess handelt es sich um eine Feedback-Schleife, die systematisch Erkenntnisse aus Gamedays erfasst und zur Verbesserung von Systemen, Prozessen und Teamfähigkeiten nutzt.

Um leichter reale Szenarien reproduzieren zu können, in denen Systemkomponenten oder Services möglicherweise unerwartet ausfallen, können Sie simulierte Fehler als Gameday-Übung einbauen. Die Teams können die Resilienz und Fehlertoleranz ihrer Systeme testen und ihre Reaktions- und Wiederherstellungsprozesse in einer kontrollierten Umgebung simulieren.

In AWS können Ihre Gamedays mit Replikaten Ihrer Produktionsumgebung unter Verwendung von Infrastructure as Code durchgeführt werden. Auf diese Weise können Sie Ihre Tests in einer sicheren Umgebung durchführen, die Ihrer Produktionsumgebung sehr ähnlich ist. Ziehen Sie die Verwendung von AWS Fault Injection Service in Betracht, um verschiedene Ausfallszenarien zu erstellen. Verwenden Sie Services wie Amazon CloudWatch und AWS X-Ray, um das Systemverhalten an Gamedays zu überwachen. Verwenden Sie AWSSystems Manager für die Verwaltung und Ausführung von Playbooks und AWS Step Functions zur Orchestrierung von wiederkehrenden Workflows am Gameday.

Implementierungsschritte

- Einrichtung eines Programms für den Gameday: Entwickeln Sie ein strukturiertes Programm, das die Häufigkeit, den Umfang und die Ziele der Gamedays festlegt. Binden Sie wichtige Stakeholder und Fachexperten in die Planung und Durchführung dieser Übungen ein.
- Vorbereitung des Gamedays:

1. Identifizieren Sie die wichtigsten geschäftskritischen Services, um die es bei dem Gameday gehen soll. Katalogisieren Sie die Mitarbeiter, Prozesse und Technologien, die diese Services unterstützen, und ordnen Sie sie zu.
2. Legen Sie die Agenda für den Gameday fest und bereiten Sie die beteiligten Teams auf die Teilnahme an dem Event vor. Bereiten Sie Ihre Automatisierungsservices so vor, dass sie die geplanten Szenarien simulieren und die entsprechenden Wiederherstellungsprozesse ausführen. AWS-Services wie AWS Fault Injection Service, AWS Step Functions und AWS Systems Manager können Ihnen dabei helfen, verschiedene Aspekte von Gamedays zu automatisieren, beispielsweise das Einschleusen von Fehlern und das Einleiten von Wiederherstellungsaktionen.

- Ausführung Ihrer Simulation: Führen Sie am Gameday das geplante Szenario aus. Beobachten und dokumentieren Sie, wie Mitarbeiter, Prozesse und Technologien auf das simulierte Ereignis reagieren.
- Durchführung von Prüfungen nach der Übung: Führen Sie nach dem Gameday eine Retrospektive durch, um die gewonnenen Erkenntnisse zu überprüfen. Identifizieren Sie Bereiche mit Verbesserungspotenzial und alle Maßnahmen, die zur Verbesserung der betrieblichen Resilienz erforderlich sind. Dokumentieren Sie Ihre Ergebnisse und verfolgen Sie alle notwendigen Änderungen, um Ihre Resilienzstrategien und Ihre Bereitschaft zu verbessern.

Ressourcen

Zugehörige bewährte Methoden:

- REL12-BP01 Untersuchen von Fehlern mit Playbooks
- REL12-BP04 Testen der Ausfallsicherheit mit Chaos-Engineering
- OPS04-BP01 Ermitteln wichtiger Leistungskennzahlen
- OPS07-BP03 Verwenden von Runbooks zur Durchführung von Verfahren
- OPS10-BP01 Verwenden eines Prozesses für die Bewältigung von Ereignissen, Vorfällen und Problemen

Zugehörige Dokumente:

- Was ist AWS GameDay?

Zugehörige Videos:

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

Zugehörige Beispiele:

- AWS Workshop – Den Sturm überstehen: Widerstandsfähige Systeme durch kontrolliertes Chaos
- Gestalten Ihres eigenen Gamedays zur Förderung der betrieblichen Resilienz

Planung der Notfallwiederherstellung

Sicherungen und redundante Workload-Komponenten sind der Ausgangspunkt Ihrer Strategie für die Notfallwiederherstellung. RTO und RPO sind Ihre Ziele für die Wiederherstellung Ihrer Workload. Legen Sie diese entsprechend den geschäftlichen Anforderungen fest. Implementieren Sie eine Strategie, um diese Ziele zu erreichen. Berücksichtigen Sie dabei Standorte und Funktionen von Workload-Ressourcen und -Daten. Die Wahrscheinlichkeit von Unterbrechungen und die Kosten von Wiederherstellungen sind ebenfalls wichtige Faktoren bei der Ermittlung des Unternehmenswerts, den Notfallwiederherstellungen von Workloads bieten.

Sowohl Verfügbarkeit als auch Notfallwiederherstellung basieren auf denselben bewährten Methoden wie der Überwachung auf Ausfälle, der Bereitstellung an mehreren Standorten und dem automatischen Failover. Verfügbarkeit konzentriert sich jedoch auf Komponenten der Workload, während Notfallwiederherstellung sich auf einzelne Kopien der gesamten Workload konzentriert. Notfallwiederherstellung verfolgt andere Ziele als Verfügbarkeit, wobei der Schwerpunkt auf der Zeit bis zur Wiederherstellung nach einem Notfall liegt.

Best Practices

- REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten
- REL13-BP02 Verwenden von definierten Wiederherstellungsstrategien, um die Wiederherstellungsziele zu erreichen
- REL13-BP03 Testen der Implementierung der Notfallwiederherstellung zur Validierung
- REL13-BP04 Verwalten der Konfigurationsabweichungen am Standort oder in der Region der Notfallwiederherstellung
- REL13-BP05 Automatisieren der Wiederherstellung

REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten

Ausfälle können sich auf verschiedene Weise auf Ihr Unternehmen auswirken. Erstens können Ausfälle zu Betriebsunterbrechungen (Ausfallzeiten) führen. Zweitens können Ausfälle dazu führen, dass Daten verloren gehen, inkonsistent oder veraltet sind. Definieren Sie für jeden Workload ein Recovery Time Objective (RTO) und Recovery Point Objective (RPO). Das Recovery Time Objective (RTO) ist die maximal zulässige Verzögerung zwischen der Unterbrechung und der Wiederherstellung des Services. Das Recovery Point Objective (RPO) ist die maximal zulässige Zeitspanne seit dem letzten Datenwiederherstellungspunkt.

Gewünschtes Ergebnis: Für jeden Workload gibt es ein bestimmtes RTO und RPO, basierend auf technischen Überlegungen und geschäftlichen Auswirkungen.

Typische Anti-Muster:

- Sie verfügen nicht über festgelegte Wiederherstellungsziele.
- Sie wählen willkürliche Wiederherstellungsziele aus.
- Sie wählen Wiederherstellungsziele aus, die nicht strikt genug sind und die Geschäftsziele nicht erfüllen.
- Sie haben die Auswirkungen von Ausfallzeiten und Datenverlusten nicht bewertet.
- Sie wählen unrealistische Wiederherstellungsziele aus (beispielsweise sofortige Wiederherstellung oder kein Datenverlust), die für Ihre Workload-Konfiguration möglicherweise nicht erreichbar sind.
- Sie wählen Wiederherstellungsziele aus, die strikter sind als die tatsächlichen Geschäftsziele. Dies erzwingt Wiederherstellungsimplementierungen, die kostspieliger und komplizierter sind als für den Workload erforderlich.
- Sie wählen Wiederherstellungsziele aus, die nicht mit denen eines abhängigen Workloads vereinbar sind.
- Sie berücksichtigen gesetzliche Vorschriften und Compliance-Anforderungen nicht.

Vorteile der Einführung dieser bewährten Methode: Durch die Festlegung von RTOs und RPOs für Ihre Workloads definieren Sie klare und messbare Ziele für die Wiederherstellung auf der Grundlage Ihrer Geschäftsanforderungen. Sobald Sie diese Ziele festgelegt haben, können Sie Pläne für die Notfallwiederherstellung erstellen, die auf die Erreichung dieser Ziele zugeschnitten sind.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Erstellen Sie eine Matrix oder ein Arbeitsblatt, um eine bessere Planung der Notfallwiederherstellung zu ermöglichen. Erstellen Sie in Ihrer Matrix verschiedene Workload-Kategorien oder -stufen basierend auf den Auswirkungen der Workloads auf das Unternehmen (z. B. kritisch, hoch, mittel und gering) und den zugehörigen RTOs und RPOs, die Sie für jeden einzelnen Workload anstreben. Die folgende Matrix enthält ein Beispiel, dem Sie folgen können (beachten Sie, dass Ihre RTO- und RPO-Werte hiervon abweichen können):

Matrix der Notfallwiederherstellung						
		Wiederherstellungszeitpunkt				
		< 1 Minute	< 1 Stunde	< 6 Stunden	< 1 Tag	+ 1 Tag
Wiederher-stellungsdauer	< 10 Minuten	Kritisch	Kritisch	Hoch	Mittel	Mittel
	< 2 Stunden	Kritisch	Hoch	Mittel	Mittel	Niedrig
	< 8 Stunden	Hoch	Mittel	Mittel	Niedrig	Niedrig
	< 24 Stunden	Mittel	Mittel	Niedrig	Niedrig	Niedrig
	24 + Stunden	Mittel	Niedrig	Niedrig	Niedrig	Niedrig

Beispielmatrix für die Notfallwiederherstellung

Sie müssen für jeden Workload die Auswirkungen von Ausfallzeiten und Datenverlusten auf Ihr Unternehmen ermitteln und verstehen. Die Auswirkungen werden in der Regel umso größer, je länger die Ausfallzeiten sind und je größer der Datenverlust ist, die Form der Auswirkung kann jedoch je nach Art des Workloads unterschiedlich sein. So kann es beispielsweise sein, dass Ausfallzeiten von bis zu einer Stunde geringe Auswirkungen haben, die Auswirkungen danach aber schnell zunehmen. Die Auswirkungen können viele verschiedene Formen annehmen. So kann es etwa finanzielle Auswirkungen (z. B. Umsatzeinbußen), Auswirkungen auf den Ruf (einschließlich Verlust von Kundenvertrauen), betriebliche Auswirkungen (z. B. verspätete Auszahlung von Gehältern oder verringerte Produktivität) und regulatorische Risiken geben. Weisen Sie den Workload nach Abschluss des Vorgangs der entsprechenden Stufe zu.

Berücksichtigen Sie bei der Analyse der Auswirkungen eines Fehlers die folgenden Fragen:

1. Wie lange kann der Workload maximal nicht verfügbar sein, bevor dies inakzeptable Auswirkungen auf das Unternehmen hat?

2. Wie stark werden die Auswirkungen einer Unterbrechung des Workloads auf das Unternehmen sein und welche Art von Auswirkungen wird es geben? Berücksichtigen Sie alle Arten von Auswirkungen, einschließlich finanzieller, den Ruf betreffender, betrieblicher und regulatorischer Auswirkungen.
3. Wie viele Daten können maximal verlorengehen oder nicht wiederherstellbar sein, bevor dies inakzeptable Auswirkungen auf das Unternehmen hat?
4. Können verlorene Daten aus anderen Quellen wiederhergestellt werden (sogenannte abgeleitete Daten)? Wenn ja, sollten Sie auch die RPOs aller Quelldaten berücksichtigen, die zur Neuerstellung der Workload-Daten verwendet werden.
5. Was sind die Wiederherstellungsziele und Verfügbarkeitserwartungen für Workloads, von denen dieser Workload abhängt (Downstream)? Die Ziele Ihres Workloads müssen unter Berücksichtigung der Wiederherstellungsmöglichkeiten seiner nachgelagerten Abhängigkeiten erreichbar sein. Erwägen Sie mögliche Behelfslösungen oder Abhilfemaßnahmen für nachgelagerte Abhängigkeiten, die die Wiederherstellungsfähigkeit dieses Workloads verbessern können.
6. Was sind die Wiederherstellungsziele und Verfügbarkeitserwartungen für Workloads, die von diesem Workload abhängen (Upstream)? Angesichts der Ziele vorgelagerter Workloads muss dieser Workload möglicherweise über striktere Wiederherstellungsmöglichkeiten verfügen, als es zunächst den Anschein hat.
7. Gibt es je nach Art des Vorfalls unterschiedliche Wiederherstellungsziele? Sie könnten beispielsweise unterschiedliche RTOs und RPOs haben, je nachdem, ob sich der Vorfall auf eine Availability Zone oder eine gesamte Region auswirkt.
8. Ändern sich Ihre Wiederherstellungsziele während bestimmter Ereignisse oder zu bestimmten Zeiten des Jahres? Sie könnten beispielsweise unterschiedliche RTOs und RPOs für das Weihnachtsgeschäft, für Sportveranstaltungen, Sonderverkaufsaktionen und Produkteinführungen haben.
9. Wie stimmen die Wiederherstellungsziele mit den Strategien Ihrer Branche und Ihres Unternehmens für die Notfallwiederherstellung überein?
10. Gibt es rechtliche oder vertragliche Konsequenzen zu beachten? Sind Sie beispielsweise vertraglich verpflichtet, einen Service mit einem bestimmten RTO oder RPO bereitzustellen? Welche Strafen könnten Ihnen bei einer Nichteinhaltung drohen?
11. Müssen Sie die Datenintegrität wahren, um gesetzliche Vorschriften oder Compliance-Anforderungen zu erfüllen?

Das folgende Arbeitsblatt kann Ihnen bei der Bewertung der einzelnen Workloads helfen. Sie können dieses Arbeitsblatt Ihren spezifischen Bedürfnissen entsprechend ändern, indem Sie beispielsweise zusätzliche Fragen hinzufügen.

Schritt 2: primäre Fragen	Gilt für Workload?	Workload-RTO	Workload-RPO	RTO anpassen	RPO anpassen	Anleitungen
[1] Maximale Zeit, in der der Workload ausfallen kann						Gemessen Zeit seit Beginn des Ausfalls bis zur Wiederherstellung
[2] Maximale Datenmenge, die verloren gehen kann						Gemessen in Zeit seit dem letzten bekannten gut wiederherstellbaren Datensatz
[3a] Vorgelagerte Abhängigkeiten						Strengste nachgelagerte Wiederherstellungsziele eingeben
[3b] Nachgelagerte Abhängigkeiten						Am wenigsten strenge nachgelagerte Wiederherstellungsziele eingeben
[3a] Abgeglichene vorgelagerte Abhängigkeiten						Wenn der vorgelagerte Wert niedriger ist als aktuelle Werte und der nachgelagerte Wert größer ist,
[3b] Abgeglichene nachgelagerte Abhängigkeiten						arbeiten Sie mit Abhängigkeiten, um auszugleichen und hier ausgeglichene Werte einzugeben.
[3] Abhängigkeiten						Werte senken, um vorgelagerte Abhängigkeiten zu erfüllen oder die basierend auf nachgelagerten Abhängigkeitsfähigkeiten zu erhöhen
Schritt 2: zusätzliche Fragen						Geben Sie an, ob die Frage zutrifft. Falls nicht, überspringen Sie sie.
Basis-RTO/-RPO						Übertragen Sie die RTO- und RPO-Werte von oben nach hier unten.
[4] Art des Ausfalls	[ ]J/[ ]N					Geben Sie Wiederherstellungsziele für Ereignisarten mit strengsten Anforderungen ein.
[5] Spezifische zeitbasierte Ziele	[ ]J/[ ]N					Geben Sie Wiederherstellungsziele für Zeiten mit strengsten Anforderungen ein.
[6] Unterbrechungen bei Kunden	[ ]J/[ ]N					Grafische Darstellung der betroffenen Kunden in Abhängigkeit von der Ausfallzeit oder dem Datenverlust. Verwenden Sie dies, um das maximal zulässige RTO und RPO auf der Grundlage der Kundenauswirkungen einzugeben.
[7] Auswirkungen auf den Ruf	[ ]J/[ ]N					Mit dem Unternehmen arbeiten, um die maximale RTO und den maximalen RPO basierend auf der Auswirkung auf die Reputation zu bestimmen
[8] Betriebliche Auswirkungen	[ ]J/[ ]N					Geben Sie das maximale RTO und RPO auf der Grundlage der betrieblichen Auswirkungen ein.
[9] Organisatorische Ausrichtung	[ ]J/[ ]N					Geben Sie das maximale RTO und RPO für Workloads dieses Typs gemäß den LOB- und Organisationsanforderungen ein.
[10] Vertragliche Verpflichtungen	[ ]J/[ ]N					Geben Sie das maximale RTO und RPO auf der Grundlage der vertraglichen Verpflichtungen ein.
[11] Gesetzliche Vorschriften	[ ]J/[ ]N					Geben Sie das maximale RTO und RPO auf der Grundlage der geltenden gesetzlichen Bestimmungen ein.
Ziel basierend auf zusätzlichen Fragen						Nehmen Sie den Mindestwert (strengerer Wert) aus den Fragen 4–11 und geben Sie ihn hier ein.
Angepasstes Ziel						Wenn die Ziele in der obigen Zeile nicht erreicht werden können, arbeiten Sie mit den Beteiligten zusammen, um die Beschränkungen zu lockern, und geben Sie hier ein neues Minimum ein.
RTO/RPO angepasst						Geben Sie die Basis-RPO-/RTO-Werte oder das angepasste Ziel ein, je nachdem, welcher Wert niedriger ist.
Schritt 3						
Zuordnung zu vordefinierter Kategorie oder Stufe						Senken Sie beide Werte (machen Sie sie strenger), um sie an die nächstgelegene definierte Stufe anzupassen.

Arbeitsblatt

Implementierungsschritte

1. Ermitteln Sie die geschäftlichen Stakeholder und technischen Teams, die für die einzelnen Workloads verantwortlich sind, und setzen Sie sich mit ihnen in Verbindung.
2. Erstellen Sie Kategorien oder Stufen der Kritikalität für die Workload-Auswirkungen in Ihrem Unternehmen. Beispielkategorien sind u. a. „Kritisch“, „Hoch“, „Mittel“ und „Niedrig“. Wählen Sie für jede Kategorie ein Ihren Geschäftszielen und Anforderungen entsprechendes RTO und RPO.
3. Weisen Sie jedem Workload eine der im vorherigen Schritt erstellten Auswirkungskategorien zu. Berücksichtigen Sie für die Zuordnung eines Workloads zu einer Kategorie die Bedeutung des Workloads für das Unternehmen sowie die Auswirkungen von Unterbrechungen oder Datenverlusten und orientieren Sie sich an den obigen Fragen. So erhalten Sie für jeden Workload ein RTO und ein RPO.

4. Sehen Sie sich das im vorherigen Schritt ermittelte RTO und RPO für jeden Workload an. Beziehen Sie die geschäftlichen und technischen Teams für den Workload mit ein, um zu entscheiden, ob die Ziele angepasst werden sollten. Geschäftliche Stakeholder könnten beispielsweise zu dem Schluss kommen, dass strengere Ziele erforderlich sind. Alternativ könnten technische Teams feststellen, dass die Ziele so geändert werden sollten, dass sie mit den verfügbaren Ressourcen und technologischen Einschränkungen erreichbar sind.

Ressourcen

Zugehörige bewährte Methoden:

- REL09-BP04 Verifizieren der Sicherungsintegrität und -verfahren durch regelmäßiges Wiederherstellen der Daten
- REL12-BP01 Untersuchen von Fehlern mit Playbooks
- REL13-BP02 Verwenden von definierten Wiederherstellungsstrategien, um die Wiederherstellungsziele zu erreichen
- REL13-BP03 Testen der Implementierung der Notfallwiederherstellung zur Validierung

Zugehörige Dokumente:

- AWS Architecture Blog: Notfallwiederherstellung
- Notfallwiederherstellung von Workloads in AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)
- Verwalten von Ausfallsicherheitsrichtlinien mit AWS Resilience Hub
- APN-Partner: Partner, die Sie bei der Notfallwiederherstellung unterstützen können
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte

Zugehörige Videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- Notfallwiederherstellung von Workloads in AWS

REL13-BP02 Verwenden von definierten Wiederherstellungsstrategien, um die Wiederherstellungsziele zu erreichen

Definieren Sie eine Notfallwiederherstellungsstrategie (Disaster Recovery, DR), die den Wiederherstellungszielen Ihrer Workloads entspricht. Wählen Sie eine Strategie aus, z. B. Backup und Wiederherstellung, Standby (aktiv/passiv) oder Aktiv/Aktiv.

Gewünschtes Ergebnis: Für jede Workload gibt es eine definierte und implementierte DR-Strategie, mit der die Workload die DR-Ziele erreichen kann. DR-Strategien zwischen Workloads nutzen wiederverwendbare Muster (wie die zuvor beschriebenen Strategien),

Typische Anti-Muster:

- Implementierung von inkonsistenten Wiederherstellungsprozeduren für Workloads mit ähnlichen DR-Zielen.
- Die DR-Strategie muss im Notfall Ad-hoc umgesetzt werden.
- Es gibt keinen Plan für die Notfallwiederherstellung.
- Abhängigkeit von Vorgängen auf der Steuerebene während der Wiederherstellung.

Vorteile der Nutzung dieser bewährten Methode:

- Durch die Nutzung definierter Wiederherstellungsstrategien können Sie verbreitet verwendete Tools und Testverfahren verwenden.
- Die Verwendung definierter Wiederherstellungsstrategien verbessert den Wissensaustausch zwischen den Teams und die Implementierung der Notfallwiederherstellung für ihre Workloads.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch. Ohne eine geplante, implementierte und getestete DR-Strategie ist es unwahrscheinlich, dass Sie Ihre Wiederherstellungsziele im Falle eines Notfalls erreichen.

Implementierungsleitfaden

Eine DR-Strategie beruht auf der Fähigkeit, Ihre Workload an einem Wiederherstellungsstandort bereitzustellen, wenn Ihr primärer Standort nicht mehr in der Lage ist, die Workload auszuführen. Die häufigsten Wiederherstellungsziele sind RTO und RPO, wie in REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten besprochen.

Eine DR-Strategie, die mehrere Availability Zones (AZs) innerhalb eines einzigen AWS-Region umfasst, kann Katastrophenereignisse wie Brände, Überschwemmungen und größere Stromausfälle abfedern. Wenn es erforderlich ist, einen Schutz gegen ein unwahrscheinliches Ereignis zu implementieren, das verhindert, dass Ihre Workload in einer bestimmten AWS-Region ausgeführt werden kann, können Sie eine DR-Strategie verwenden, die mehrere Regionen nutzt.

Wenn Sie eine DR-Strategie für mehrere Regionen entwickeln, sollten Sie eine der folgenden Strategien wählen. Sie sind in aufsteigender Reihenfolge nach Kosten und Komplexität und in absteigender Reihenfolge nach RTO und RPO aufgeführt. Die Wiederherstellungsregion bezieht sich auf eine andere AWS-Region als die primäre Region, die für Ihre Workload verwendet wird.

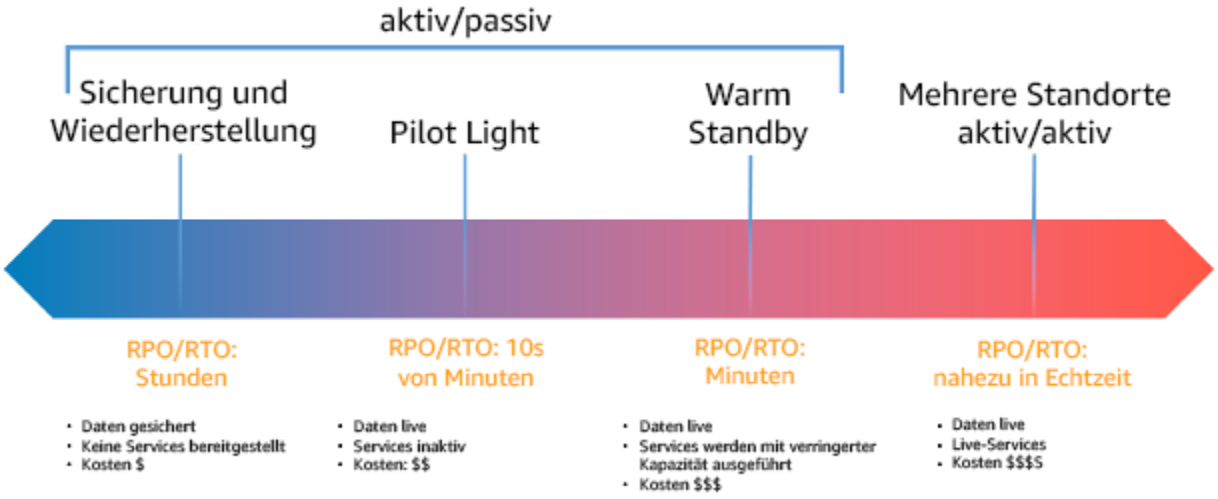


Abbildung 17: Strategien für die Notfallwiederherstellung (Disaster Recovery, DR)

- Sicherung und Wiederherstellung (RPO in Stunden, RTO in 24 Stunden oder weniger): Sichern Sie Ihre Daten und Anwendungen in der Wiederherstellungsregion. Die Verwendung automatisierter oder kontinuierlicher Backups ermöglicht eine zeitpunktbezogene Wiederherstellung (PITR), wodurch das RPO in einigen Fällen auf bis zu 5 Minuten gesenkt werden kann. Im Falle eines Notfalls stellen Sie Ihre Infrastruktur bereit (wobei Sie Infrastruktur als Code verwenden, um das RTO zu verkürzen), stellen Ihren Code bereit und stellen die gesicherten Daten wieder her, um eine Wiederherstellung nach einem Notfall in der Wiederherstellungsregion zu erfahren.
- Pilot Light (RPO in Minuten, RTO in 10-Minuten-Intervallen): Stellen Sie eine Kopie Ihrer Kern-Workload-Infrastruktur in der Wiederherstellungsregion bereit. Replizieren Sie Ihre Daten in die Wiederherstellungsregion und erstellen Sie dort Sicherungskopien der Daten. Ressourcen, die zur Unterstützung der Datenreplikation und -sicherung erforderlich sind, wie Datenbanken

und Objektspeicher, sind immer eingeschaltet. Andere Elemente wie Anwendungsserver oder Serverless-Datenverarbeitung werden nicht bereitgestellt, sondern können bei Bedarf mit der erforderlichen Konfiguration und dem Anwendungscode erstellt werden.

- Warm Standby (RPO in Sekunden, RTO in Minuten): Eine herunterskalierte, aber voll funktionsfähige Version Ihrer Workload wird dauerhaft in der Wiederherstellungsregion ausgeführt. Geschäftskritische Systeme sind vollständig dupliziert und ständig aktiv, aber mit herunterskalierter Flotte. Die Daten werden repliziert und sind in der Wiederherstellungsregion live. Wenn eine Wiederherstellung erforderlich ist, wird das System zur Bewältigung der Produktionslast schnell hochskaliert. Je höher die Skalierung des Warm Standby, desto geringer ist die Abhängigkeit von RTO und Steuerebene. Bei vollständiger Skalierung wird dies als Hot Standby bezeichnet.
- Regionsübergreifend (Multi-Site) Aktiv/Aktiv (RPO nahe Null, RTO potenziell Null): Ihre Workload wird auf mehrere AWS-Regionen verteilt und stellt aktiv Datenverkehr daraus aus. Bei dieser Strategie müssen Sie die Daten zwischen den Regionen synchronisieren. Mögliche Konflikte, die durch Schreibvorgänge auf denselben Datensatz in zwei verschiedenen regionalen Repliken verursacht werden, müssen vermieden oder behandelt werden, was sehr komplex sein kann. Die Datenreplikation ist nützlich für die Datensynchronisation und schützt Sie vor einigen Arten von Notfällen, aber sie schützt Sie nicht vor Datenbeschädigung oder -zerstörung, es sei denn, Ihre Lösung umfasst auch Optionen für eine zeitpunktbezogene Wiederherstellung.

> **Note**
>
> Der Unterschied zwischen Pilot Light und Warm Standby kann schwer zu überblicken sein. Beide beinhalten eine Umgebung in Ihrer Wiederherstellungsregion mit Kopien der Assets Ihrer Primärregion. Der Unterschied besteht darin, dass Pilot Light keine Anfragen bearbeiten kann, ohne dass zuvor zusätzliche Maßnahmen ergriffen werden, während Warm Standby den Datenverkehr (mit reduzierter Kapazität) sofort bearbeiten kann. Bei Pilot Light müssen Sie die Server einschalten, möglicherweise zusätzliche (nicht zum Kerngeschäft gehörende) Infrastruktur bereitstellen und die Leistung hochskalieren, während Sie bei Warm-Standby nur die Leistung hochskalieren müssen (alles ist bereits bereitgestellt und läuft). Wählen Sie je nach RTO- und RPO-Anforderungen zwischen diesen Varianten.
> Wenn die Kosten eine Rolle spielen und Sie ähnliche RPO- und RTO-Ziele wie bei der Warm-Standby-Strategie erreichen möchten, könnten Sie cloudnative Lösungen wie AWS Elastic Disaster Recovery in Betracht ziehen, die den Pilot-Light-Ansatz verfolgen und bessere RPO- und RTO-Ziele bieten.

Implementierungsschritte

1. Bestimmen Sie eine DR-Strategie, die die Wiederherstellungsanforderungen für diese Workload erfüllt.

   Die Wahl einer DR-Strategie ist eine Abwägung zwischen der Reduzierung von Ausfallzeiten und Datenverlusten (RTO und RPO) und den Kosten und der Komplexität der Implementierung der Strategie. Sie sollten vermeiden, eine Strategie zu verfolgen, die strikter ist als nötig, da dies unnötige Kosten verursacht.

   Im folgenden Diagramm hat das Unternehmen beispielsweise sein maximal zulässiges RTO sowie die Grenze der Ausgaben für seine Strategie zur Wiederherstellung von Diensten festgelegt. In Anbetracht der Ziele des Unternehmens erfüllen die DR-Strategien Pilot Light oder Warm Standby sowohl die RTO- als auch die Kostenkriterien.

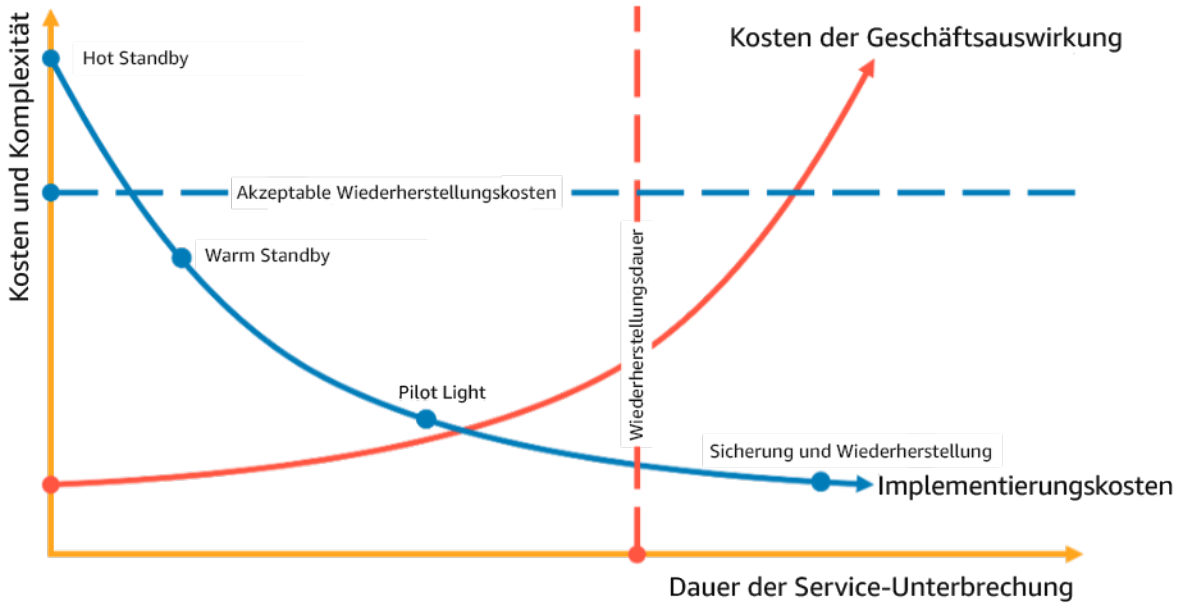


   Abbildung 18: Auswahl einer DR-Strategie auf der Grundlage von RTO und Kosten

   Weitere Informationen finden Sie unter Geschäftsfortführungsplan.

2. Sehen Sie sich anhand der Muster an, wie die gewählte DR-Strategie umgesetzt werden kann.

   In diesem Schritt geht es darum, zu verstehen, wie Sie die gewählte Strategie umsetzen wollen. Die Strategien werden durch die Verwendung von AWS-Regionen als primäre und Wiederherstellungsstandort erläutert. Sie können jedoch auch Availability Zones innerhalb einer einzigen Region als DR-Strategie verwenden, die Elemente mehrerer dieser Strategien nutzt.

In den folgenden Schritten können Sie die Strategie auf Ihre spezifische Workload anwenden.

Backup und Wiederherstellung

Backup und Wiederherstellung ist die einfachste Strategie zur Implementierung, erfordert jedoch mehr Zeit- und Arbeitsaufwand bei der Wiederherstellung der Workload, was zu einem höheren RTO und RPO führt. Es ist eine gute Vorgehensweise, immer Sicherungskopien Ihrer Daten zu erstellen und diese auf einen anderen Standort (z. B. einen anderen AWS-Region) zu kopieren.

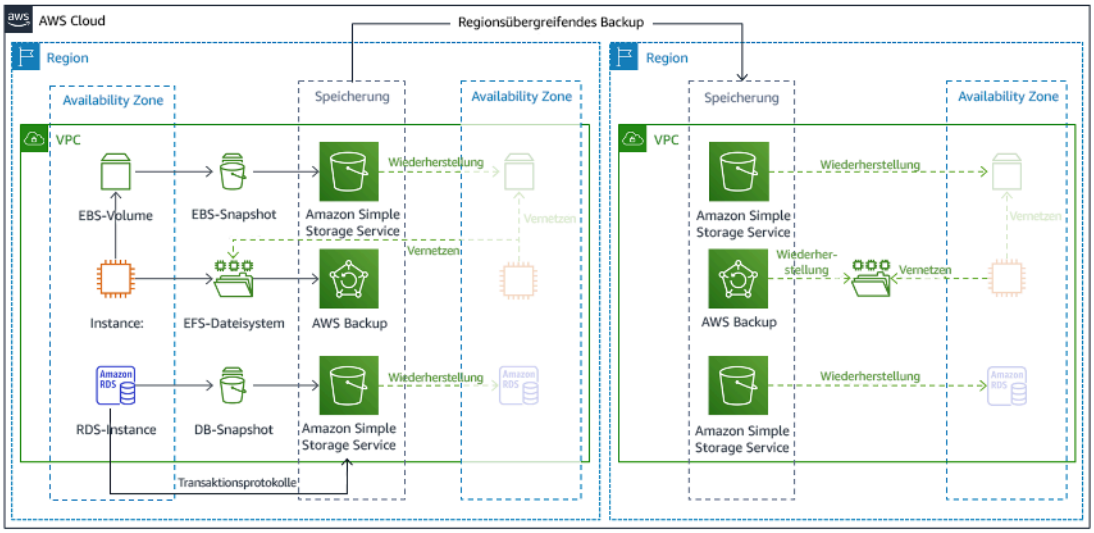


Abbildung 19: Backup- und Wiederherstellungsarchitektur

Weitere Informationen zu dieser Strategie finden Sie unter Architektur für die Notfallwiederherstellung in AWS, Teil II: Sicherung und Wiederherstellung mit Rapid Recovery.

Pilot light

Beim Pilot-Light-Ansatz replizieren Sie die Daten von Ihrer primären Region in Ihre Wiederherstellungsregion. Die Kernressourcen, die für die Workload-Infrastruktur verwendet werden, werden in der Wiederherstellungsregion bereitgestellt, jedoch werden noch zusätzliche Ressourcen und Abhängigkeiten benötigt, um diesen Stack funktionsfähig zu machen. In Abbildung 20 werden zum Beispiel keine Datenverarbeitungs-Instances bereitgestellt.

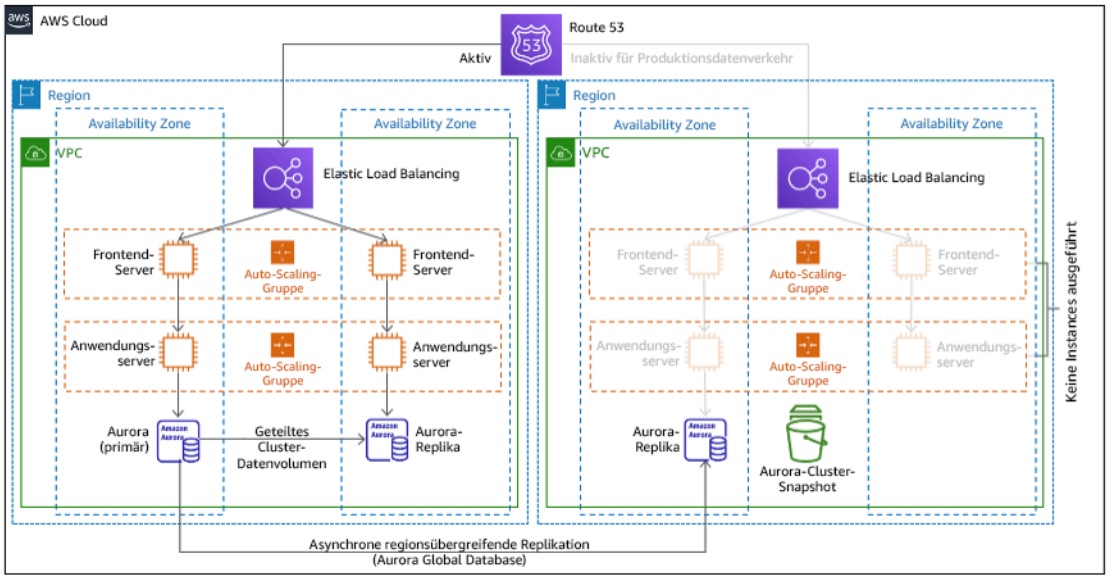


Abbildung 20: Pilot-Light-Architektur

Weitere Informationen zu dieser Strategie finden Sie unter Architektur für die Notfallwiederherstellung in AWS, Teil III: Pilot Light und Warm Standby.

Warmer Bereitschaftsmodus

Beim Warm-Standby-Ansatz wird sichergestellt, dass eine herunterskalierte, aber voll funktionsfähige Kopie Ihrer Produktionsumgebung in einer anderen Region vorhanden ist. Dieser Ansatz erweitert das Konzept des Pilot Light und verkürzt die Zeit bis zur Wiederherstellung, da die Workload in einer anderen Region ständig präsent ist. Wenn die Wiederherstellungsregion mit voller Kapazität bereitgestellt wird, wird dies als Hot Standby bezeichnet.

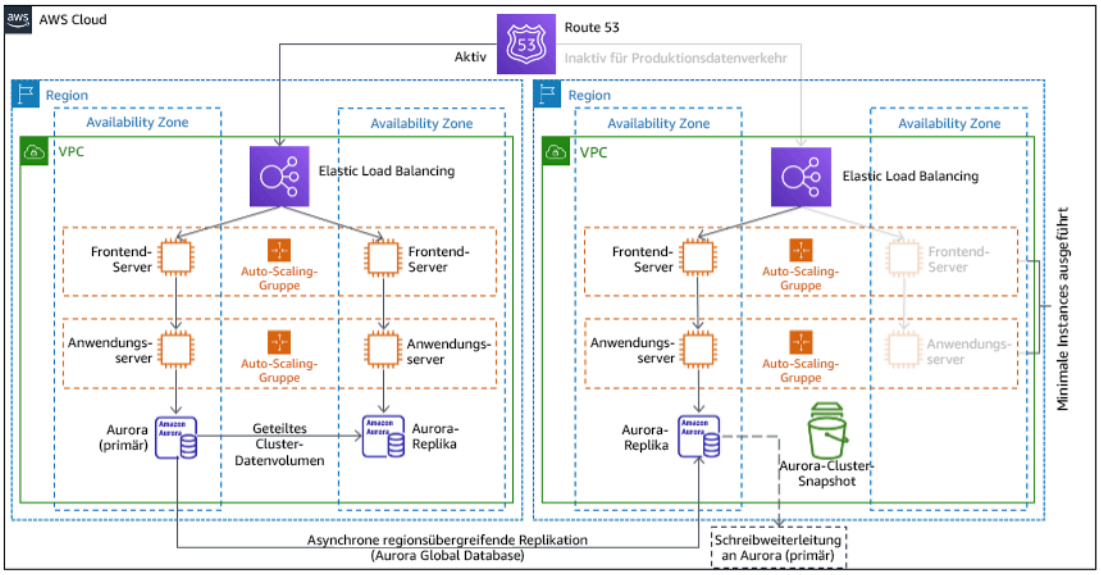


Abbildung 21: Warm-Standby-Architektur

Der Einsatz von Warm Standby oder Pilot Light erfordert ein Hochskalieren der Ressourcen in der Wiederherstellungsregion. Um zu überprüfen, ob bei Bedarf Kapazität verfügbar ist, sollten Sie die Nutzung von Kapazitätsreservierungen für EC2-Instances in Betracht ziehen. Wenn Sie AWS Lambda verwenden, kann die bereitgestellte Parallelität Laufzeitumgebungen bereitstellen, damit sie sofort auf die Aufrufe Ihrer Funktion reagieren können.

Weitere Informationen zu dieser Strategie finden Sie unter Architektur für die Notfallwiederherstellung in AWS, Teil III: Pilot Light und Warm Standby.

Multi-Site Aktiv/Aktiv

Im Rahmen einer Multi-Site Aktiv/Aktiv-Strategie können Sie Ihre Workload in mehreren Regionen gleichzeitig ausführen. Multi-Site Aktiv/Aktiv bedient den Datenverkehr aus allen Regionen, in denen es eingesetzt wird. Diese Strategie kann zur Erhöhung der Verfügbarkeit oder bei der Bereitstellung einer Workload für eine globale Zielgruppe verwendet werden (um den Endpunkt näher an die Benutzer zu bringen und/oder um Stacks bereitzustellen, die für die Zielgruppe in dieser Region lokalisiert sind). Wenn die Workload in einer der AWS-Regionen, in denen sie bereitgestellt wird, nicht unterstützt werden kann, wird diese Region evakuiert und die verbleibenden Regionen werden zur Aufrechterhaltung der Verfügbarkeit genutzt. Multi-Site Aktiv/Aktiv ist die betrieblich komplexeste der DR-Strategien und sollte nur dann gewählt werden, wenn die Geschäftsanforderungen dies erfordern.

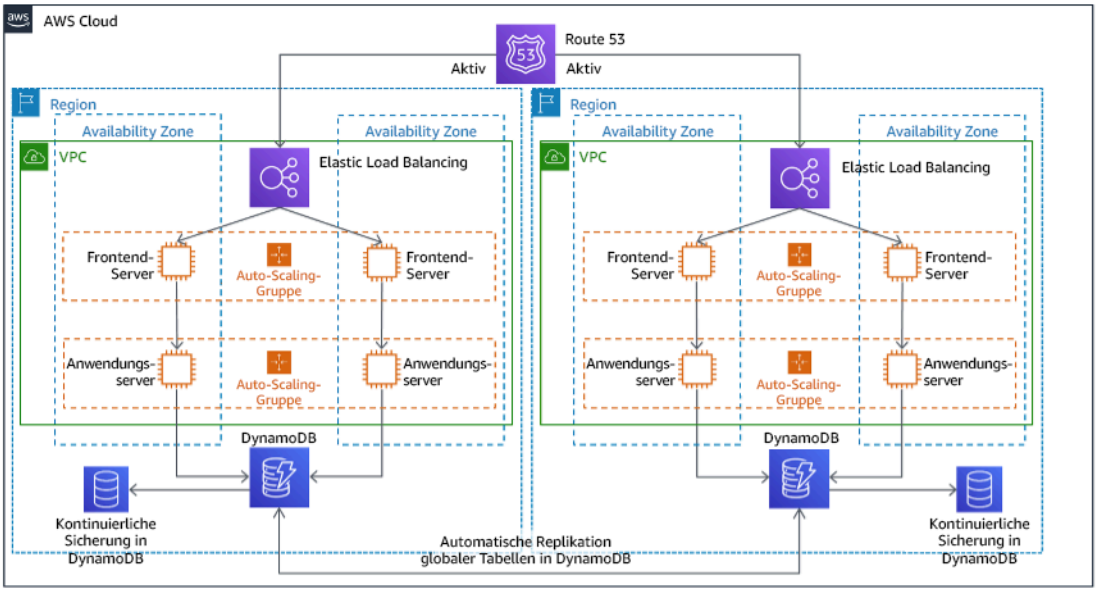


Abbildung 22: Multi-Site Aktiv/Aktiv-Architektur

Weitere Informationen zu dieser Strategie finden Sie unter Architektur für die Notfallwiederherstellung in AWS, Teil IV: Multi-Site Aktiv/Aktiv.

AWS Elastic Disaster Recovery

Wenn Sie für die Notfallwiederherstellung die Pilot-Light- oder Warm-Standby-Strategie in Betracht ziehen, könnte AWS Elastic Disaster Recovery einen alternativen Ansatz mit verbesserten Vorteilen bieten. Elastic Disaster Recovery kann ein ähnliches RPO- und RTO-Ziel wie Warm Standby bieten, behält aber den kostengünstigen Ansatz von Pilot Light bei. Elastic Disaster Recovery repliziert Ihre Daten von Ihrer primären Region auf Ihre Wiederherstellungsregion und nutzt dabei die kontinuierliche Datensicherung, um ein RPO im Sekundenbereich und ein RTO im Minutenbereich zu erreichen. In der Wiederherstellungsregion werden nur die für die Replikation der Daten erforderlichen Ressourcen bereitgestellt, was die Kosten ähnlich wie bei der Pilot-Light-Strategie niedrig hält. Bei Verwendung von Elastic Disaster Recovery koordiniert und orchestriert der Service die Wiederherstellung von Datenverarbeitungs-Ressourcen, wenn die Initiierung als Teil eines Failover oder Drills erfolgt.

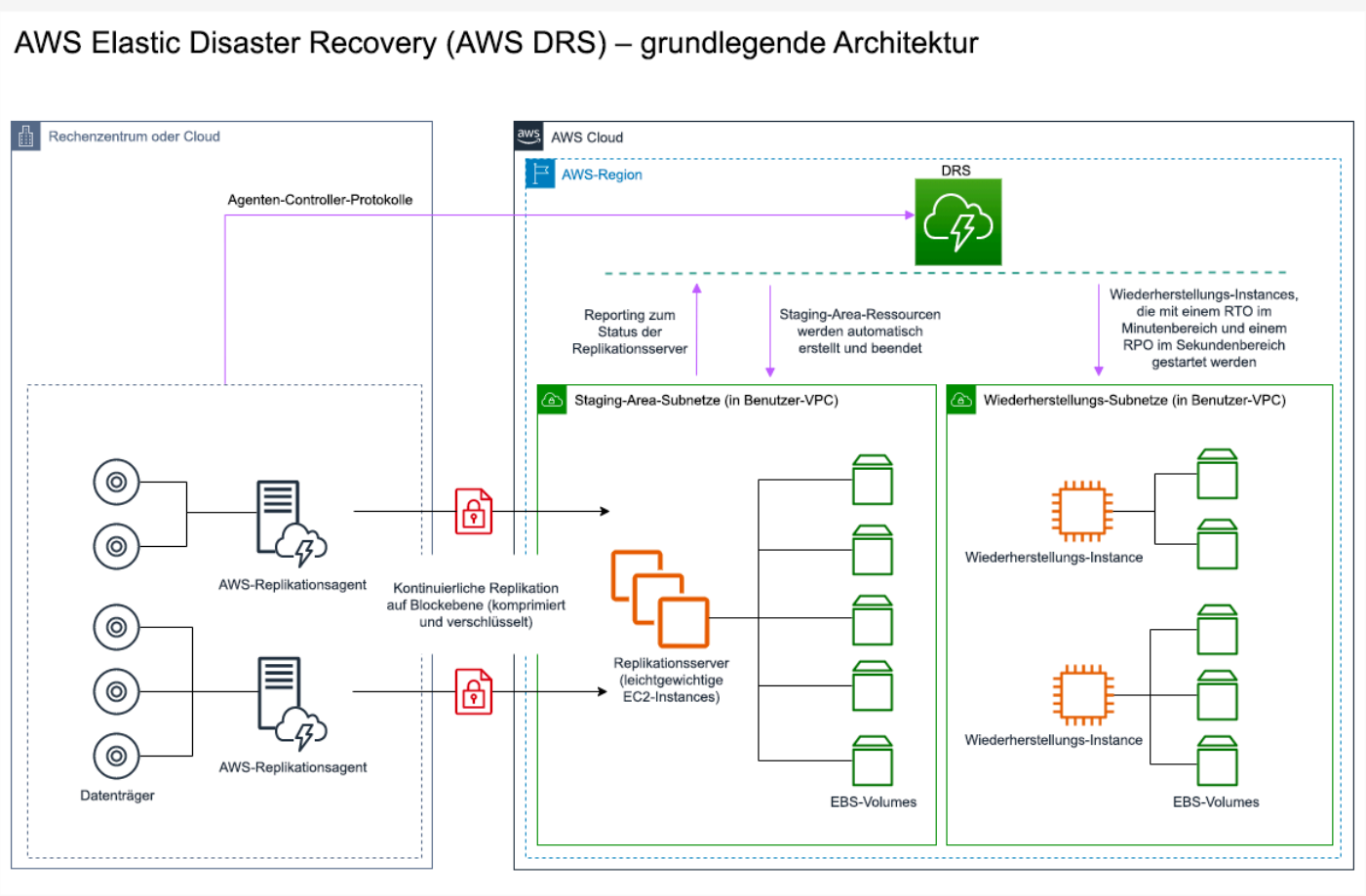


Abbildung 23: AWS Elastic Disaster Recovery-Architektur

Zusätzliche Methoden zum Schutz von Daten

Bei allen Strategien müssen Sie sich auch gegen einen Datennotfall wappnen. Kontinuierliche Datenreplikation schützt Sie vor einigen Arten von Notfällen, aber sie schützt Sie möglicherweise nicht vor Datenbeschädigung oder -zerstörung, es sei denn, Ihre Strategie umfasst auch die Versionsverwaltung gespeicherter Daten oder Optionen für eine zeitpunktbezogene Wiederherstellung. Sie müssen auch die replizierten Daten in der Wiederherstellungssite sichern, um zusätzlich zu den Replikaten zeitpunktgenaue Sicherungen zu erstellen.

Verwendung mehrerer Availability Zones (AZs) innerhalb einer einzigen AWS-Region

Wenn Sie mehrere AZs in einer einzigen Region verwenden, nutzt Ihre DR-Implementierung mehrere Elemente der oben genannten Strategien. Zunächst müssen Sie eine Hochverfügbarkeitsarchitektur (High Availability, HA) mit mehreren AZs erstellen, wie in Abbildung 23 dargestellt. Diese Architektur verwendet einen Multi-Site Aktiv/Aktiv-Ansatz, da die Amazon

EC2-Instances und der Elastic Load Balancer über Ressourcen verfügen, die in mehreren AZs bereitgestellt werden und Anfragen aktiv bearbeiten. Die Architektur demonstriert auch Hot Standby, d. h. bei einem Ausfall der primären Amazon RDS-Instance (oder der AZ selbst) wird die Standby-Instance zur primären Instance hochgestuft.

Abbildung 24: Multi-AZ-Architektur

Zusätzlich zu dieser HA-Architektur müssen Sie Backups aller Daten hinzufügen, die für die Ausführung Ihrer Workloads erforderlich sind. Dies ist besonders wichtig für Daten, die auf eine einzelne Zone beschränkt sind, etwa Amazon-EBS-Volumes oder Amazon-Redshift-Cluster. Wenn eine AZ ausfällt, müssen Sie diese Daten in einer anderen AZ wiederherstellen. Wenn möglich, sollten Sie auch Datensicherungen auf einen anderen AWS-Region kopieren, um eine zusätzliche Sicherheit zu gewährleisten.

Ein weniger verbreiteter alternativer Ansatz für einzelne Regionen, die Multi-AZ-Notfallwiederherstellung, wird im Blogbeitrag Entwickeln hoch resilienter Anwendungen mit Amazon Application Recovery Controller, Teil 1: Stack für eine einzelne Region beschrieben. Hier besteht die Strategie darin, so viel Isolation wie möglich zwischen den AZs aufrechtzuerhalten, ähnlich wie bei den Regionen. Bei dieser alternativen Strategie können Sie sich für einen Aktiv/Aktiv- oder Aktiv/Passiv-Ansatz entscheiden.

> **Note**
>
> Für einige Workloads gibt es gesetzliche Vorschriften im Hinblick auf die Datenresidenz. Wenn dies auf Ihre Workload in einer Region zutrifft, in der es derzeit nur eine AWS-

Region gibt, dann ist die Multi-Region für Ihre geschäftlichen Anforderungen nicht geeignet. Multi-AZ-Strategien bieten einen guten Schutz gegen die meisten Notfälle.

3. Bewerten Sie vor dem Failover (während des normalen Betriebs) die Ressourcen Ihrer Workloads und deren Konfiguration in der Wiederherstellungsregion.

   Verwenden Sie für Infrastruktur- und AWS-Ressourcen Infrastruktur als Code, etwa AWS CloudFormation oder Tools von Drittanbietern wie Hashicorp Terraform. Um die Bereitstellung über mehrere Konten und Regionen hinweg mit einer einzigen Operation durchzuführen, können Sie AWS CloudFormation StackSets verwenden. Bei Multi-Site-Aktiv/Aktiv- und Hot-Standby-Strategien verfügt die in Ihrer Wiederherstellungsregion bereitgestellte Infrastruktur über dieselben Ressourcen wie Ihre Primärregion. Bei den Strategien Pilot Light und Warm Standby sind zusätzliche Maßnahmen erforderlich, um die Infrastruktur produktionsreif zu machen. Mithilfe von CloudFormation-Parametern und bedingter Logik können Sie mit einer einzigen Vorlage steuern, ob ein bereitgestellter Stack aktiv oder im Standby-Modus ist. Wenn Sie Elastic Disaster Recovery verwenden, repliziert und orchestriert der Service die Wiederherstellung von Anwendungskonfigurationen und Datenverarbeitungs-Ressourcen.

   Alle DR-Strategien erfordern, dass Datenquellen innerhalb der AWS-Region gesichert werden und diese Backups anschließend in die Wiederherstellungsregion kopiert werden. AWS Backup bietet eine zentrale Ansicht, in der Sie Backups für diese Ressourcen konfigurieren, planen und überwachen können. Für Pilot Light, Warm Standby und Multi-Site Aktiv/Aktiv sollten Sie auch Daten aus der primären Region auf Datenressourcen in der Wiederherstellungsregion replizieren, etwa Amazon Relational Database Service (Amazon RDS)-DB-Instances oder Amazon DynamoDB-Tabellen. Diese Datenressourcen sind daher aktiv und bereit, Anfragen in der Wiederherstellungsregion zu bedienen.

   Weitere Informationen darüber, wie AWS-Services in verschiedenen Regionen funktionieren, finden Sie in der Blogserie Erstellen einer regionsübergreifenden Anwendung mit AWS.

4. Ermitteln und implementieren Sie, wie Sie Ihre Wiederherstellungsregion bei Bedarf (im Notfall) auf ein Failover vorbereiten.

   Bei Multi-Site Aktiv/Aktiv bedeutet Failover, dass eine Region evakuiert wird und die verbleibenden aktiven Regionen genutzt werden. Im Allgemeinen sind diese Regionen bereit, Datenverkehr aufzunehmen. Bei den Strategien Pilot Light und Warm Standby müssen Ihre Wiederherstellungsmaßnahmen die fehlenden Ressourcen bereitstellen, z. B. die EC2-Instances in Abbildung 20, sowie alle anderen fehlenden Ressourcen.

Bei allen oben genannten Strategien müssen Sie möglicherweise schreibgeschützte Instances von Datenbanken zur primären Lese-/Schreib-Instance machen.

Bei der Sicherung und Wiederherstellung werden durch die Wiederherstellung von Daten aus der Sicherung Ressourcen für diese Daten wie EBS-Volumes, RDS-DB-Instances und DynamoDB-Tabellen erstellt. Außerdem müssen Sie die Infrastruktur wiederherstellen und Code bereitstellen. Sie können AWS Backup nutzen, um Daten in der Wiederherstellungsregion wiederherzustellen. Weitere Details finden Sie unter REL09-BP01 Ermitteln und Sichern aller zu sichernden Daten oder Reproduzieren der Daten aus Quellen. Der Wiederaufbau der Infrastruktur umfasst die Erstellung von Ressourcen wie EC2-Instances zusätzlich zur Amazon Virtual Private Cloud (Amazon VPC), zu Subnetzen und zu benötigten Sicherheitsgruppen. Sie können einen Großteil des Wiederherstellungsprozesses automatisieren. Wie das geht, erfahren Sie in diesem Blogbeitrag.

5. Ermitteln und implementieren Sie, wie Sie den Datenverkehr bei Bedarf (im Notfall) zum Failover umleiten.

   Dieser Failover-Vorgang kann entweder automatisch oder manuell eingeleitet werden. Ein automatisch eingeleiteter Failover auf der Grundlage von Zustandsprüfungen oder Alarmen ist mit Vorsicht zu genießen, da ein unnötiger Failover (Fehlalarm) Kosten wie Nichtverfügbarkeit und Datenverlust verursacht. Daher wird häufig ein manuell initiierter Failover verwendet. In diesem Fall sollten Sie die Schritte für den Failover dennoch automatisieren, sodass die manuelle Auslösung wie ein Knopfdruck wirkt.

   Bei der Inanspruchnahme von AWS-Services gibt es mehrere Optionen für die Verwaltung des Datenverkehrs zu berücksichtigen. Eine Option ist die Verwendung von Amazon Route 53. Mit Amazon Route 53 können Sie mehrere IP-Endpunkte in einem oder mehreren AWS-Regionen mit einem Route-53-Domainnamen verknüpfen. Um ein manuell eingeleitetes Failover zu implementieren, können Sie Amazon Application Recovery Controller verwenden, das eine hochverfügbare Datenebenen-API zur Umleitung des Datenverkehrs in die Wiederherstellungsregion bereitstellt. Verwenden Sie bei der Implementierung von Failover Vorgänge auf der Datenebene und vermeiden Sie solche auf der Steuerebene, wie in REL11-BP04 Nutzen der Datenebene und nicht der Steuerebene während der Wiederherstellung beschrieben.

   Weitere Informationen zu diesen und anderen Optionen finden Sie in diesem Abschnitt des Whitepapers zur Notfallwiederherstellung.

6. Entwerfen Sie einen Plan für das Failback Ihrer Workload.

Failback bedeutet, dass Sie den Workload-Betrieb in der primären Region wieder aufnehmen, nachdem ein Notfallereignis abgeklungen ist. Die Bereitstellung von Infrastruktur und Code für die primäre Region erfolgt im Allgemeinen in denselben Schritten wie ursprünglich, wobei Infrastruktur als Code und Code-Bereitstellungspipelines verwendet werden. Die Herausforderung beim Failback ist die Wiederherstellung von Datenspeichern und die Sicherstellung ihrer Konsistenz mit der in Betrieb befindlichen Wiederherstellungsregion.

Im ausgefallenen Zustand sind die Datenbanken in der Wiederherstellungsregion aktiv und verfügen über die aktuellen Daten. Ziel ist es dann, eine erneute Synchronisierung von der Wiederherstellungsregion mit der primären Region vorzunehmen, um sicherzustellen, dass diese auf dem neuesten Stand ist.

Einige AWS-Services werden das automatisch tun. Wenn Sie globale Amazon DynamoDB-Tabellen verwenden, setzt DynamoDB die Propagierung aller ausstehenden Schreibvorgänge fort, sobald die Tabelle wieder online ist, auch wenn sie in der primären Region nicht mehr verfügbar war. Wenn Sie Amazon Aurora Global Database und ein verwaltetes geplantes Failover verwenden, wird die vorhandene Replikationstopologie der globalen Aurora-Datenbank beibehalten. Daher wird die ehemalige Lese-/Schreib-Instance in der primären Region zu einem Replikat und erhält Aktualisierungen von der Wiederherstellungsregion.

In Fällen, in denen dies nicht automatisch geschieht, müssen Sie die Datenbank in der primären Region als Replikat der Datenbank in der Wiederherstellungsregion neu einrichten. In vielen Fällen bedeutet dies, dass die alte primäre Datenbank gelöscht und neue Replikate erstellt werden müssen.

Wenn Sie nach einem Failover in Ihrer Wiederherstellungsregion weiterarbeiten können, sollten Sie diese zur neuen Primärregion machen. Sie würden trotzdem alle oben genannten Schritte durchführen, um die ehemalige Primärregion in eine Wiederherstellungsregion zu verwandeln. Einige Unternehmen führen eine planmäßige Rotation durch und tauschen ihre Primär- und Wiederherstellungsregionen in regelmäßigen Abständen aus (z. B. alle drei Monate).

Alle für Failover und Failback erforderlichen Schritte sollten in einem Playbook festgehalten werden, das allen Teammitgliedern zur Verfügung steht und regelmäßig überprüft wird.

Wenn Sie Elastic Disaster Recovery verwenden, hilft der Service bei der Orchestrierung und Automatisierung des Failback-Prozesses. Weitere Informationen finden Sie unter Durchführen eines Failbacks.

Aufwand für den Implementierungsplan: Hoch

Ressourcen

Zugehörige bewährte Methoden:

- the section called “REL09-BP01 Ermitteln und Sichern aller zu sichernden Daten oder Reproduzieren der Daten aus Quellen”
- the section called “REL11-BP04 Nutzen der Datenebene und nicht der Steuerebene während der Wiederherstellung”
- the section called “REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten”

Zugehörige Dokumente:

- AWS Architecture Blog: Notfallwiederherstellung
- Notfallwiederherstellung von Workloads in AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)
- Optionen für die Notfallwiederherstellung in der Cloud
- Entwickeln einer Multi-Region-Serverless-Backend-Lösung, die aktiv/aktiv ist
- Multi-Region-Serverless-Backend – neu aufgelegt
- RDS: Regionsübergreifendes Replizieren von Lesereplikaten
- Route 53: Konfiguration des DNS-Failovers
- S3: Regionsübergreifende Replikation
- Was ist AWS Backup?
- Was ist Amazon Application Recovery Controller?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Erste Schritte – AWS
- APN-Partner: Partner, die Sie bei der Notfallwiederherstellung unterstützen können
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte

Zugehörige Videos:

- Notfallwiederherstellung von Workloads in AWS

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Erste Schritte mit AWS Elastic Disaster Recovery | Amazon Web Services

REL13-BP03 Testen der Implementierung der Notfallwiederherstellung zur Validierung

Testen Sie regelmäßig den Failover zu Ihrem Wiederherstellungsstandort, um zu überprüfen, ob er ordnungsgemäß funktioniert und ob das RTO und RPO eingehalten werden.

Typische Anti-Muster:

- Failover sollten nie in der Produktion getestet werden.

Vorteile der Nutzung dieser bewährten Methode: Durch regelmäßige Tests des Notfallwiederherstellungsplans ist gewährleistet, dass er bei Bedarf funktioniert und von Ihrem Team umgesetzt werden kann.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Vom Erstellen selten durchgeführter Wiederherstellungspfade wird abgeraten. So könnten Sie beispielsweise einen zweiten Datenspeicher unterhalten, der nur für Leseabfragen verwendet wird. Wenn Sie Daten in einen Datenspeicher schreiben und der primäre Datenspeicher einen Fehler ausgibt, können Sie einen Failover auf den zweiten Datenspeicher durchführen. Wenn Sie diesen Failover nicht regelmäßig testen, werden Sie möglicherweise feststellen, dass Ihre Annahmen zu den Möglichkeiten des sekundären Datenspeichers unzutreffend sind. Die Kapazität des sekundären Datenspeichers, die bei den letzten Tests möglicherweise noch ausreichend war, genügt möglicherweise nicht mehr den Anforderungen dieses Szenarios. Unsere Erfahrungen haben gezeigt, dass bei einer Wiederherstellung nach einem Fehler nur der Pfad funktioniert, den Sie regelmäßig testen. Daher ist es ratsam, mehrere Wiederherstellungspfade zu pflegen. Sie können Wiederherstellungsmuster erstellen und diese regelmäßig testen. Auch komplexe oder kritische Wiederherstellungspfade müssen regelmäßig mittels Fehlersimulationen in der Produktion durchgeführt werden, um sicherzustellen, dass sie funktionieren. In dem gerade besprochenen Beispiel sollten Sie regelmäßig und unabhängig von der Erfordernis einen Failover auf die Standby-Ressourcen durchführen.

Implementierungsschritte

1. Legen Sie die Workloads für die Wiederherstellung aus. Testen Sie regelmäßig Ihre Wiederherstellungspfade Die Recovery-orientierte Datenverarbeitung identifiziert die Merkmale von Systemen, die die Wiederherstellung verbessern: Isolierung und Redundanz, systemweite Fähigkeit zur Rücknahme von Änderungen, Fähigkeit zur Überwachung und Bestimmung des Zustands, Fähigkeit zur Diagnose, automatisierte Wiederherstellung, modularer Aufbau und Fähigkeit zum Neustart. Testen Sie den Wiederherstellungspfad, um zu überprüfen, ob Sie die Wiederherstellung in der angegebenen Zeit und in dem angegebenen Zustand durchführen können. Dokumentieren Sie während dieser Wiederherstellung auftretende Probleme in Ihren Runbooks und suchen Sie vor dem nächsten Test nach Lösungen.
2. Verwenden Sie für Amazon EC2-basierte Workloads AWS Elastic Disaster Recovery, um Drill-Instances für Ihre DR-Strategie zu implementieren und zu starten. AWS Elastic Disaster Recoverybietet die Möglichkeit, Drills effizient durchzuführen, sodass Sie sich auf ein Failover-Ereignis vorbereiten können. Sie können Ihre Instances mit Elastic Disaster Recovery außerdem regelmäßig zu Test- und Übungszwecken starten, ohne den Datenverkehr weiterleiten zu müssen.

Ressourcen

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Notfallwiederherstellung unterstützen können
- AWS Architecture Blog: Notfallwiederherstellung
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte
- AWS Elastic Disaster Recovery
- Notfallwiederherstellung von Workloads in AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)
- AWS Elastic Disaster Recovery – Vorbereitung auf das Failover
- The Berkeley/Stanford Recovery-Oriented Computing Project
- Was ist AWS Fault Injection Simulator?

Zugehörige Videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Backup-and-Restore and Disaster-Recovery Solutions with AWS

REL13-BP04 Verwalten der Konfigurationsabweichungen am Standort oder in der Region der Notfallwiederherstellung

Um ein erfolgreiches Verfahren für die Notfallwiederherstellung durchführen zu können, müssen Ihre Workloads in der Lage sein, den normalen Betrieb zeitnah wieder aufzunehmen, ohne dass es zu nennenswerten Funktions- oder Datenverlusten kommt, sobald die Umgebung für die Notfallwiederherstellung online gegangen ist. Um dieses Ziel zu erreichen, ist es unerlässlich, eine konsistente Infrastruktur sowie konsistente Daten und Konfigurationen zwischen Ihrer Umgebung für die Notfallwiederherstellung und der primären Umgebung aufrechtzuerhalten.

Gewünschtes Ergebnis: Die Konfiguration und die Daten Ihres Standorts für die Notfallwiederherstellung entsprechen denen des primären Standorts, wodurch bei Bedarf eine schnelle und vollständige Wiederherstellung möglich ist.

Typische Anti-Muster:

- Sie versäumen es, die Wiederherstellungsstandorte zu aktualisieren, wenn Änderungen an den primären Standorten vorgenommen werden. Dadurch kommt es zu veralteten Konfigurationen, die die Wiederherstellungsbemühungen behindern könnten.
- Sie lassen mögliche Einschränkungen wie Serviceunterschiede zwischen primärem Standort und Wiederherstellungsstandort, die zu unerwarteten Fehlern beim Failover führen können, außer Acht.
- Sie verlassen sich bei der Aktualisierung und Synchronisierung der Umgebung für die Notfallwiederherstellung auf manuelle Prozesse, was das Risiko von menschlichen Fehlern und Inkonsistenzen erhöht.
- Sie erkennen Konfigurationsabweichungen nicht und haben daher vor einem Vorfall fälschlicherweise das Gefühl, dass der Standort für die Notfallwiederherstellung bereit ist.

Vorteile der Einführung dieser bewährten Methode: Durch die Konsistenz zwischen der Umgebung für die Notfallwiederherstellung und der primären Umgebung erhöht sich die Wahrscheinlichkeit einer erfolgreichen Wiederherstellung nach einem Vorfall erheblich und das Risiko eines fehlgeschlagenen Wiederherstellungsvorgangs verringert sich.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Ein umfassender Ansatz für das Konfigurationsmanagement und die Failover-Bereitschaft kann Ihnen dabei helfen, sicherzustellen, dass der Standort für die Notfallwiederherstellung regelmäßig aktualisiert wird und bei einem Ausfall des primären Standorts zur Übernahme bereit ist.

Um Konsistenz zwischen Ihrer primären Umgebung und Ihrer Umgebung für die Notfallwiederherstellung zu erreichen, sollten Sie sicherstellen, dass Ihre Bereitstellungs-Pipelines Anwendungen sowohl an Ihren primären Standort als auch an Ihren Standort für die Notfallwiederherstellung verteilen. Führen Sie Änderungen an den Standorten für die Notfallwiederherstellung nach einem angemessenen Evaluierungszeitraum durch (sogenannte gestaffelte Bereitstellungen), um Probleme am primären Standort zu erkennen und die Bereitstellung zu stoppen, bevor sie sich ausbreiten. Implementieren Sie eine Überwachung, um Konfigurationsabweichungen zu erkennen und Änderungen sowie die Einhaltung von Vorschriften in Ihren Umgebungen nachzuverfolgen. Führen Sie automatisierte Problembehebungen am Standort für die Notfallwiederherstellung durch, um vollständige Konsistenz sicherzustellen und dafür zu sorgen, dass der Standort bei einem Vorfall übernahmebereit ist.

Implementierungsschritte

1. Stellen Sie sicher, dass die Region für die Notfallwiederherstellung die AWS-Services und -Features enthält, die für eine erfolgreiche Ausführung Ihres Plans für die Notfallwiederherstellung erforderlich sind.
2. Verwenden Sie Infrastructure as Code (IaC). Sorgen Sie dafür, dass Ihre Konfigurationsvorlagen für die Produktionsinfrastruktur und die Anwendungen korrekt sind, und wenden Sie die Vorlagen regelmäßig auf Ihre Umgebung für die Notfallwiederherstellung an. AWS CloudFormation kann Abweichungen zwischen den Vorgaben Ihrer CloudFormation-Vorlagen und der tatsächlichen Bereitstellung erkennen.
3. Konfigurieren Sie CI/CD-Pipelines, um Anwendungen und Infrastrukturupdates für alle Umgebungen bereitzustellen, einschließlich primärer Standorte und Standorte für die Notfallwiederherstellung. CI/CD-Lösungen wie AWS CodePipeline können den Bereitstellungsprozess automatisieren, wodurch das Risiko von Konfigurationsabweichungen verringert wird.
4. Staffeln Sie die Bereitstellungen zwischen der primären Umgebung und der Umgebung für die Notfallwiederherstellung. Auf diese Weise können Updates zunächst in der primären Umgebung bereitgestellt und getestet und so Probleme am primären Standort isoliert werden, bevor die Updates an den Standort für die Notfallwiederherstellung weitergegeben werden.

Dieser Ansatz verhindert, dass Fehler gleichzeitig an die Produktion und den Standort für die Notfallwiederherstellung weitergegeben werden, und stellt die Integrität der Umgebung für die Notfallwiederherstellung sicher.

5. Überwachen Sie die Ressourcenkonfigurationen sowohl in der primären Umgebung als auch in der Umgebung für die Notfallwiederherstellung kontinuierlich. Lösungen wie AWS Config können helfen, die Einhaltung von Konfigurationen durchzusetzen und Abweichungen zu erkennen. Auf diese Weise können konsistente Konfigurationen in allen Umgebungen aufrechterhalten werden.
6. Implementieren Sie Warnmechanismen, damit jegliche Konfigurationsabweichungen oder Unterbrechungen oder Verzögerungen der Datenreplikation nachverfolgt werden und eine entsprechende Benachrichtigung erfolgt.
7. Automatisieren Sie die Behebung erkannter Konfigurationsabweichungen.
8. Planen Sie regelmäßige Überwachungen und Konformitätsprüfungen ein, um die fortlaufende Abstimmung zwischen der primären Konfiguration und der Konfiguration für die Notfallwiederherstellung zu überprüfen. Regelmäßige Überprüfungen helfen Ihnen dabei, die Einhaltung definierter Regeln sicherzustellen und etwaige Unstimmigkeiten zu identifizieren, die behoben werden müssen.
9. Prüfen Sie, ob es Diskrepanzen in Bezug auf die von AWS bereitgestellte Kapazität, die Service Quotas, die Drosselungsgrenzen, Konfigurationen und Versionen gibt.

Ressourcen

Zugehörige bewährte Methoden:

- REL01-BP01 Kenntnis von Service Quotas und Einschränkungen
- REL01-BP02 Verwalten von Service Quotas für mehrere Konten und Regionen
- REL01-BP04 Überwachen und Verwalten von Kontingenten
- REL13-BP03 Testen der Implementierung der Notfallwiederherstellung zur Validierung

Zugehörige Dokumente:

- Korrigieren von nicht konformen AWS-Ressourcen mit AWS-Config-Regeln
- AWS Systems Manager Automation
- AWS CloudFormation: Erkennen von nicht verwalteten Konfigurationsänderungen an Stacks und Ressourcen
- AWS CloudFormation: Ermitteln von Abweichungen im gesamten CloudFormation-Stack

- AWS Systems Manager Automation
- Notfallwiederherstellung von Workloads in AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)
- Wie implementiere ich eine Lösung für die Verwaltung der Infrastrukturkonfiguration in AWS?
- Korrigieren von nicht konformen AWS-Ressourcen mit AWS-Config-Regeln

Zugehörige Videos:

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

Zugehörige Beispiele:

- CloudFormation Registry
- Quota Monitor for AWS
- Implementieren der automatischen Behebung von Abweichungen für AWS CloudFormation unter Verwendung von Amazon CloudWatch und AWS Lambda
- AWS Architecture Blog: Notfallwiederherstellung
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte
- Automatisierung sicherer, vollautomatischer Bereitstellungen

REL13-BP05 Automatisieren der Wiederherstellung

Implementieren Sie getestete und automatisierte Wiederherstellungsmechanismen, die zuverlässig, beobachtbar und reproduzierbar sind, um das Risiko und die Auswirkungen von Ausfällen auf das Geschäft zu reduzieren.

Gewünschtes Ergebnis: Sie haben einen gut dokumentierten, standardisierten und gründlich getesteten Automatisierungs-Workflow für Wiederherstellungsprozesse implementiert. Ihre Wiederherstellungsautomatisierung behebt automatisch kleinere Probleme, bei denen das Risiko von Datenverlusten oder Nichtverfügbarkeit gering ist. Sie sind in der Lage, bei schwerwiegenden Vorfällen schnell Wiederherstellungsprozesse aufzurufen, das Problembehebungsverhalten während der Ausführung zu beobachten und die Prozesse zu beenden, wenn Sie gefährliche Situationen oder Fehler beobachten.

Typische Anti-Muster:

- Für Ihren Wiederherstellungsplan sind Sie auf Komponenten oder Mechanismen angewiesen, die ausgefallen oder beeinträchtigt sind.
- Ihre Wiederherstellungsprozesse erfordern manuelle Eingriffe, z. B. den Zugriff auf die Konsole (auch als ClickOps bezeichnet).
- Sie leiten automatisch Wiederherstellungsverfahren in Situationen ein, in denen ein hohes Risiko von Datenverlusten oder Nichtverfügbarkeit besteht.
- Sie beziehen keinen Mechanismus (wie z. B. ein Andon-Cord oder eine große rote Stopptaste) ein, mit dem Sie ein Wiederherstellungsverfahren abbrechen können, das nicht funktioniert oder zusätzliche Risiken birgt.

Vorteile der Nutzung dieser bewährten Methode:

- Höhere Zuverlässigkeit, Vorhersehbarkeit und Konsistenz der Wiederherstellungsvorgänge.
- Fähigkeit, strengere Ziele für die Wiederherstellung, einschließlich Recovery Time Objective (RTO) und Recovery Point Objective (RPO), zu erfüllen.
- Geringere Wahrscheinlichkeit, dass die Wiederherstellung während eines Vorfalls fehlschlägt.
- Geringeres Risiko von Fehlern im Zusammenhang mit manuellen Wiederherstellungsprozessen, die anfällig für menschliche Fehler sind.

Risikostufe bei fehlender Befolgung dieser bewährten Methode: Mittel

Implementierungsleitfaden

Um die automatisierte Wiederherstellung zu implementieren, benötigen Sie einen umfassenden Ansatz, der AWS-Services und bewährte Methoden nutzt. Identifizieren Sie zunächst kritische Komponenten und potenzielle Fehlerquellen in Ihrem Workload. Entwickeln Sie automatisierte Prozesse, mit denen Sie Ihre Workloads und Daten nach Ausfällen ohne menschliches Eingreifen wiederherstellen können.

Entwickeln Sie Ihre Wiederherstellungsautomatisierung unter Berücksichtigung der Prinzipien von Infrastructure as Code (IaC). Auf diese Weise wird Ihre Wiederherstellungsumgebung mit der Quellumgebung konsistent und es ist eine Versionsverwaltung Ihrer Wiederherstellungsprozesse möglich. Um komplexe Wiederherstellungs-Workflows zu orchestrieren, sollten Sie Lösungen wie AWS Systems Manager Automations oder AWS Step Functions in Betracht ziehen.

Die Automatisierung von Wiederherstellungsprozessen bietet erhebliche Vorteile und kann Ihnen helfen, Ihr Recovery Time Objective (RTO) und Recovery Point Objective (RPO) leichter zu

erreichen. Die Prozesse können jedoch auf unerwartete Situationen stoßen. Diese können zu einem Ausfall der Prozesse führen oder neue Risiken wie zusätzliche Ausfallzeiten und Datenverluste mit sich bringen. Um dieses Risiko zu minimieren, sollten Sie die Möglichkeit bieten, eine laufende Wiederherstellungsautomatisierung schnell zu unterbrechen. Nach der Unterbrechung des Prozesses können Sie Nachforschungen anstellen und Korrekturmaßnahmen ergreifen.

Für unterstützte Workloads sollten Sie Lösungen wie AWS Elastic Disaster Recovery (AWSDRS) in Betracht ziehen, um ein automatisiertes Failover zu ermöglichen. AWS DRS repliziert Ihre Computer (einschließlich Betriebssystem, Systemstatuskonfiguration, Datenbanken, Anwendungen und Dateien) kontinuierlich in einen Staging-Bereich in Ihrem Ziel-AWS-Konto und in Ihrer bevorzugten Region. Bei einem Vorfall automatisiert AWS DRS die Konvertierung Ihrer replizierten Server in vollständig bereitgestellte Workloads in Ihrer Wiederherstellungsregion in AWS.

Die Wartung und Verbesserung der automatisierten Wiederherstellung ist ein fortlaufender Prozess. Testen und verfeinern Sie Ihre Wiederherstellungsverfahren kontinuierlich auf der Grundlage der gewonnenen Erkenntnisse und halten Sie sich über neue AWS-Services und -Features auf dem Laufenden, die Ihre Wiederherstellungsmöglichkeiten verbessern können.

Implementierungsschritte

1. Planen der automatisierten Wiederherstellung
    a. Führen Sie eine gründliche Überprüfung Ihrer Workload-Architektur, Ihrer Komponenten und Abhängigkeiten durch, um automatisierte Wiederherstellungsmechanismen zu identifizieren und zu planen. Unterteilen Sie die Abhängigkeiten Ihres Workloads in harte und weiche Abhängigkeiten. Harte Abhängigkeiten sind Abhängigkeiten, ohne die der Workload nicht funktionieren kann und für die kein Ersatz bereitgestellt werden kann. Weiche Abhängigkeiten sind Abhängigkeiten, die der Workload normalerweise nutzt, die aber durch temporäre Ersatzsysteme oder -prozesse ersetzt werden können oder die durch eine Graceful Degradation bewältigt werden können.
    b. Richten Sie Prozesse ein, um fehlende oder beschädigte Daten zu identifizieren und wiederherzustellen.
    c. Definieren Sie Schritte zur Bestätigung eines wiederhergestellten stabilen Zustands nach Abschluss der Wiederherstellungsmaßnahmen.
    d. Berücksichtigen Sie alle Maßnahmen, die erforderlich sind, um das wiederhergestellte System vollständig einsatzbereit zu machen, z. B. das Vorwärmen und das Auffüllen von Caches.
    e. Denken Sie an Probleme, die während des Wiederherstellungsprozesses auftreten könnten, und überlegen Sie, wie Sie diese erkennen und beheben können.

   f. Stellen Sie sich Szenarien vor, in denen kein Zugriff auf den primären Standort und die zugehörige Steuerungsebene möglich ist. Stellen Sie sicher, dass Wiederherstellungsaktionen unabhängig durchgeführt werden können, ohne auf den primären Standort angewiesen zu sein. Ziehen Sie Lösungen wie Amazon Application Recovery Controller (ARC) in Betracht, um den Datenverkehr umzuleiten, ohne dass DNS-Einträge manuell mutiert werden müssen.

2. Entwicklung eines automatisierten Wiederherstellungsprozesses

   a. Implementieren Sie automatisierte Fehlererkennungs- und Failover-Mechanismen für eine Wiederherstellung ohne manuelle Eingriffe. Erstellen Sie Dashboards, beispielsweise mit Amazon CloudWatch, um den Fortschritt und den Zustand automatisierter Wiederherstellungsverfahren zu melden. Schließen Sie Verfahren zur Validierung einer erfolgreichen Wiederherstellung ein. Stellen Sie einen Mechanismus bereit, um eine laufende Wiederherstellung abzubrechen.

   b. Erstellen Sie Playbooks als Ausweichlösung für Fehler, die nicht automatisch behoben werden können, und berücksichtigen Sie Ihren Plan für die Notfallwiederherstellung.

   c. Testen Sie die Wiederherstellungsprozesse, wie in REL13-BP03 beschrieben.

3. Vorbereitung auf die Wiederherstellung

   a. Evaluieren Sie den Zustand Ihres Wiederherstellungsstandorts und stellen Sie wichtige Komponenten im Voraus bereit. Weitere Informationen finden Sie unter REL13-BP04.

   b. Definieren Sie klare Rollen, Verantwortlichkeiten und Entscheidungsprozesse für Wiederherstellungsoperationen und beziehen Sie dabei die relevanten Stakeholder und Teams im gesamten Unternehmen ein.

   c. Definieren Sie die Bedingungen für die Einleitung Ihrer Wiederherstellungsprozesse.

   d. Erstellen Sie einen Plan, um den Wiederherstellungsprozess rückgängig zu machen und bei Bedarf, oder nachdem dies als sicher erachtet wird, auf Ihren primären Standort zurückzugreifen.

Ressourcen

Zugehörige bewährte Methoden:

- REL07-BP01 Automatisiertes Abrufen oder Skalieren von Ressourcen
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler
- REL13-BP02: Verwenden von definierten Wiederherstellungsstrategien, um die Wiederherstellungsziele zu erreichen

- REL13-BP03 Testen der Implementierung der Notfallwiederherstellung zur Validierung
- REL13-BP04 Verwalten der Konfigurationsabweichungen am Standort oder in der Region der Notfallwiederherstellung

Zugehörige Dokumente:

- AWS Architecture Blog: Notfallwiederherstellung
- Notfallwiederherstellung von Workloads in AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)
- Orchestrieren der Automatisierung der Notfallwiederherstellung mit Amazon Route 53 ARC und AWS Step Functions
- Erstellen von AWS-Systems-Manager-Automation-Runbooks mithilfe von AWS CDK
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte
- AWS Systems Manager Automation
- AWS Elastic Disaster Recovery
- Verwenden von Elastic Disaster Recovery für Failover und Failback
- AWS Elastic Disaster Recovery – Ressourcen
- APN-Partner: Partner, die Sie bei der Notfallwiederherstellung unterstützen können

Zugehörige Videos:

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

Schlussfolgerung

Ganz gleich, ob diese Themen zu Verfügbarkeit und Zuverlässigkeit neu für Sie sind oder ob Sie bereits über große Erfahrungen verfügen und nach Einblicken suchen, um die Verfügbarkeit Ihrer geschäftskritischen Workloads zu maximieren, hoffen wir, dass dieses Whitepaper Ihnen Denkanstöße gegeben, neue Ideen hervorgebracht oder neue Sichtweisen aufgezeigt hat. Wir hoffen, dass Sie mit diesem Whitepaper das richtige Maß an Verfügbarkeit auf Basis Ihrer geschäftlichen Anforderungen abschätzen und die dazu nötigen Designentscheidungen bezüglich Zuverlässigkeit treffen können. Wir empfehlen Ihnen, die hier angebotenen Empfehlungen für Design, Betrieb und Wiederherstellung sowie das Wissen und die Erfahrung unserer AWS Solution Architects zu nutzen. Wir würden uns freuen, von Ihnen zu hören — insbesondere über Ihre Erfolgsgeschichten mit hoher Verfügbarkeit. AWS Wenden Sie sich an das für Sie zuständige Vertriebsteam oder kontaktieren Sie uns über unsere Website.

Mitwirkende

Zu den Mitwirkenden an diesem Dokument gehören:

- Michael Fischer, Principal Solutions Architect, Amazon Web Services
- Seth Eliot, Principal Developer Advocate, Amazon Web Services
- Mahanth Jayadeva, Solutions Architect – Well-Architected, Amazon Web Services
- Amulya Sharma, Principal Solutions Architect, Amazon Web Services
- Jason DiDomenico, Senior Solutions Architect – Cloud Foundations, Amazon Web Services
- Marcin Bednarz, Principal Solutions Architect, Amazon Web Services
- Tyler Applebaum, Senior Solutions Architect, Amazon Web Services
- Rodney Lester, Principal Solutions Architect, Amazon Web Services
- Joe Chapman, Senior Solutions Architect, Amazon Web Services
- Adrian Hornsby, Principal System Development Engineer, Amazon Web Services
- Kevin Miller, Vice President – S3, Amazon Web Services
- Shannon Richards, Principal Technical Program Manager, Amazon Web Services
- Laurent Domb, Chief Technologist - Fed Fin, Amazon Web Services
- Kevin Schwarz, Sr. Solutions Architect, Amazon Web Services
- Rob Martell, Principal Cloud Resilience Architect, Amazon Web Services
- Priyam Reddy, Senior Solutions Architect Manager DR, Amazon Web Services
- Jeff Ferris, Principal Technologist, Amazon Web Services
- Matias Battaglia, Senior Solutions Architect, Amazon Web Services

Weitere Informationen

Weitere Informationen finden Sie unter:

- AWS Well-Architected Framework
- AWS -Architekturzentrum

Dokumentversionen

Abonnieren Sie den RSS-Feed, um über Aktualisierungen des Whitepapers benachrichtigt zu werden.

Änderung	Beschreibung	Datum
Leitfäden zu bewährten Methoden aktualisiert	Bewährte Verfahren wurden mit neuen Leitlinien in den folgenden Bereichen aktualisiert: REL 1, REL 2, REL 4, REL 6, REL 7, REL 8, REL 10, REL 12 und REL 13. Die Leitlinien wurden für die gesamte Säule erweitert und präzisiert. Die Leitlinien in REL10-BP02 und REL12-BP03 wurden mit anderen bewährten Verfahren zusammengeführt. Die Ressourcen wurden in der gesamten Säule aktualisiert.	6. November 2024
Leitfäden zu bewährten Methoden aktualisiert	Kleine Aktualisierungen der bewährten Methoden in REL 2, 4, 5, 6, 7 und 8	27. Juni 2024
Leitfäden zu bewährten Methoden aktualisiert	Bewährte Methoden wurden mit neuen Leitfäden in den folgenden Bereichen aktualisiert: Entwerfen von Interaktionen in einem verteilten System, um Ausfälle zu vermeiden, Entwerfen von Interaktionen in einem verteilten System, um Ausfälle abzumildern oder zu verkrafte	6. Dezember 2023

	n, Überwachen von Workload-Ressourcen, Entwerfen einer Workload, die sich an Bedarfsänderungen anpasst, Implementierung von Änderungen und Testen der Zuverlässigkeit.	
Leitfäden zu bewährten Methoden aktualisiert	Bewährte Methoden wurden mit neuen Leitfäden in den folgenden Bereichen aktualisiert: Überwachen von Workload-Ressourcen und Entwerfen von Workloads , die Komponentenausfälle verkraften.	3. Oktober 2023
Leitfäden zu bewährten Methoden aktualisiert	Bewährte Methoden wurden mit neuen Leitfäden in den folgenden Bereichen aktualisiert: Entwerfen Ihrer Workload-Servicearchitektur, Entwerfen von Interaktionen in einem verteilten System, um Ausfälle abzumildern oder zu verkraften und Überwachen von Workload-Ressourcen.	13. Juli 2023
Kleines Update	Entfernung von nicht-inklusiver Sprache	13. April 2023
Aktualisierungen für das neue Framework	Bewährte Methoden mit verbindlichen Anleitungen aktualisiert und neue bewährte Methoden hinzugefügt.	10. April 2023

Whitepaper aktualisiert	Bewährte Methoden mit neuen Implementierungsanleitungen aktualisiert.	15. Dezember 2022
Kleinere Updates	Abbildungsnummern korrigiert und generell kleinere Änderungen vorgenommen.	17. November 2022
Whitepaper aktualisiert	Weitere bewährte Methoden und Verbesserungspläne hinzugefügt.	20. Oktober 2022
Whitepaper aktualisiert	In den Abschnitten Fehlerisolierung zum Schutz von Workloads und Entwurf von Workloads, die Komponentenausfälle verkraften, wurden der Säule „Zuverlässigkeit“ zwei neue bewährte Methoden hinzugefügt.	5. Mai 2022
Whitepaper aktualisiert	Aktualisierung der Informationen zur Notfallwiederherstellung und Einbeziehung des Route 53 Application Recovery Controller. Hinzufügen von Verweisen auf DevOps Guru. Aktualisierung verschiedener Ressourcenlinks und andere geringfügige redaktionelle Änderungen.	26. Oktober 2021
Kleines Update	Informationen zu AWS Fault Injection Service (AWS FIS) hinzugefügt.	15. März 2021
Kleines Update	Geringfügige Aktualisierung des Texts.	04. Januar 2021

Whitepaper aktualisiert	Anhang A aktualisiert, um das Verfügbarkeitsdesignziel für Amazon SQS, Amazon SNS und Amazon MQ anzupasse n; Neuanordnung der Zeilen in der Tabelle für bessere Übersichtlichkeit; Verbesser ung der Erläuterung des Unterschieds zwischen Verfügbarkeit und Notfallwi ederherstellung und der Erklärung, wie beide zur Ausfallsicherheit beitragen; Erweiterung der Abdeckung von multiregionalen Architekt uren (für Verfügbarkeit) und multiregionalen Strategien (für Notfallwiederherstellung); Aktualisierung des Buchs, auf das verwiesen wird, auf die neueste Version; Erweiteru ng der Verfügbarkeitsbere chnungen und Einbezieh ung von anfragebasierten Berechnungen und schnellen Berechnungen; Verbesserung der Beschreibung für Ernstfall übungen	7. Dezember 2020
Kleines Update	Anhang A aktualisiert, um das Verfügbarkeitsdesignziel für AWS Lambda anzupassen	27. Oktober 2020

Kleines Update	Anhang A aktualisiert, um das Verfügbarkeitsdesignziel für AWS Global Accelerator hinzuzufügen	24. Juli 2020

Aktualisierungen für das neue Framework	Erhebliche Aktualisierungen und neue/überarbeitete Inhalte, einschließlich: Abschnitt mit bewährten Methoden zu „Workload -Architektur“ hinzugefü gt, bewährte Methoden in die Bereiche Änderungs management und Fehlerbew ältigung unterteilt, Ressource n aktualisiert, Aktualisierungen zur Einbeziehung neuester AWS-Ressourcen und - Services wie AWS Global Accelerator, AWS Service Quotas und AWS Transit Gateway, Definitionen für Zuverlässigkeit, Verfügbar keit und Ausfallsicherheit hinzugefügt, Whitepaper besser abgestimmt auf AWS Well-Architected Tool (Fragen und bewährte Methoden) für Well-Architected-Überprüfun gen, Designprinzipien neu angeordnet, Automatische Wiederherstellung nach einem Fehler vor Testen von Wiederherstellungsverfahren verschoben, Diagramme und Formate für Gleichung en aktualisiert, Kapitel über wichtige Services entfernt und stattdessen Referenzen zu wichtigen AWS-Services in die	8. Juli 2020

	bewährten Methoden integrier t.	
Kleines Update	Fehlerhafter Link behoben	1. Oktober 2019
Whitepaper aktualisiert	Anhang A aktualisiert	1. April 2019
Whitepaper aktualisiert	Spezifische AWS Direct Connect-Netzwerkem pfehlungen und zusätzliche Service-Designziele hinzugefü gt	1. September 2018
Whitepaper aktualisiert	Abschnitte zu konzeptionellen Grundsätzen und zur Limit-Verwaltung hinzugefügt. Links aktualisiert, Mehrdeutigkeit bei Upstream-/Downstream-Terminologie entfernt und explizite Verweise auf die übrigen Themen der Säule für Zuverlässigkeit in den Verfügbarkeitsszenarios hinzugefügt.	1. Juni 2018
Whitepaper aktualisiert	Regionsübergreifende DynamoDB-Lösung in Globale Tabellen für DynamoDB Global geändert Service-D esignziele hinzugefügt	1. März 2018
Kleinere Updates	Geringfügige Korrektur bei der Verfügbarkeitsberechnung zur Berücksichtigung der Anwendungsverfügbarkeit	1. Dezember 2017

Whitepaper aktualisiert	Aktualisiert, um Hilfestellung bei Hochverfügbarkeitsdesigns zu geben, darunter Konzepte, bewährte Methoden und beispielhafte Implementierungen.	1. November 2017
Erste Veröffentlichung	Säule „Zuverlässigkeit“ des AWS Well-Architected-Framework veröffentlicht.	1. November 2016

Hinweise

Kunden sind dafür verantwortlich, Ihre eigene unabhängige Bewertung der Informationen in diesem Dokument vorzunehmen. Dieses Dokument: (a) dient nur zu Informationszwecken, (b) stellt aktuelle AWS Produktangebote und Praktiken dar, die ohne vorherige Ankündigung geändert werden können, und (c) stellt keine Verpflichtungen oder Zusicherungen von AWS und seinen verbundenen Unternehmen, Lieferanten oder Lizenzgebern dar. AWS Produkte oder Dienstleistungen werden „wie sie sind“ ohne ausdrückliche oder stillschweigende Garantien, Zusicherungen oder Bedingungen jeglicher Art bereitgestellt. Die Verantwortlichkeiten und Verbindlichkeiten AWS gegenüber seinen Kunden werden durch AWS Vereinbarungen geregelt, und dieses Dokument ist weder Teil einer Vereinbarung zwischen AWS und seinen Kunden noch ändert es diese.

AWS Glossar

Die neueste AWS Terminologie finden Sie im AWS Glossar in der AWS-Glossar Referenz.