AWS Well-Architected Framework

Säule „Zuverlässigkeit“

Säule „Zuverlässigkeit“: AWS Well-Architected Framework

Table of Contents

Säule „Zuverlässigkeit“ – AWS Well-Architected Framework

Veröffentlichungsdatum: 6. Dezember 2023 (Dokumentversionen)

Der Schwerpunkt dieses Dokuments liegt auf 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:

- Operative Exzellenz
- Sicherheit
- Zuverlässigkeit
- Leistungseffizienz
- Kostenoptimierung
- Nachhaltigkeit

Dieses Dokument legt den Fokus auf die Säule für 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 Nutzer in technologischen Rollen, z. B. CTOs (Chief Technology Officers), Architekten, Entwickler und Mitglieder von Operations-Teams. 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 eines Workloads, die beabsichtigte Funktion erwartungsgemäß korrekt und konsistent auszuführen. Dies umfasst die Möglichkeit, den Workload während des gesamten Lebenszyklus zu betreiben und zu testen. Dieses Dokument bietet umfassende Informationen mit Best Practices 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.

Verantwortungsbereich von AWS – 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, die Software, das Netzwerk und die Einrichtungen, die AWS Cloud-Services bereitstellen. AWS unternimmt wirtschaftlich vertretbare Anstrengungen, um diese AWS Cloud-Services verfügbar zu machen und sicherzustellen, dass die Service-Verfügbarkeit die AWS-Service Level Agreements (SLAs) erfüllt oder übertrifft.

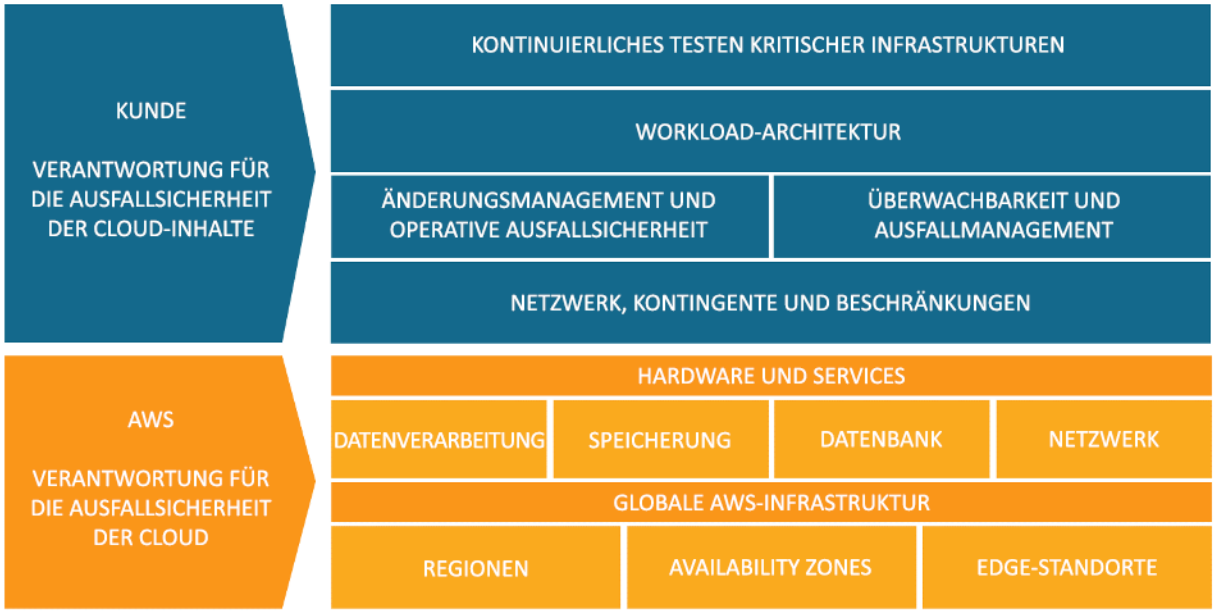
Die globale Cloud-Infrastruktur von AWS ist so konzipiert, dass sie Kunden in die Lage versetzt, hochgradig belastbare Workload-Architekturen zu erstellen. Jede AWS-Region ist vollständig isoliert und besteht aus mehreren Availability Zones, die physisch isolierte Partitionen der Infrastruktur sind. Availability Zones isolieren Fehler, die die Ausfallsicherheit des 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.

Verantwortungsbereich des Kunden – 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 (z. B. AWS Availability Zones), die Self-Healing-Implementierung mithilfe von Services wie Auto Scaling und die Anwendung der bewährten Methoden der resilienten Workload-Architektur für die auf den Instances installierten Anwendungen verantwortlich. Bei verwalteten Services wie Amazon S3 und Amazon DynamoDB verwaltet AWS die Infrastrukturschicht, das Betriebssystem und die Plattformen. Die Kunden greifen auf die Endpunkte zu, um Daten zu speichern und abzurufen. Sie sind dafür verantwortlich, die Ausfallsicherheit Ihrer Daten zu verwalten, einschließlich Backup-, Versionsverwaltungs- und Replikationsstrategien.

Das Bereitstellen Ihres Workloads über mehrere 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 den 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- und außerhalb der Cloud 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 Details 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-Kontingente und die Beschränkungen der genutzten Services. Berücksichtigen Sie die erwartete Zunahme der Last, falls anwendbar.
- 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.

Überwachbarkeit und Ausfallmanagement

- Die Überwachung von Ausfällen ist erforderlich, um das Self-Healing zu automatisieren, damit Ihre Workloads Komponentenausfällen überstehen können.
- Das Ausfallmanagement erfordert eine Datensicherung, die Anwendung bewährter Methoden, um Ihren Workload gegen Komponentenausfälle zu aktivieren, und die Planung einer Notfallwiederherstellung.

Workload-Architektur

- Ihre Workload-Architektur umfasst die Art und Weise, wie Sie Services um geschäftliche Bereiche herum entwerfen, SOA und das Design verteilter Systeme anwenden, um Fehler zu vermeiden, und Funktionen wie die Drosselung, Wiederholungen, das 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 AWS-Microservices und verwaltete Services, 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 Game-Day-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) eines Workloads 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 der 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 Szenarios reproduzieren, die zuvor zu Fehlern geführt haben. Diese Vorgehensweise legt Fehlerpfade offen, die Sie testen und beheben können, bevor ein echtes Fehlerszenario auftritt. Dadurch werden die Risiken verringert.
- 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 aufweisen.
- Genaue Analyse der verfügbaren Kapazität: Eine häufige Fehlerursache bei On-Premises-Workloads ist die Ressourcensättigung. Ein solches Szenario liegt vor, wenn die Anforderungen an den Workload die Kapazität dieses Workloads 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 Verwalten von Servicekontingenten 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 Servicekontingente und Netzwerktopologie dem 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
- Availability
- Notfallwiederherstellungsziele

Ausfallsicherheit und die Komponenten der Zuverlässigkeit

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

- Unter Ausfallsicherheit versteht man die Fähigkeit eines Workloads, sich von Infrastruktur- oder Service-Unterbrechungen zu erholen, Computing-Ressourcen 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 des Workloads auswirken:

- Betriebliche 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 Ihren 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 Frameworks abgedeckt. Viele der Best Practices hier kommen auch diesen Aspekten der Zuverlässigkeit zugute, der Fokus liegt aber auf der Ausfallsicherheit.

Availability

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

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

Verfügbar 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 die Verfügbarkeit wie folgt:

$$Verfügbarkeit = \frac{Verfügbar\ für\ Nutzungszeit}{Gesamtzeit}$$

- 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 Service-Ausfallzeiten (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.

Availability	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 Werkzeuge wie Wissensmanagement, Projektverfolgung
99,95 %	4 Stunden und 22 Minuten	Online-Handel, Verkaufsort
99,99 %	52 Minuten	Workloads zur Videoberitstellung und für Broadcasts

Availability	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:

$$Verfügbarkeit = \frac{Erfolgreiche\ Reaktionen}{Gültige\ Anforderungen}$$

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 das System 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 %:

$Avail_{günstiger} = Avail_{MAX} - ((100\ \% - Avail_{\text{-}Abhängigkeit}) \times (100\ \% - Avail_{\text{-}Abhängigkeit}))$

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 (siehe Anhang A: Konzipiert für Verfügbarkeit ausgewählter 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 Ausfallzeit (Mean Time Between Failure (MTBF)) und der mittleren Reparaturzeit (Mean Time to Recover (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 (Verfügbarkeit und mehr: Verstehen und Verbessern der Ausfallsicherheit verteilter Systeme auf 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 Entwicklung 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.

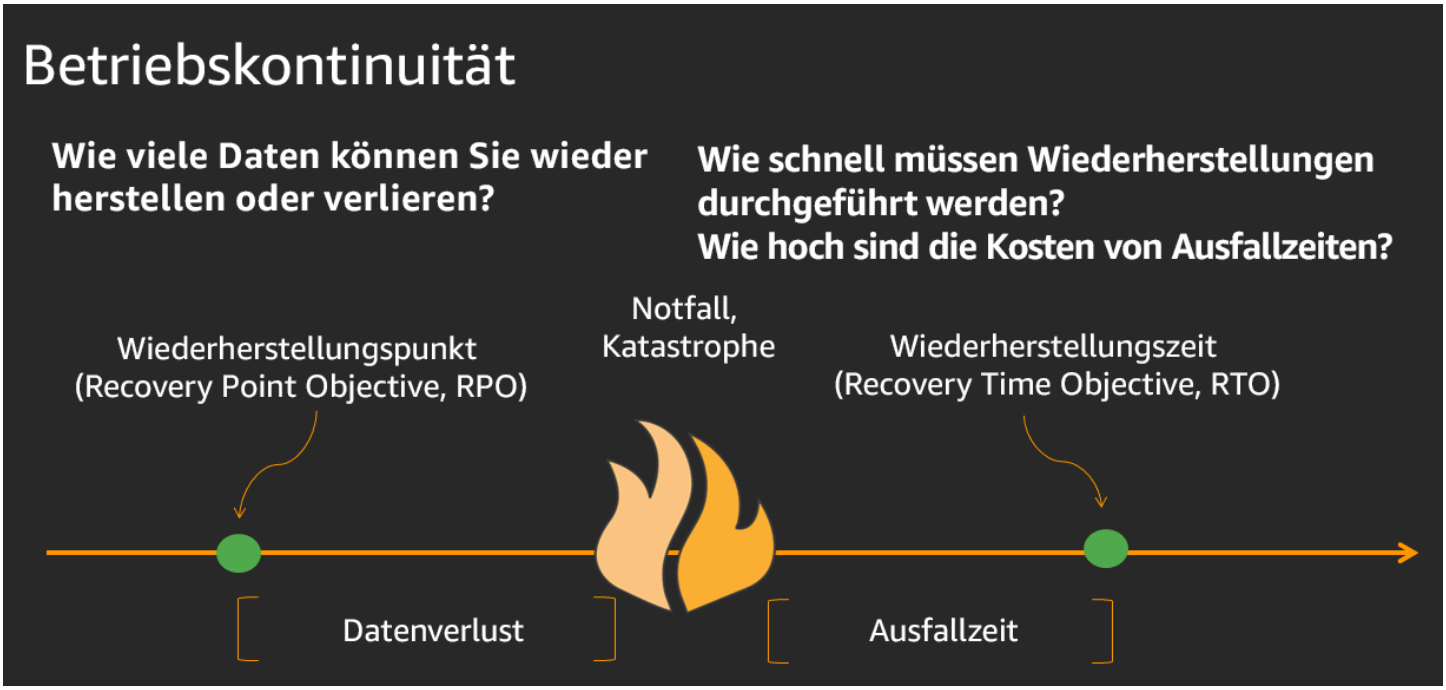
Eine andere Möglichkeit, dass Kosten in Systemen eskalieren, die unter Designzielen für eine höhere Verfügbarkeit betrieben werden, ist die Auswahl von Abhängigkeiten. Unter diesen höheren Zielen wird sich das Angebot an Software und Services, die als Abhängigkeiten ausgewählt werden können, abhängig 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 Verfugbarkeitszielen 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 Service-Unterbrechung gilt.



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

RTO ist ähnlich wie MTTR (Mean Time to Recovery), da beide die Zeit zwischen dem Start eines Ausfalls und der Wiederherstellung des Workloads 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 Echtzeit-Vorgänge 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 Echtzeit-Services, während die Kontrollebene 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 Datentransportvorgänge. 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 Kontrollebene. Ein hoher Verfügbarkeitsgrad ist für all diese Funktionen wichtig. Allerdings verfolgt die Datenebene in der Regel Designziele für eine höhere Verfügbarkeit als die Kontrollebene. 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 Ihres Workloads. 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 einen 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 des Workloads beteiligt sind, über das Framework und die Auswirkungen ihres Workloads auf die Geschäftsanforderungen informiert sind.

Grundlagen

Grundlegende Anforderungen sind diejenigen, deren Umfang über einen einzelnen 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 lokalen Umgebung können diese Anforderungen aufgrund von Abhängigkeiten lange Durchlaufzeiten zur Folge haben, daher sollten sie schon in der ersten Planungsphase berücksichtigt werden. In AWS sind die meisten dieser grundlegenden Anforderungen bereits berücksichtigt oder können nach Bedarf erfüllt werden. Die Cloud bietet nahezu unbegrenzte Möglichkeiten. Daher liegt es in der Verantwortung von AWS, die Anforderungen in Bezug auf ausreichende Netzwerk- und Rechenkapazität zu erfüllen, sodass Sie die Ressourcengröße und die Zuweisungen nach Bedarf ändern können.

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

Themen

Verwalten von Servicekontingenten und Einschränkungen

Für cloudbasierte Workload-Architekturen gibt es Servicekontingente (die auch als Service Limits bezeichnet werden). Diese Kontingente dienen dazu, nicht versehentlich mehr Ressourcen bereitzustellen als nötig und Anfrageraten für API-Vorgänge zu begrenzen, um Services vor Missbrauch zu schützen. Darüber hinaus gibt es Ressourceneinschränkungen, z. B. die Rate, mit der Bits durch ein Glasfaserkabel geschleust werden können, oder die Speichermenge 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.

Bewährte Methoden

REL01-BP01 Kenntnis von Servicekontingenten 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, wenn diese bewährte Methode nicht eingeführt wird: 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 Servicekontingent-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 Servicekontingente auf die Leistung Ihrer Anwendung auswirken und eine Anpassung an Ihre Anforderungen nicht möglich ist, wenden Sie sich an den AWS 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 begrenzt ein Servicekontingent 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 (10.0000 iops) 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 AWS 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 Servicekontingent-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 Management Console 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 Servicekontingenten 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 Ihrer 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 Management Console oder über den Befehl `list-stack-resources` AWS CLI in der Befehlszeilenschnittstelle 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 Servicekontingente. Nutzen Sie die programmgesteuert über Trusted Advisor und Service Quotas zugänglichen Informationen.
- Richten Sie eine automatisierte Überwachungsmethode ein (siehe REL01-BP02 Verwalten von Servicekontingenten für mehrere Konten und Regionen und REL01-BP04 Überwachen und Verwalten von Kontingenten), 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 Servicekontingenten für mehrere Konten und Regionen und REL01-BP04 Überwachen und Verwalten von Kontingenten).
- 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 Servicekontingenten für mehrere Konten und Regionen
- REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur
- REL01-BP04 Überwachen und Verwalten von Kontingenten
- 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 Segmentierung Ihres Workloads
- 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-BP05 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework’s Reliability Pillar: Availability (Säule für Zuverlässigkeit des AWS Well-Architected Framework)
- 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 for AWS (Kontingentüberwachung für AWS)
- AWS Fault Isolation Boundaries (AWS-Grenzen für die Fehlerisolierung)
- Availability with redundancy (Verfügbarkeit mit Redundanz)
- AWS für Daten
- What is Continuous Integration? (Was ist Continuous integration?)
- What is Continuous Delivery? (Was ist Continuous Delivery?)
- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Verwaltung des Kontolebenszyklus in SaaS-Umgebungen mit Konto pro Mandant auf AWS)
- Verwalten und Überwachen der API-Drosselung in Ihren Workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Umfangreiche AWS Trusted Advisor-Empfehlungen mit AWS Organizations anzeigen)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatisieren von Service-Limit-Erhöhungen und Enterprise Support mit AWS Control Tower)

Zugehörige Videos:

- AWS Live re:Inforce 2019 – Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Kontingente für AWS Services, die Service Quotas verwenden, anzeigen und verwalten)
- AWS IAM Quotas Demo (AWS IAM-Kontingente – 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 Servicekontingenten 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 Resilienzdesign 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 Servicekontingente 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 Servicekontingenten und Einschränkungen
- REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur
- REL01-BP04 Überwachen und Verwalten von Kontingenten
- 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 Segmentierung Ihres Workloads
- 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-BP05 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework’s Reliability Pillar: Availability (Säule für Zuverlässigkeit des AWS Well-Architected Framework)
- 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 for AWS (Kontingentüberwachung für AWS)
- AWS Fault Isolation Boundaries (AWS-Grenzen für die Fehlerisolierung)
- Availability with redundancy (Verfügbarkeit mit Redundanz)

- AWS für Daten
- What is Continuous Integration? (Was ist Continuous integration?)
- What is Continuous Delivery? (Was ist Continuous Delivery?)
- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Verwaltung des Kontolebenszyklus in SaaS-Umgebungen mit Konto pro Mandant auf AWS)
- Verwalten und Überwachen der API-Drosselung in Ihren Workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Umfangreiche AWS Trusted Advisor-Empfehlungen mit AWS Organizations anzeigen)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatisieren von Service-Limit-Erhöhungen und Enterprise Support mit AWS Control Tower)

Zugehörige Videos:

- AWS Live re:Inforce 2019 – Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Kontingente für AWS Services, die Service Quotas verwenden, anzeigen und verwalten)
- AWS IAM Quotas Demo (AWS IAM-Kontingente – 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 veränderbare Service-Kontingente, Service-Einschränkungen und physische Ressourcenbeschränkungen. Entwerfen Sie Architekturen für Anwendungen und Services, um zu verhindern, dass sich diese Beschränkungen 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, wenn diese bewährte Methode nicht eingeführt wird: 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 die Spalten `ANPASSBAR = Nein`anzeigen, 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 Servicekontingenten und Einschränkungen
- REL01-BP02 Verwalten von Servicekontingenten für mehrere Konten und Regionen
- REL01-BP04 Überwachen und Verwalten von Kontingenten
- 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 Segmentierung Ihres Workloads
- 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-BP05 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework’s Reliability Pillar: Availability (Säule für Zuverlässigkeit des AWS Well-Architected Framework)
- 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 for AWS (Kontingentüberwachung für AWS)

- AWS Fault Isolation Boundaries (AWS-Grenzen für die Fehlerisolierung)
- Availability with redundancy (Verfügbarkeit mit Redundanz)
- AWS für Daten
- What is Continuous Integration? (Was ist Continuous integration?)
- What is Continuous Delivery? (Was ist Continuous Delivery?)
- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Verwaltung des Kontolebenszyklus in SaaS-Umgebungen mit Konto pro Mandant auf AWS)
- Verwalten und Überwachen der API-Drosselung in Ihren Workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Umfangreiche AWS Trusted Advisor-Empfehlungen mit AWS Organizations anzeigen)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatisieren von Service-Limit-Erhöhungen und Enterprise Support mit 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 (Kontingente für AWS Services, die Service Quotas verwenden, anzeigen und verwalten)
- AWS IAM Quotas Demo (AWS IAM-Kontingente – Demo)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: Details und Strategien: Wie man die Kontrolle über große und kleine Systeme übernimmt)

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 Überwachen und Verwalten von Kontingenten

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.
- Es werden nur Alarme für Services konfiguriert, die von AWS Service Quotas unterstützt werden, und es erfolgt keine Überwachung anderer AWS-Services.
- 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 Nutzung dieser bewährten Methode: Automatische Verfolgung der AWS Service Quotas und die Überwachung der Nutzung dieser Kontingente ermöglichen die Erkennung von

nahen Kontingentbeschränkungen. Sie können diese Überwachungsdaten außerdem nutzen, um Verschlechterungen aufgrund einer Kontingentausschöpfung zu begrenzen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: 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 Warnungen können von AWS Config, Lambda-Funktionen, Amazon CloudWatch oder von AWS Trusted Advisor ausgelöst werden. Sie können außerdem metrische Filter auf CloudWatch Logs verwenden, um Muster in den Protokollen zu suchen und zu extrahieren, und so festzustellen, ob sich die Nutzung den Schwellenwerten für Kontingente nähert.

Implementierungsschritte

Für die Überwachung:

- Erfassen Sie den aktuellen Ressourcenverbrauch (z. B. Buckets oder Instances). Nutzen Sie Service-API-Operationen, wie z. B. die Amazon EC2 `DescribeInstances` API, um die aktuelle Nutzung von Ressourcen zu erfassen.
- Erfassen Sie Ihre aktuellen Kontingente, die für die Services wesentlich und anwendbar sind. Nutzen Sie dazu:
  - AWS Service Quotas
  - AWS Trusted Advisor
  - AWS-Dokumentation
  - Entsprechende Seiten von AWS-Services
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- Verwenden Sie AWS Service Quotas, ein AWS-Service, der Sie bei der Verwaltung von mehr als 250 AWS-Services an einem einzigen Ort unterstützt.
- Nutzen Sie Trusted Advisor-Service-Beschränkungen, um Ihre aktuellen Service-Beschränkungen zu verschiedenen Schwellenwerten zu überwachen.
- Nutzen Sie die Historie der Service-Kontingente (Konsole oder AWS CLI), um regionale Erhöhungen zu prüfen.

- Vergleichen Sie die Änderungen der Service-Kontingente in jeder Region und jedem Konto, um bei Bedarf auszugleichen.

Für die Verwaltung:

- Automatisiert: Richten Sie eine angepasste AWS Config-Regel ein, um Service-Kontingente in den Regionen zu prüfen und Abweichungen zu ermitteln.
- Automatisiert: Richten Sie eine geplante Lambda-Funktion ein, um Service-Kontingente in den Regionen zu scannen und Abweichungen zu ermitteln.
- Manuell: Scannen Sie Service-Kontingente über AWS CLI, die API oder die AWS-Konsole, um Service-Kontingente in den Regionen zu scannen und Abweichungen zu ermitteln. 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 Servicekontingenten und Einschränkungen
- REL01-BP02 Verwalten von Servicekontingenten 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 Segmentierung Ihres Workloads
- 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-BP05 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework’s Reliability Pillar: Availability (Säule für Zuverlässigkeit des AWS Well-Architected Framework)
- 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 for AWS (Kontingentüberwachung für AWS)
- AWS Fault Isolation Boundaries (AWS-Grenzen für die Fehlerisolierung)
- Availability with redundancy (Verfügbarkeit mit Redundanz)
- AWS für Daten
- What is Continuous Integration? (Was ist Continuous integration?)
- What is Continuous Delivery? (Was ist Continuous Delivery?)
- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Verwaltung des Kontolebenszyklus in SaaS-Umgebungen mit Konto pro Mandant auf AWS)
- Verwalten und Überwachen der API-Drosselung in Ihren Workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Umfangreiche AWS Trusted Advisor-Empfehlungen mit AWS Organizations anzeigen)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatisieren von Service-Limit-Erhöhungen und Enterprise Support mit 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 (Kontingente für AWS Services, die Service Quotas verwenden, anzeigen und verwalten)

- AWS IAM Quotas Demo (AWS IAM-Kontingente – Demo)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: Details und Strategien: Wie man die Kontrolle über große und kleine Systeme übernimmt)

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-BP05 Automatisieren der Kontingentverwaltung

Implementieren Sie Tools, um vor dem Erreichen von Schwellenwerten benachrichtigt zu werden. Durch die Verwendung von AWS Service Quotas-APIs können Sie Anfragen zur Kontingenterhöhung automatisieren.

Wenn Sie Ihre Konfigurationsmanagementdatenbank (CMDB) oder das Ticketing-System mit Service Quotas integrieren, können Sie die Verfolgung von Kontingenterhöhungsanfragen und von aktuellen Kontingenten automatisieren. Zusätzlich zum AWS SDK bietet Service Quotas Automatisierung unter Verwendung der AWS Command Line Interface (AWS CLI).

Gängige Antimuster:

- Die Kontingente und die Nutzung werden in Tabellen verfolgt.
- Es werden Berichte zur täglichen, wöchentlichen oder monatlichen Nutzung ausgeführt und anschließend wird die Nutzung mit den Kontingenten verglichen.

Vorteile der Einführung dieser bewährten Methode: Durch die automatisierte Nachverfolgung der AWS-Servicekontingente und die Überwachung ihrer Nutzung können Sie feststellen, wann ein Kontingent zu Neige geht. Sie können die Automatisierung einrichten, damit Sie beim Anfordern einer Kontingenterhöhung bei Bedarf unterstützt werden. Wenn sich Ihre Nutzung in die entgegengesetzte Richtung entwickelt, sollten Sie einige Kontingente reduzieren, um von den verringerten Risiken (im Falle von kompromittierten Anmeldeinformationen) und von Kosteneinsparungen zu profitieren.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Richten Sie eine automatisierte Überwachung ein. Implementieren Sie Tools mithilfe von SDKs, um vor dem Erreichen von Schwellenwerten benachrichtigt zu werden.
  - Nutzen Sie Service Quotas und erweitern Sie den Service mit einer Lösung zur automatisierten Kontingentüberwachung, z. B. mit AWS Limit Monitor oder einem Angebot aus AWS Marketplace.
    - Was ist Service Quotas?
    - Quota Monitor on AWS – AWS-Lösung
  - Richten Sie automatische Reaktionen anhand von Schwellenwerten für Kontingente mit Amazon SNS- und AWS Service Quotas-APIs ein.
  - Testen Sie die Automatisierung.
    - Konfigurieren Sie Limit-Schwellenwerte.
    - Integrieren Sie Änderungsereignisse von AWS Config-Bereitstellungspipelines, Amazon EventBridge oder Ereignisse von Drittanbietern.
    - Legen Sie unnatürlich niedrige Schwellenwerte für Kontingente fest, um die Reaktionen zu testen.
    - Richten Sie Trigger ein, damit bei Benachrichtigungen geeignete Maßnahmen ergriffen werden und bei Bedarf der AWS Support kontaktiert wird.
    - Lösen Sie Änderungsereignisse manuell aus.
    - Führen Sie eine Ernstfallübung aus, um den Prozess für die Kontingenterhöhung zu testen.

Ressourcen

Ähnliche Dokumente:

- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können

- AWS Marketplace: CMDB-Produkte zur Nachverfolgung von Limits
- AWS Service Quotas (früher als Service Limits bezeichnet)
- Bewährte AWS Trusted Advisor Trusted Advisor-Methoden (Prüfungen) (siehe Abschnitt „Servicelimits“)
- Quota Monitor on AWS – AWS-Lösung
- Amazon EC2 Service Limits
- Was ist Service Quotas?

Ähnliche Videos:

- AWS Live re:Inforce 2019 – Service Quotas

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

Wenn eine Ressource ausfällt oder nicht erreichbar ist, wird diese Ressource möglicherweise noch auf ein Kontingent angerechnet, bis sie erfolgreich beendet wird. 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, wenn diese bewährte Methode nicht eingeführt wird: 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.

Fehler wie in der obigen Liste können der Auslöser für ein Failover-Ereignis sein. 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 aktuelle 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 Servicekontingenten und Einschränkungen
- REL01-BP02 Verwalten von Servicekontingenten für mehrere Konten und Regionen
- REL01-BP03 Berücksichtigen von festen Servicekontingenten und Einschränkungen durch die Architektur

- REL01-BP04 Überwachen und Verwalten von Kontingenten
- REL01-BP05 Automatisieren der Kontingentverwaltung
- REL03-BP01 Segmentierung Ihres Workloads
- 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-BP05 Testen der Ausfallsicherheit mit Chaos-Engineering

Zugehörige Dokumente:

- AWS Well-Architected Framework’s Reliability Pillar: Availability (Säule für Zuverlässigkeit des AWS Well-Architected Framework)
- 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 for AWS (Kontingentüberwachung für AWS)
- AWS Fault Isolation Boundaries (AWS-Grenzen für die Fehlerisolierung)
- Availability with redundancy (Verfügbarkeit mit Redundanz)
- AWS für Daten
- What is Continuous Integration? (Was ist Continuous integration?)
- What is Continuous Delivery? (Was ist Continuous Delivery?)
- APN-Partner: Partner, die Sie bei der Konfigurationsverwaltung unterstützen können
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Verwaltung des Kontolebenszyklus in SaaS-Umgebungen mit Konto pro Mandant auf AWS)
- Verwalten und Überwachen der API-Drosselung in Ihren Workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Umfangreiche AWS Trusted Advisor-Empfehlungen mit AWS Organizations anzeigen)

- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatisieren von Service-Limit-Erhöhungen und Enterprise Support mit 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 (Kontingente für AWS Services, die Service Quotas verwenden, anzeigen und verwalten)
- AWS IAM Quotas Demo (AWS IAM-Kontingente – Demo)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: Details und Strategien: Wie man die Kontrolle über große und kleine Systeme übernimmt)

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 häufig in mehreren Umgebungen. Dazu gehören mehrere Cloud-Umgebungen (öffentlich zugängliche und private) und möglicherweise die vorhandene Infrastruktur Ihres Rechenzentrums. Die Pläne müssen Netzwerkaspekte umfassen, wie z. B. Konnektivität innerhalb und zwischen Systemen, Verwaltung öffentlicher und privater IP-Adressen und Auflösung von Domänennamen.

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 Abschnitt der AWS Cloud bereitstellen, in dem Sie AWS-Ressourcen in einem virtuellen Netzwerk starten können.

Bewährte Methoden

- 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 Nabe-und-Speiche-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

Domänennamen, 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 Domänennamen 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 Domänennamen 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 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.

Um Kunden vor DDoS-Angriffen zu schützen, bietet AWS AWS Shield Standard. 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 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. Richten Sie ein hochverfügbares DNS ein: Amazon Route 53 ist ein hochverfügbarer und skalierbarer Domain Name System (DNS)-Webservice. 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 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“.
   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-Trefferverhältnis)
      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 (Funktionsweise von AWS WAF). Wenn Sie bereit sind, den Schutz vor HTTP-POST- und -GET-Floods auf der Anwendungsebene zu implementieren, lesen Sie Getting started with AWS WAF (Erste Schritte mit AWS WAF). Sie können außerdem AWS WAF mit CloudFront verwenden. In der Dokumentation erfahren Sie, wie 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 (Erste Schritte mit AWS Shield Advanced).

Ressourcen

Zugehörige bewährte Methoden:

- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL10-BP02 Auswählen der geeigneten Standorte für Ihre Multi-Standort-Bereitstellung

- 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 (Funktionalität zur Netzwerkkonnektivität – Etablieren Ihrer Cloud-Grundlagen)
- Was ist Amazon API Gateway?
- What are AWS WAF, AWS Shield, and AWS Firewall Manager? (Was sind AWS WAF, AWS Shield und AWS Firewall Manager?)
- Was ist Amazon Route 53 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 2022 – Verbessern der Leistung und Verfügbarkeit mit AWS Global Accelerator)
- AWS re:Invent 2020: Global traffic management with Amazon Route 53 (AWS re:Invent 2020: Globales Datenverkehrsmanagement mit AWS)
- AWS re:Invent 2022 – Operating highly available Multi-AZ applications (AWS re:Invent 2022 – Betrieb hochverfügbarer Multi-AZ Anwendungen)
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure (AWS re:Invent 2022 – Details zur AWS-Netzwerkinfrastruktur)
- AWS re:Invent 2022 – Building resilient networks (AWS re:Invent 2022 – Aufbau widerstandsfähiger Netzwerke)

Zugehörige Beispiele:

- Disaster Recovery with Amazon Route 53 Application Recovery Controller (ARC) (Notfallwiederherstellung mit Amazon Route 53 Application Recovery Controller (ARC))
- Workshops zur Zuverlässigkeit
- AWS Global Accelerator-Workshop

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

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, stellen Sie in mindestens zwei davon Redundanz sicher. Erwägen Sie gegebenenfalls den Einsatz von AWS Marketplace-Appliances als Endpunkte von VPNs. Stellen Sie bei Verwendung von AWS Marketplace-Appliances redundante Instances bereit, um eine hohe Verfügbarkeit in verschiedenen Availability Zones zu gewährleisten.

Mit dem Cloud-Service AWS Direct Connect ist es einfach, eine dedizierte Netzwerkverbindung zwischen Ihrer On-Premises-Umgebung und AWS herzustellen. Mit Direct Connect Gateway kann Ihr On-Premises-Rechenzentrum mit mehreren AWS-VPCs verbunden werden, die über mehrere AWS-Regionen verteilt sind.

Diese Redundanz behebt mögliche Ausfälle, die sich auf die Ausfallsicherheit der Konnektivität auswirken:

- Wie können Sie sich gegen Fehler in Ihrer Topologie wappnen?
- Was passiert, wenn Sie etwas falsch konfigurieren oder die Konnektivität entfernen?
- Sind Sie in der Lage, eine unerwartete Erhöhung des Datenverkehrs bzw. der Nutzung Ihrer Services aufzufangen?
- Sind Sie in der Lage, den Versuch eines Distributed Denial of Service (DDoS)-Angriffs abzuwehren?

Berücksichtigen Sie bei der Verbindung Ihrer VPC mit Ihrem On-Premise-Rechenzentrum über VPN auch die Ausfallsicherheits- und Bandbreitenanforderungen, die Sie benötigen, wenn Sie den Anbieter und die Instance-Größe für die Ausführung der Appliance auswählen. Bei der Auswahl

einer VPN-Appliance, die in ihrer Implementierung keine Ausfallsicherheit bietet, sollten Sie eine redundante Verbindung über eine zweite Appliance aufbauen. Bei all diesen Szenarios müssen Sie eine akzeptable Wiederherstellungszeit definieren und testen, um sicherzustellen, dass Sie diese Anforderungen erfüllen können.

Wenn Sie Ihre VPC über eine Direct-Connect-Verbindung mit Ihrem Rechenzentrum verbinden und diese Verbindung hochverfügbar sein muss, benötigen Sie redundante Direct-Connect-Verbindungen mit jedem Rechenzentrum. Die redundante Verbindung sollte eine zweite Direct-Connect-Verbindung von einem anderen Standort als der ersten verwenden. Wenn Sie mehrere Rechenzentren betreiben, stellen Sie sicher, dass Ihre Verbindungen an unterschiedlichen Orten enden. Verwenden Sie das Direct Connect Resiliency Toolkit, um dies einzurichten.

Wenn Sie sich für ein internetbasiertes Failover auf ein VPN mit einem AWS VPN entscheiden, ist es wichtig zu verstehen, dass es einen Datendurchsatz von bis zu 1,25 Gbit/s pro VPN-Tunnel bietet, dass Equal Cost Multi Path (ECMP) für ausgehenden Datenverkehr jedoch nicht unterstützt wird, wenn mehrere von AWS verwaltete VPN-Tunnel auf demselben VGW enden. Wir raten davon ab, AWS Managed VPN als Sicherung für Direct-Connect-Verbindungen zu verwenden, es sei denn, Geschwindigkeiten von weniger als 1 Gbit/s während des Failovers stellen für Sie kein Problem dar.

Sie können VPC-Endpunkte auch verwenden, um Ihre VPC privat mit unterstützten AWS-Services und VPC-Endpunktservices zu verbinden, powered by AWS PrivateLink, ohne das öffentliche Internet zu durchlaufen. Endpunkte sind virtuelle Geräte. Sie sind horizontal skalierte, redundante und hochverfügbare VPC-Komponenten. Sie ermöglichen die Kommunikation zwischen Instances in Ihrer VPC und Ihren Services, ohne dass es zu Verfügbarkeitsrisiken oder Bandbreitenbeschränkungen für Ihren Netzwerkdatenverkehr kommt.

Gängige Antimuster:

- Einsatz nur eines Konnektivitätsanbieters zwischen dem lokalen Netzwerk und AWS.
- Die Konnektivitätsfunktionen der AWS Direct Connect-Verbindung werden genutzt, es gibt aber nur eine Verbindung.
- Es gibt nur einen Pfad für die VPN-Konnektivität.

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

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Stellen Sie sicher, dass eine hochverfügbare Konnektivität zwischen AWS und der On-Premises-Umgebung vorhanden ist. 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, stellen Sie in mindestens zwei davon Redundanz sicher. Erwägen Sie gegebenenfalls den Einsatz von AWS Marketplace-Appliances als Endpunkte von VPNs. Stellen Sie bei Verwendung von AWS Marketplace-Appliances redundante Instances bereit, um eine hohe Verfügbarkeit in verschiedenen Availability Zones zu gewährleisten.
  - Stellen Sie sicher, dass eine redundante Verbindung zu Ihrer On-Premises-Umgebung besteht. Möglicherweise benötigen Sie redundante Verbindungen zu mehreren AWS-Regionen, um Ihre Verfügbarkeitsanforderungen zu erfüllen.
    - AWS Direct Connect Resiliency Recommendations (AWS Direct Connect-Resilienzempfehlungen)
    - Verwenden redundanter Site-to-Site-VPN-Verbindungen für Failover
      - Ermitteln Sie über die Service-API die ordnungsgemäße Nutzung von Direct-Connect-Verbindungen.
        - DescribeConnections
        - DescribeConnectionsOnInterconnect
        - DescribeDirectConnectGatewayAssociations
        - DescribeDirectConnectGatewayAttachments
        - DescribeDirectConnectGateways
        - DescribeHostedConnections
        - DescribeInterconnects
      - Wenn nur eine oder gar keine Direct-Connect-Verbindung besteht, richten Sie redundante VPN-Tunnel zu Ihren Virtual Private Gateways ein.
        - Was ist AWS-Site-to-Site VPN?
  - Erfassen Sie die aktuelle Konnektivität (z. B. Direct Connect, Virtual Private Gateways, AWS Marketplace-Appliances).
    - Ermitteln Sie über die Service-API die Konfiguration von Direct-Connect-Verbindungen.
      - DescribeConnections

    - DescribeConnectionsOnInterconnect
    - DescribeDirectConnectGatewayAssociations
    - DescribeDirectConnectGatewayAttachments
    - DescribeDirectConnectGateways
    - DescribeHostedConnections
    - DescribeInterconnects
  - Erfassen Sie über die Service API die von Routing-Tabellen genutzten Virtual Private Gateways.
    - DescribeVpnGateways
    - DescribeRouteTables
  - Erfassen Sie über die Service-API die von Routing-Tabellen genutzten AWS Marketplace-Anwendungen.
    - DescribeRouteTables

Ressourcen

Relevante Dokumente:

- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Direct Connect Resiliency Recommendations (AWS Direct Connect-Resilienzempfehlungen)
- AWS Marketplace für Netzwerkinfrastruktur
- Amazon Virtual Private Cloud-Konnektivitätsoptionen – Whitepaper
- Hochverfügbare Netzwerkkonnektivität zwischen mehreren Rechenzentren
- Verwenden redundanter Site-to-Site-VPN-Verbindungen für Failover
- Erste Schritte mit Direct Connect Resiliency Toolkit
- VPC-Endpunkte und VPC-Endpunktservices (AWS PrivateLink)
- Was ist Amazon VPC?
- Was ist ein Transit-Gateway?
- Was ist AWS-Site-to-Site VPN?
- Arbeiten mit Direct-Connect-Gateways

Relevante Videos:

- AWS re:Invent 2018: Erweitertes VPC-Design und neue Funktionen für Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway-Referenzarchitekturen für verschiedene VPCs (NET406-R1)

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 einer Workload 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, die sich auf mehrere Availability Zones erstrecken.
- 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 EC2-Instances zu erfüllen, die Sie möglicherweise verwenden, z. B. Spot-Flotten für Machine Learning, Amazon EMR-Cluster oder Amazon Redshift-Cluster.
- 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.
- Sie sollten die Bereitstellung großer VPC CIDR-Blöcke planen. 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. Bedenken Sie, dass die Bereitstellung der größtmöglichen VPC (/16) mehr als 65 000 IP-Adressen zur Folge hat. Allein im IP-Adressbereich 10.x.x.x könnten Sie 255 solcher VPCs bereitstellen. Sie sollten daher eher auf eine zu große als eine zu kleine Lösung setzen, um die Verwaltung Ihrer VPCs zu vereinfachen.

Gängige Antimuster:

- Es werden kleine VPCs erstellt.
- Es werden kleine Subnetze erstellt und anschließend müssen beim Wachstum Subnetze zu Konfigurationen hinzugefügt werden.
- 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.

Vorteile der Einführung 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.
    - Legen Sie VPCs so groß wie möglich an. Der VPC-CIDR-Block, der anfänglich Ihrer VPC zugewiesen war, kann nicht geändert oder gelöscht werden. Sie können der VPC jedoch zusätzliche nicht überlappende CIDR-Blöcke hinzufügen. Dies kann jedoch zu einer Fragmentierung der Adressbereiche führen.

- Legen Sie VPCs so groß wie möglich an. Der VPC-CIDR-Block, der anfänglich Ihrer VPC zugewiesen war, kann nicht geändert oder gelöscht werden. Sie können der VPC jedoch zusätzliche nicht überlappende CIDR-Blöcke hinzufügen. Dies kann jedoch zu einer Fragmentierung der Adressbereiche führen.

Ressourcen

Relevante Dokumente:

- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Marketplace für Netzwerkinfrastruktur
- Amazon Virtual Private Cloud-Konnektivitätsoptionen – Whitepaper
- Hochverfügbare Netzwerkkonnektivität zwischen mehreren Rechenzentren
- Multi-VPC-Konnektivität in einer Region
- Was ist Amazon VPC?

Relevante Videos:

- AWS re:Invent 2018: Erweitertes VPC-Design und neue Funktionen für Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway-Referenzarchitekturen für verschiedene VPCs (NET406-R1)

REL02-BP04 Vorziehen von Nabe-und-Speiche-Topologien gegenüber M-zu-N-Netzen

Wenn mehr als zwei Netzwerkadressbereiche (z. B. VPCs und On-Premises-Netzwerke) über VPC-Peering, AWS Direct Connect oder VPN verbunden sind, verwenden Sie ein Nabe-und-Speiche-Modell, wie es von AWS Transit Gateway bereitgestellt wird.

Wenn Sie nur zwei solche Netzwerke haben, können Sie sie einfach miteinander verbinden, doch wenn die Anzahl der Netzwerke zunimmt, ist die Komplexität derart vernetzter Verbindungen nicht mehr tragbar. AWS Transit Gateway bietet ein einfach zu wartendes Nabe-zu-Speiche-Modell, das die Weiterleitung des Datenverkehrs über mehrere Netzwerke ermöglicht.

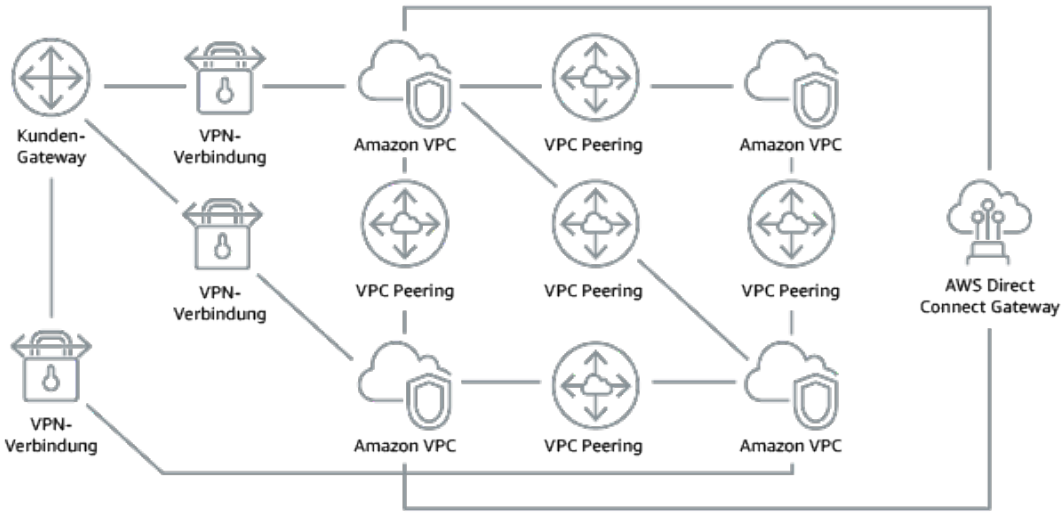


Abbildung 1: Ohne AWS Transit Gateway: Sie müssen jede Amazon VPC über eine VPN-Verbindung miteinander und mit jedem Standort verbinden. Bei der Skalierung kann dies sehr komplex werden.

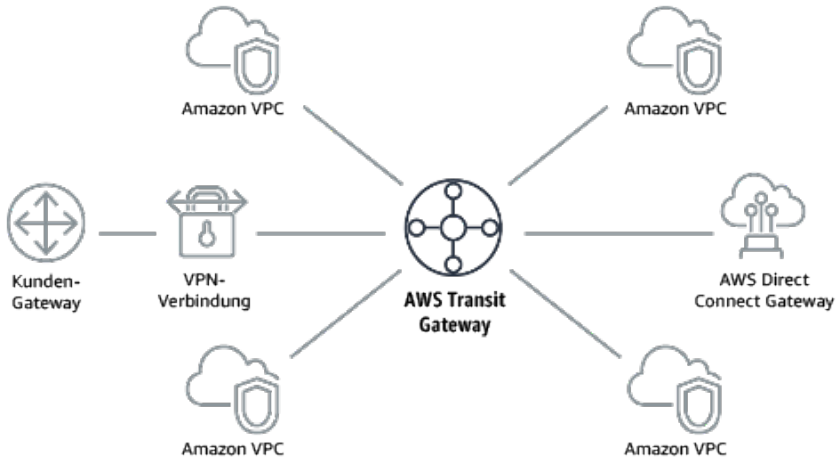


Abbildung 2: Mit AWS Transit Gateway: Sie verbinden einfach jede Amazon VPC oder jedes VPN mit dem AWS Transit Gateway und leiten den Datenverkehr zu und von jeder VPC oder VPN weiter.

Gängige Antimuster:

- Verbinden von mehr als zwei VPCs mit VPC-Peering.
- Es werden mehrere BGP-Sitzungen für jede VPC eingerichtet, um Konnektivität für mehrere Virtual Private Clouds (VPCs) in mehreren AWS-Regionen herzustellen.

Vorteile der Einführung dieser bewährten Methode: Mit der zunehmenden Anzahl der Netzwerke wird die Komplexität solcher verflochtenen Verbindungen immer größer. AWS Transit Gateway bietet ein einfach zu wartendes Nabe-und-Speiche-Modell, das die Weiterleitung des Datenverkehrs über mehrere Netzwerke ermöglicht.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Ziehen Sie Nabe-und-Speiche-Topologien gegenüber M-zu-N-Netzen vor. Wenn mehr als zwei Netzwerkadressbereiche (VPCs, On-Premises-Netzwerke) über VPC-Peering, AWS Direct Connect oder VPN verbunden sind, verwenden Sie ein Nabe-und-Speiche-Modell, wie es von AWS Transit Gateway bereitgestellt wird.
  - Bei nur zwei derartigen Netzwerken können Sie sie einfach miteinander verbinden, doch mit der zunehmenden Anzahl der Netzwerke wird die Komplexität solcher verflochtenen Verbindungen immer größer. AWS Transit Gateway bietet ein einfach zu wartendes Nabe-und-Speiche-Modell, das die Weiterleitung des Datenverkehrs über mehrere Netzwerke ermöglicht.
    - Was ist ein Transit-Gateway?

Ressourcen

Relevante Dokumente:

- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Marketplace für Netzwerkinfrastruktur
- Hochverfügbare Netzwerkkonnektivität zwischen mehreren Rechenzentren
- VPC-Endpunkte und VPC-Endpunktservices (AWS PrivateLink)
- Was ist Amazon VPC?
- Was ist ein Transit-Gateway?

Relevante Videos:

- AWS re:Invent 2018: Erweitertes VPC-Design und neue Funktionen für Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway-Referenzarchitekturen für verschiedene VPCs (NET406-R1)

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 VPN verbunden sind. Ebenso müssen Sie IP-Adresskonflikte zwischen einer VPC und lokalen Umgebungen oder anderen verwendeten Cloud-Anbietern vermeiden. Sie müssen bei Bedarf auch die Möglichkeit haben, private IP-Adressbereiche zuzuweisen.

Ein IP-Adressenverwaltungssystem (IPAM) kann dabei helfen. Im AWS Marketplace stehen mehrere IPAMs zur Verfügung.

Gängige Antimuster:

- Verwenden Sie denselben IP-Bereich in Ihrer VPC wie im lokalen Netzwerk oder in Ihrem Unternehmensnetzwerk.
- Keine Verfolgung von IP-Bereichen von VPCs, die zur Bereitstellung der Workloads verwendet werden.

Vorteile der Einführung 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

- Überwachen und verwalten Sie die CIDR-Nutzung. Bewerten Sie die potenzielle Nutzung in AWS, fügen Sie vorhandenen VPCs CIDR-Bereiche hinzu und erstellen Sie neue VPCs, um das geplante Wachstum abzudecken.
  - Ermitteln Sie den aktuellen CIDR-Umfang (z. B. VPCs, Subnetze).
    - Erfassen Sie über die Service-API den aktuellen CIDR-Umfang.
  - Erfassen Sie die aktuelle Subnetzauslastung.
    - Ermitteln Sie über die Service-API die in jeder Region pro VPC vorhandenen Subnetze.
      - DescribeSubnets

- 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 NAT-Appliances (Network and Port Translation) aus AWS Marketplace verwenden, wenn Sie die sich überschneidenden Bereiche verbinden müssen.

Ressourcen

Ähnliche Dokumente:

- APN-Partner: Partner, die Sie bei der Planung Ihres Netzwerks unterstützen können
- AWS Marketplace für Netzwerkinfrastruktur
- Amazon Virtual Private Cloud-Konnektivitätsoptionen – Whitepaper
- Hochverfügbare Netzwerkkonnektivität zwischen mehreren Rechenzentren
- Was ist Amazon VPC?
- Was ist IPAM?

Ähnliche Videos:

- AWS re:Invent 2018: Erweitertes VPC-Design und neue Funktionen für Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway-Referenzarchitekturen für verschiedene VPCs (NET406-R1)

Workload-Architektur

Ausgangspunkt für einen zuverlässigen 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 verwendet, 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 Segmentierung Ihres Workloads
- REL03-BP02 Entwickeln von Services, die sich auf bestimmte Geschäftsdomänen und Funktionen konzentrieren
- REL03-BP03 Bereitstellen von Serviceverträgen pro API

REL03-BP01 Segmentierung Ihres Workloads

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. Abhängig von den Anforderungen Ihrer Anwendung könnte es sich im Endergebnis um eine Kombination aus einer serviceorientierten Architektur (SOA) und Microservices handeln, wenn dies möglich ist. 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 Einrichtung dieser Best Practice:

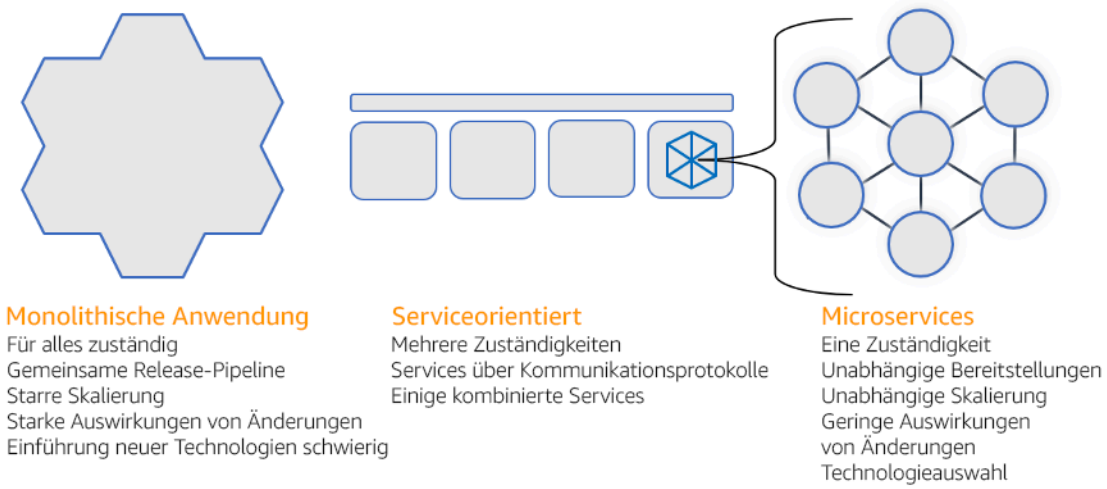
- Spezifischere Segmente führen zu einer größeren Agilität, zu organisatorischer Flexibilität und zu Skalierbarkeit.
- Die Auswirkungen von Service-Unterbrechungen werden reduziert.
- Die einzelnen Komponenten einer Anwendung besitzen möglicherweise unterschiedliche Anforderungen an die Verfügbarkeit,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

Risikostufe bei fehlender Befolgung dieser Best Practice: Hoch

Implementierungsleitfaden

Wählen Sie Ihren Architekturtyp basierend auf der Segmentierung Ihres Workloads aus. Wählen Sie eine serviceorientierte Architektur (SOA) oder eine Microservices-Architektur aus. (In seltenen Fällen ist möglicherweise auch eine monolithische Architektur geeignet.) Auch wenn Sie mit einer monolithischen Architektur beginnen möchten, müssen Sie sicherstellen, dass diese modular ist und zu einer SOA oder zu Microservices weiterentwickeln werden kann, wenn Ihr Produkt aufgrund der zunehmenden Einführung durch Benutzer skaliert wird. SOA und Microservices ermöglichen eine kleinteiligere Segmentierung, die als moderne skalierbare und zuverlässige Architektur bevorzugt wird. Es gibt jedoch auch Nachteile, die besonders bei der Bereitstellung einer Microservice-Architektur berücksichtigt werden sollten.

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. SOA und Microservices bieten eine jeweils kleinere Segmentierung, die als moderne skalierbare und zuverlässige Architektur bevorzugt wird. SOA kann ein guter Kompromiss für das Erreichen einer kleineren Segmentierung sein, während die Komplexität von Microservices zum Teil vermieden wird. Weitere Informationen finden Sie in Kompromisse bei Microservices.
- 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 in Implementieren von Microservices in 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 in Nahtlose Integration ältere On-Premises-Workloads unter Anwendung eines Strangler-Fig-Musters.
- Die Implementierung von Microservices erfordert möglicherweise einen Mechanismus für die Entdeckung von Services, damit diese verteilten Services miteinander kommunizieren können. AWS App Mesh kann mit serviceorientierten Architekturen verwendet werden, um eine zuverlässige Erkennung von Services und den Zugriff auf sie zu unterstützen. AWS Cloud Map kann für die dynamische, DNS-basierte Serviceerkennung verwendet werden.

- Wenn Sie von einem Monolithen zur SOA migrieren, kann Amazon MQ helfen, als Service-Bus die Lücke zu überbrücken, wenn Sie ältere Anwendungen in der Cloud neu entwerfen.
- 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 Faktorwechsel-Prozesses sollten Sie entscheiden, ob Sie eine relationale oder eine nicht relationale (NoSQL) Datenbank verwenden. Weitere Informationen finden Sie in Von SQL zu NoSQL.

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: Konfigurieren einer REST-API mit OpenAPI
- Was ist eine serviceorientierte Architektur?
- Bounded Context (Begrenzter Kontext) (ein zentrales Muster im domänengesteuerten Design)
- Implementieren von Microservices in AWS
- Kompromisse bei Microservices
- Microservices – eine Definition dieses neuen Architekturbegriffs
- Microservices in AWS
- Was ist AWS App Mesh?

Zugehörige Beispiele:

- Workshop für die iterative App-Modernisierung

Zugehörige Videos:

- Kompetenz mit Microservices in 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 Domänenmodelle 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 Monolithe 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.

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

Typische Anti-Muster:

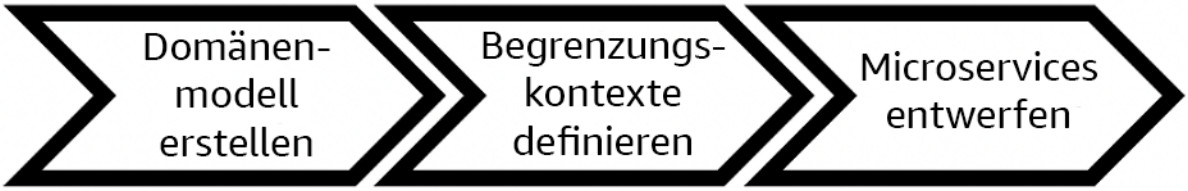
- Teams werden für bestimmte technische Bereiche wie UI und UX, Middleware oder Datenbank gebildet, anstatt für bestimmte Geschäftsdomänen.
- Anwendungen erstrecken sich über die Zuständigkeiten der einzelnen Bereiche. Services, die sich über begrenzte Kontexte erstrecken, können schwieriger zu verwalten sein, erfordern einen größeren Testaufwand und erfordern die Teilnahme mehrerer Domänenteams an Softwareupdates.
- Domänenabhängigkeiten wie Domain-Entity-Bibliotheken werden von allen Services gemeinsam genutzt, sodass Änderungen für eine Servicedomäne Änderungen an anderen Service-Domains erfordern.
- Serviceverträge und Geschäftslogik formulieren Entities nicht in einer gemeinsamen und konsistenten Domänensprache, 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 domänenspezifischen Resilienzanforderungen für die implementierte Domäne.

Risikostufe bei fehlender Befolgung dieser Best Practice: Hoch

Implementierungsleitfaden

Domain-driven Decision (DDD, Domänengesteuerte Entscheidung) 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.



Domänengesteuerte Entscheidung

Implementierungsschritte

- Teams können Event-Storming-Workshops veranstalten, um rasch Ereignisse, Befehle, Mengen und Domänen in einem unkomplizierten Notizformat zu sammeln.
- Sobald Domain-Entities und -Funktionen in einem Domänenkontext gebildet wurden, können Sie Ihre Domäne mithilfe eines begrenzten Kontextsweiter in kleinere Modelle unterteilt, wobei Entities 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 Entities 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.

- Zerlegung von Monolithen in Microservices skizziert Muster für das Refactoring von Microservices. Die Verwendung von Mustern für die Unterteilung nach Geschäftsfähigkeit, Subdomäne oder Transaktion passt gut zu domänengesteuerten 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 Korruptionsbekämpfungerstellt, die das neu definierte Domänenmodell vor äußeren Einflüssen schützt.

Nachdem die Teams eine Domänenanalyse durchgeführt und Entities und Serviceverträge definiert haben, können sie AWS-Services nutzen, um ihr domänengesteuertes Design als Cloud-basierte Services zu implementieren.

- Beginnen Sie Ihre Entwicklung, indem Sie Tests definieren, die die Geschäftsregeln Ihrer Domäne 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 die AWS-Services, die den Anforderungen Ihrer Geschäftsdomänen und Ihrer Microservice-Architekturam besten entsprechen:
  - AWS Serverless ermöglicht es Ihrem Team, sich auf eine bestimmte Domänenlogik zu konzentrieren, anstatt Server und Infrastruktur zu verwalten.
  - Container in AWS vereinfachen die Verwaltung Ihrer Infrastruktur, sodass Sie sich auf Ihre Domänenanforderungen konzentrieren können.
  - Speziell entwickelte Datenbanken helfen Ihnen dabei, Ihre Domänenanforderungen dem am besten geeigneten Datenbanktyp zuzuordnen.
- Hexagonale Architekturen auf AWS skizzieren ein Framework zur Integration von Geschäftslogik in Services. Dabei wird rückwärts von der Geschäftsdomäne 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 Domäne zu konzentrieren und die Softwarequalität zu verbessern.

Ressourcen

Zugehörige bewährte Methoden:

- REL03-BP01 Segmentierung Ihres Workloads
- REL03-BP03 Bereitstellen von Serviceverträgen pro API

Zugehörige Dokumente:

- AWS Microservices
- Implementieren von Microservices in AWS
- How to break a Monolith into Microservices (Aufschlüsseln eines Monolithen in Microservices)
- Getting Started with DDD when Surrounded by Legacy Systems (Erste Schritte mit DDD, wenn die Umgebung aus Legacy-Systemen besteht)
- Domain-Driven Design: Tackling Complexity in the Heart of Software (Domänengesteuertes Design: Umgang mit der Komplexität im Herzen der Software)
- Hexagonale Architekturen auf AWS
- Zerlegung von Monolithen in Microservices
- Event Storming
- Nachrichten zwischen begrenzten Kontexten
- Microservices
- Testgetriebene Entwicklung
- Verhaltensgetriebene Entwicklung

Zugehörige Beispiele:

- Workshop „Enterprise Cloud Native“
- Designing Cloud Native Microservices on AWS (from DDD/EventStormingWorkshop) (Entwerfen Cloud-nativer Microservices in AWS (aus DDD/EventStormingWorkshop))

Zugehörige Tools:

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

REL03-BP03 Bereitstellen von Serviceverträgen pro API

Serviceverträge sind dokumentierte Vereinbarungen zwischen API-Herstellern und Verbrauchern, die in einer maschinenlesbaren API-Definition festgehalten sind. Eine Vertragsversionsverwaltungsstrategie ermöglicht es Verbrauchern, die vorhandene API weiter zu verwenden und ihre Anwendungen auf eine neuere API zu migrieren, wenn sie bereit sind. Die Bereitstellung durch den Produzenten kann jederzeit erfolgen, solange der Vertrag eingehalten wird. Die Serviceteams können den Technologie-Stack ihrer Wahl verwenden, um den API-Vertrag zu erfüllen.

Gewünschtes Ergebnis:

Typische Anti-Muster: Anwendungen, die mit serviceorientierten Architekturen oder Microservice-Architekturen erstellt wurden, können unabhängig voneinander arbeiten und verfügen gleichzeitig über eine integrierte Laufzeitabhängigkeit. Änderungen, die für einen API-Verbraucher oder -Hersteller bereitgestellt werden, beeinträchtigen die Stabilität des Gesamtsystems nicht, wenn beide Seiten einen gemeinsamen API-Vertrag einhalten. Komponenten, die über Service-APIs kommunizieren, können unabhängige funktionale Releases, Upgrades von Laufzeitabhängigkeiten oder ein Failover auf eine Notfallwiederherstellung (DR) ausführen, ohne dass sich dies gegenseitig beeinträchtigt. Darüber hinaus können spezialisierte Services unabhängig voneinander skaliert werden und können dabei den Ressourcenbedarf absorbieren, ohne dass andere Services ebenfalls skaliert werden müssen.

- Erstellung von Service-APIs ohne stark typisierte Schemata. Dies führt zu APIs, die nicht zum Generieren von API-Bindungen und Payloads verwendet werden können, die nicht programmgesteuert validiert werden können.
- Keine Versionsverwaltungsstrategie, weshalb API-Verbraucher dazu gezwungen sind, Updates zu installieren, Releases einzuspielen oder eine Notfallwiederherstellung durchzuführen, wenn sich Serviceverträge weiterentwickeln.
- Fehlermeldungen, die Details der zugrundeliegenden Service-Implementierung preisgeben, anstatt Integrationsfehler im Kontext und in der Sprache der Domäne zu beschreiben.
- Keine Verwendung von API-Verträgen zur Entwicklung von Testfällen und zur Simulation von API-Implementierungen, um unabhängige Tests von Servicekomponenten zu ermöglichen.

Vorteile der Nutzung dieser bewährten Methode: Verteilte Systeme, die aus Komponenten bestehen, die über API-Serviceverträge kommunizieren, können die Zuverlässigkeit verbessern. Entwickler können potenzielle Probleme schon früh im Entwicklungsprozess erkennen, indem sie während

der Kompilierung eine Typprüfung durchführen, um sicherzustellen, dass Anfragen und Antworten dem API-Vertrag entsprechen und die erforderlichen Felder vorhanden sind. API-Verträge bieten eine übersichtliche, selbstdokumentierende Schnittstelle für APIs und sorgen für eine bessere Interoperabilität zwischen verschiedenen Systemen und Programmiersprachen.

Risikostufe bei fehlender Befolgung dieser Best Practice: Mittel

Implementierungsleitfaden

Sobald Sie Geschäftsbereiche identifiziert und Ihre Workload-Segmentierung festgelegt haben, können Sie Ihre Service-APIs entwickeln. Definieren Sie zunächst maschinenlesbare Serviceverträge für APIs und implementieren Sie dann eine Strategie zur API-Versionsverwaltung. Wenn Sie bereit sind, Services über gängige Protokolle wie REST, GraphQL oder asynchrone Ereignisse zu implementieren, können Sie AWS-Services in Ihre Architektur einbinden, um Ihre Komponenten mit stark typisierten API-Verträgen zu integrieren.

AWS-Services für API-Serviceverträge

Implementieren Sie AWS-Services wie Amazon API Gateway, AWS AppSyncund Amazon EventBridge in Ihre Architektur, um API-Serviceverträge in Ihrer Anwendung zu verwenden. Amazon API Gateway hilft Ihnen bei der direkten Integration in native AWS-Services und andere Webservices. API Gateway unterstützt die OpenAPI-Spezifikation sowie die Versionsverwaltung. AWS AppSync ist ein verwalteter GraphQL -Endpunkt, den Sie konfigurieren, indem Sie ein GraphQL-Schema definieren, um eine Serviceschnittstelle für Abfragen, Mutationen und Abonnements festzulegen. Amazon EventBridge verwendet Ereignisschemata, um Ereignisse zu definieren und Codebindungen für Ihre Ereignisse zu generieren.

Implementierungsschritte

- Definieren Sie zunächst einen Vertrag für Ihre API. In einem Vertrag werden die Funktionen einer API festgehalten und stark typisierte Datenobjekte und Felder für die API-Eingabe und -Ausgabe definiert.
- Wenn Sie APIs in API Gateway konfigurieren, können Sie OpenAPI-Spezifikationen für Ihre Endpunkte importieren und exportieren.
  - Eine OpenAPI-Definition zu importieren, vereinfacht die Erstellung Ihrer API und kann in AWS-Infrastrukturen wie AWS Serverless Application Model und AWS Cloud Development Kit (AWS CDK) integriert werden.
  - Eine API-Definition zu exportieren, vereinfacht die Integration in API-Testtools und bietet Servicekunden eine Integrationsspezifikation.

- Definieren und verwalten Sie GraphQL-APIs mit AWS AppSync, indem Sie eine GraphQL-Schema-Datei definieren, um Ihre Vertragsschnittstelle zu generieren und die Interaktion mit komplexen REST-Modellen, mehreren Datenbanktabellen oder Legacy-Services zu vereinfachen.
- AWS Amplify -Projekte, die in AWS AppSync integriert sind, generieren stark typisierte JavaScript-Abfragedateien, die Sie sowohl in Ihrer Anwendung als auch in einer AWS AppSync-GraphQL-Client-Bibliothek für Amazon DynamoDB -Tabellen verwenden können.
- Wenn Sie Serviceereignisse aus Amazon EventBridge verarbeiten, befolgen diese Ereignisse Schemata, die bereits in der Schemaregistrierung existieren oder die Sie mit der OpenAPI-Spezifikation definieren. Mit einem in der Registrierung definierten Schema können Sie auch Client-Bindungen aus dem Schemavertrag generieren, um Ihren Code in Ereignisse zu integrieren.
- API erweitern oder versionieren Die Erweiterung einer API ist eine einfachere Option, wenn Felder hinzugefügt werden, die mit optionalen Feldern oder Standardwerten für Pflichtfelder konfiguriert werden können.
  - JSON-basierte Verträge für Protokolle wie REST und GraphQL können sich gut für eine Vertragserweiterung eignen.
  - XML-basierte Verträge für Protokolle wie SOAP sollten mit Service-Verbrauchern getestet werden, um festzustellen, ob eine Vertragserweiterung durchführbar ist.
- Erwägen Sie bei der Versionsverwaltung einer API die Implementierung einer Proxy-Versionsverwaltung, 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 Anfrage- und von Antwortzuordnungen nutzen, um Vertragsänderungen einfacher zu übernehmen. Hierzu wird eine Fassade eingerichtet, die Standardwerte für neue Felder bereitstellt oder entfernte Felder aus einer Anfrage oder Antwort herausnimmt. Mit diesem Ansatz kann der zugrunde liegende Service mit einer einzelnen Codebasis betrieben werden.

Ressourcen

Zugehörige bewährte Methoden:

- REL03-BP01 Segmentierung Ihres Workloads
- REL03-BP02 Entwickeln von Services, die sich auf bestimmte Geschäftsdomänen und Funktionen konzentrieren
- REL04-BP02 Implementieren lose gekoppelter Abhängigkeiten
- REL05-BP03 Steuern und Einschränken von Wiederholungsaufrufen

- REL05-BP05 Festlegen von Client-Zeitüberschreitungen

Zugehörige Dokumente:

- Was ist eine API (Anwendungsprogrammierschnittstelle)?
- Implementieren von Microservices in AWS
- Kompromisse bei Microservices
- Microservices – eine Definition dieses neuen Architekturbegriffs
- Microservices in AWS
- Arbeiten mit API Gateway-Erweiterungen für OpenAPI
- OpenAPI-Spezifikation
- GraphQL: Schemata und Typen
- Amazon EventBridge-Codebindungen

Zugehörige Beispiele:

- Amazon API Gateway: Konfigurieren einer REST-API mit OpenAPI
- Amazon API Gateway zu Amazon DynamoDB CRUD-Anwendung mit OpenAPI
- Moderne Anwendungsintegrationsmuster in einem serverlosen Zeitalter: API Gateway-Serviceintegration
- Implementieren einer Header-basierten API Gateway-Versionsverwaltung mit Amazon CloudFront
- AWS AppSync: Erstellen einer Client-Anwendung

Zugehörige Videos:

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

Zugehörige Tools:

- Amazon API Gateway
- AWS AppSync
- Amazon EventBridge

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

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. Komponenten des verteilten Systems müssen so funktionieren, dass sie keine negativen Auswirkungen auf andere Komponenten oder die Workload haben. Diese bewährten Methoden verhindern Ausfälle und verbessern die mittlere Zeit zwischen Ausfällen (MTBF).

Bewährte Methoden

- REL04-BP01 Bestimmen, welches verteilte System erforderlich ist
- REL04-BP02 Implementieren lose gekoppelter Abhängigkeiten
- REL04-BP03 Konstante Ausführung
- REL04-BP04 Festlegen aller Reaktionen als idempotent

REL04-BP01 Bestimmen, welches verteilte System erforderlich ist

Harte verteilte Echtzeitsysteme erfordern synchrone und schnelle Antworten, während bei weichen Echtzeitsystemen ein großzügigeres Zeitfenster von Minuten (oder mehr) für Antworten besteht. Offline-Systeme verarbeiten Antworten über Stapelverarbeitung oder asynchrone Verarbeitung. Harte verteilte Echtzeitsysteme haben die strengsten Zuverlässigkeitsanforderungen.

Die schwierigsten Herausforderungen mit verteilten Systemen gelten für die harten verteilten Echtzeitsysteme, die auch als Anfrage-/Antwortservices bezeichnet werden. Die Schwierigkeiten entstehen dadurch, dass Anfragen unvorhersehbar eingehen und schnelle Antworten ausgegeben werden müssen (z. B. weil der Kunde aktiv auf die Antwort wartet). Beispiele sind Frontend-Webserver, die Auftragspipeline, Kreditkartentransaktionen, jede AWS-API und Telefonie.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Bestimmen Sie, welches verteilte System erforderlich ist. Zu den Herausforderungen verteilter Systeme gehörten die Latenz, die Skalierung, das Verständnis von Netzwerk-APIs, das Marshalling und Unmarshalling von Daten sowie die Komplexität von Algorithmen wie Paxos. Angesichts des

zunehmenden Wachstums und Verteilungsgrads von Systemen werden theoretische Edge-Fälle zu regelmäßigen Ereignissen.

- Die Amazon Builders' Library: Herausforderungen bei verteilten Systemen
  - In Echtzeit verteilte Systeme erfordern synchrone und schnelle Antworten.
  - Bei weichen Echtzeitsystemen besteht ein großzügigeres Zeitfenster von Minuten (oder mehr) für Antworten.
  - Offline-Systeme verarbeiten Antworten über Stapelverarbeitung oder asynchrone Verarbeitung.
  - Harte verteilte Echtzeitsysteme haben die strengsten Zuverlässigkeitsanforderungen.

Ressourcen

Relevante Dokumente:

- Amazon EC2: Idempotenz sicherstellen
- Die Amazon Builders' Library: Herausforderungen bei verteilten Systemen
- Die Amazon Builders' Library: Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee
- Was ist Amazon EventBridge?
- Was ist Amazon Simple Queue Service?

Relevante Videos:

- AWS New York Summit 2019: Einführung in ereignisgesteuerte Architekturen und Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Kreisläufe schließen & aufgeschlossen sein: Wie man die Kontrolle über Systeme übernimmt – große und kleine ARC337 (umfasst lose Verkoppelung, konstante Ausführung, statische Stabilität)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308) (Umstieg auf ereignisgesteuerte Architekturen)

REL04-BP02 Implementieren lose gekoppelter 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.

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 Verkoppelung unterbricht diese Abhängigkeit, sodass abhängige Komponenten nur die versionierte und veröffentlichte Schnittstelle kennen müssen. Die Implementierung einer losen Kopplung 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 Kopplung 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 lassen sich nicht durch direkte Punkt-zu-Punkt-Interaktion integrieren, sondern in der Regel über eine temporäre, robuste Speicherschicht, z. B. eine Amazon SQS-Warteschlange oder eine Streaming-Datenplattform wie Amazon Kinesis oder AWS Step Functions.

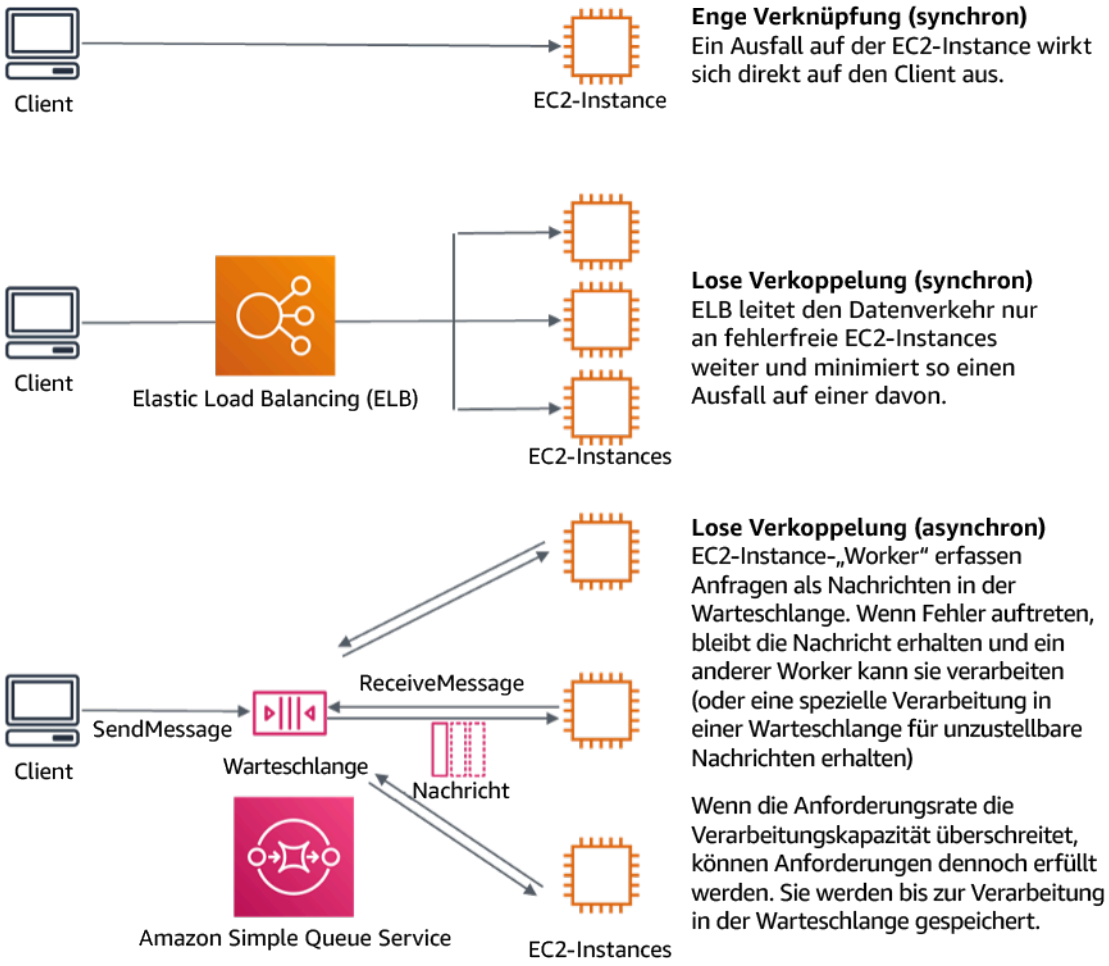


Abbildung 4: Abhängigkeiten etwa zwischen Warteschlangensystemen und Load Balancer sind lose gekoppelt

Amazon SQS-Warteschlangen und Elastic Load Balancers sind nur zwei Möglichkeiten, um eine Zwischenschicht für lose Kopplung hinzuzufügen. Ereignisgesteuerte Architekturen können auch in der AWS Cloud mithilfe von Amazon EventBridge erstellt werden, was Clients (Ereignisproduzenten) von den Services abstrahieren kann, auf die sie sich verlassen (Ereignisverbraucher). Amazon Simple Notification Service (Amazon SNS) ist eine effektive Lösung, wenn Sie Push-basiertes Many-to-Many-Messaging mit hohem Durchsatz benötigen. Mithilfe von Amazon SNS-Themen können Ihre Publisher-Systeme Nachrichten zur parallelen Verarbeitung an eine große Anzahl von Abonnenten-Endpunkten senden.

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.
- APIs werden zwischen Workload-Ebenen direkt aufgerufen, ohne Möglichkeit eines Failovers oder einer asynchronen Verarbeitung der Anfrage.
- Enge Verkoppelung 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 Verkoppelung 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 u. a. Services für die Implementierung vollständig verwalteter Warteschlangen, automatisierter Workflows, die Reaktion auf Ereignisse und APIs, die dazu beitragen können, das Verhalten von Komponenten gegenüber anderen Komponenten zu isolieren und so die Ausfallsicherheit und Agilität zu erhöhen.

- Aufbau ereignisgesteuerter Architekturen: Amazon EventBridge hilft Ihnen beim Aufbau lose gekoppelter und verteilter ereignisgesteuerter Architekturen.

- Implementieren von Warteschlangen in verteilten Systemen: Sie können Amazon Simple Queue Service (Amazon SQS) verwenden, um verteilte Systeme zu integrieren und zu entkoppeln.
- Containerisieren Sie Komponenten als Microservices: Microservices ermöglichen es Teams, Anwendungen zu erstellen, die aus kleinen unabhängigen Komponenten bestehen, die über wohldefinierte APIs kommunizieren. Amazon Elastic Container Service (Amazon ECS) und Amazon Elastic Kubernetes Service (Amazon EKS) können Ihnen helfen, schneller mit Containern zu beginnen.
- Verwalten der Workflows mit Step Functions: Step Functions hilft Ihnen, mehrere AWS-Dienste in flexiblen Workflows zu koordinieren.
- Nutzen von Publish-Subscribe (Pub/Sub)-Messaging-Architekturen: Amazon Simple Notification Service(Amazon SNS) sorgt für die Zustellung von Nachrichten von Publishern an Abonnenten (auch als Produzenten und Verbraucher bezeichnet).

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.
  - Erstellen ereignisgesteuerter Anwendungen mit Amazon EventBridge
  - AWS re:Invent 2022 - Designing Event-Driven Integrations using Amazon EventBridge (AWS re:Invent 2022 – Entwurf ereignisgesteuerter Integrationen mit Amazon EventBridge)
- Verteilte Nachrichtensysteme haben drei Hauptbestandteile, die für eine warteschlangenbasierte Architektur implementiert werden müssen. Dazu gehören Komponenten des verteilten Systems, die Warteschlange, die für die Entkopplung verwendet wird (auf Amazon SQS-Servern verteilt), 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 Amazon SQS-Servern.
  - Grundlegende Amazon SQS-Architektur
  - Senden von Nachrichten zwischen verteilten Anwendungen mit 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.

  - Implementieren von Microservices in AWS
  - Let's Architect! Architektur von Microservices mit Containern
- Mit AWS Step Functions können 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 eines Serverless Workflows mit AWS Step Functions und AWS Lambda
  - Erste Schritte mit AWS Step Functions

Ressourcen

Zugehörige Dokumente:

- Amazon EC2: 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 (Teilen Sie den Monolithen auf)
- Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS (Orchestrieren der warteschlangenbasierten Microservices mit AWS Step Functions und Amazon SQS)
- Grundlegende Amazon SQS-Architektur
- Warteschlangenbasierte Architektur

Zugehörige Videos:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205) (AWS New York Summit 2019: Einführung in ereignisgesteuerte Architekturen und Amazon EventBridge [MAD205])
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability) (AWS re:Invent 2018: Close Loops und Opening Minds: Wie man die Kontrolle über große und kleine Systeme übernimmt ARC337 [umfasst lose Verkoppelung, konstante Ausführung, statische Stabilität])

- AWS re:Invent 2019: Moving to event-driven architectures (SVS308) (Umstieg auf ereignisgesteuerte Architekturen)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (API304) (AWS re:Invent 2019: Skalierbare ereignisgesteuerte Serverless-Anwendungen, die Amazon SQS und Lambda nutzen [API304])
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (AWS re:Invent 2019: Skalierbare ereignisgesteuerte Serverless-Anwendungen, die Amazon SQS und Lambda nutzen)
- AWS re:Invent 2022 - Designing event-driven integrations using Amazon EventBridge (AWS re:Invent 2022 – Entwurf ereignisgesteuerter Integrationen mit Amazon EventBridge)
- AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices (AWS re:Invent 2017: Elastic Load Balancing – Vertiefung und bewährte Praktiken)

REL04-BP03 Konstante Ausführung

Bei größeren, schnellen Lastveränderungen können Systeme ausfallen. Wenn Ihre Workload beispielsweise eine Zustandsprüfung ausführt, die den Zustand vieler tausend Server überwacht, sollte sie 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: Kreisläufe schließen & aufgeschlossen sein: Wie man die Kontrolle über große und kleine Systeme übernimmt ARC337 (umfasst konstante Ausführung)
    - 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

Ähnliche Dokumente:

- Amazon EC2: Idempotenz sicherstellen
- Die Amazon Builders' Library: Herausforderungen für verteilte Systeme
- Die Amazon Builders' Library: Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee

Ähnliche Videos:

- AWS New York Summit 2019: Einführung in ereignisgesteuerte Architekturen und Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Kreisläufe schließen & aufgeschlossen sein: Wie man die Kontrolle über große und kleine Systeme übernimmt ARC337 (umfasst konstante Ausführung)
- AWS re:Invent 2018: Kreisläufe schließen & aufgeschlossen sein: Wie man die Kontrolle über Systeme übernimmt – große und kleine ARC337 (umfasst lose Verkoppelung, konstante Ausführung, statische Stabilität)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308) (Umstieg auf ereignisgesteuerte Architekturen)

REL04-BP04 Festlegen aller Reaktionen als idempotent

Ein idempotenter Service garantiert, dass jede Anfrage genau einmal abgeschlossen wird. Das bedeutet, dass das Senden mehrerer identischer Anfragen den gleichen Effekt hat wie das Senden einer einzelnen Anfrage. Ein idempotenter Service erleichtert es einem 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.

In einem verteilten System ist es einfach, eine Aktion höchstens einmal (der Client stellt nur eine Anforderung) oder mindestens einmal (Anforderung so lange, bis der Client erfolgreich ist) durchzuführen. Es ist jedoch schwer zu gewährleisten, dass eine Aktion idempotent ist, was bedeutet, dass sie genau einmal ausgeführt wird, sodass 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 sich verändernde Anfrage erhalten, ohne dass doppelte Datensätze erstellt werden oder sonstige Probleme entstehen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Legen Sie alle Reaktionen als idempotent fest. Ein idempotenter Service garantiert, dass jede Anfrage genau einmal abgeschlossen wird. Das bedeutet, dass das Senden mehrerer identischer Anfragen den gleichen Effekt hat wie das Senden einer einzelnen Anfrage.
  - Clients können API-Anfragen mit einem Idempotenz-Token ausgeben. Das gleiche Token wird bei einer Wiederholung der Anfrage verwendet. 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.
    - Amazon EC2: Idempotenz sicherstellen

Ressourcen

Ähnliche Dokumente:

- Amazon EC2: Idempotenz sicherstellen
- Die Amazon Builders' Library: Herausforderungen bei verteilten Systemen

- Die Amazon Builders' Library: Zuverlässigkeit, stetige Ausführung und eine gute Tasse Kaffee

Ähnliche Videos:

- AWS New York Summit 2019: Einführung in ereignisgesteuerte Architekturen und Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Kreisläufe schließen & aufgeschlossen sein: Wie man die Kontrolle über Systeme übernimmt – große und kleine ARC337 (umfasst lose Verkoppelung, konstante Ausführung, statische Stabilität)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308) (Umstieg auf ereignisgesteuerte Architekturen)

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. Komponenten des verteilten Systems müssen so funktionieren, dass sie keine negativen Auswirkungen auf andere Komponenten oder die Workload haben. Mit den folgenden bewährten Methoden können Workloads Belastungen oder Ausfällen standhalten, schneller wiederhergestellt werden und die Auswirkungen solcher Beeinträchtigungen verringern. Das Ergebnis ist eine verbesserte mittlere Reparaturzeit (MTTR).

Diese bewährten Methoden verhindern Ausfälle und verbessern die mittlere Zeit 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 Steuern und Einschränken von Wiederholungsaufrufen
- REL05-BP04 Schnelles Scheitern und Begrenzen von Warteschlangen
- REL05-BP05 Festlegen von Client-Zeitüberschreitungen
- REL05-BP06 Erstellen zustandsloser Anwendungen
- REL05-BP07 Implementieren von Nothebeln

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 bei fehlender Befolgung dieser Best Practice: 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. Zum 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 (Amazon API Gateway: Drosseln von API-Anfragen für einen besseren Durchsatz)
- CircuitBreaker (Zusammenfassung des Circuit Breaker aus dem Buch „Release It!“)
- Error Retries and Exponential Backoff in AWS (Fehlerwiederholungen und exponentielles 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:

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

Zugehörige Beispiele:

- Well-Architected Lab: Level 300: Implementieren von Zustandsprüfungen und Verwalten von Abhängigkeiten zur Verbesserung der Zuverlässigkeit

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.

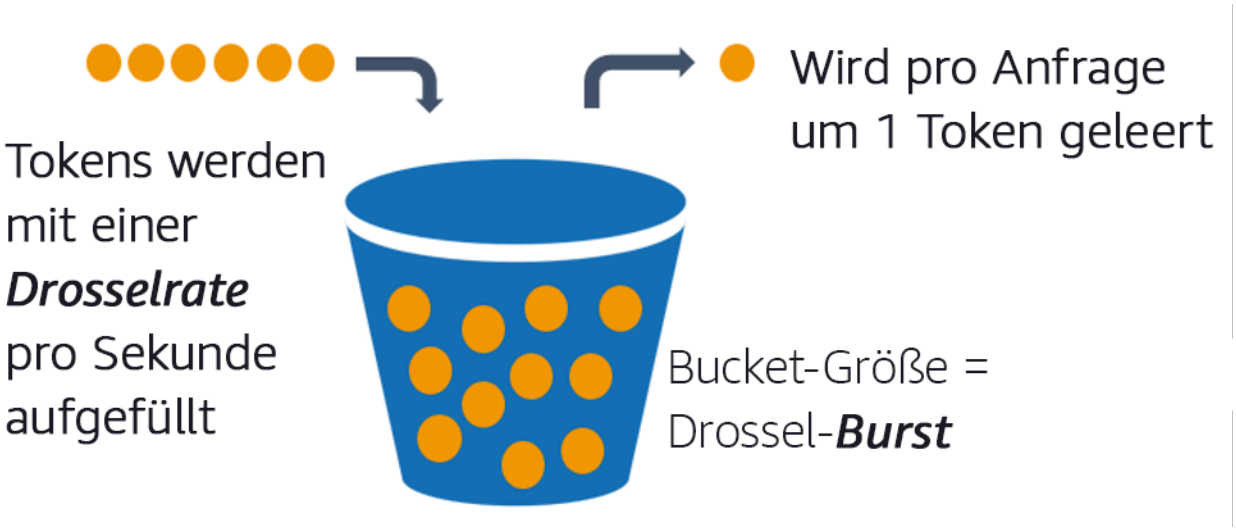
Vorteile der Nutzung dieser bewährten Methode: Workloads, die Drosselgrenzwerte festlegen, können normal arbeiten und akzeptierte Anfragen auch bei unerwarteten Volumenspitzen erfolgreich verarbeiten. Plötzliche oder anhaltende Spitzen von Anfragen an APIs und Warteschlangen werden gedrosselt und verbrauchen 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 bei fehlender Befolgung dieser Best Practice: 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 Konstante Ausführung
- REL05-BP03 Steuern und Einschränken von Wiederholungsaufrufen

Zugehörige Dokumente:

- Amazon API Gateway: Throttle API Requests for Better Throughput (Amazon API Gateway: Drosseln von API-Anfragen für einen besseren Durchsatz)
- AWS WAF: Rate-based rule statement (AWS WAF: Ratenbasierte Regelaussage)
- Introducing maximum concurrency of AWS Lambda when using Amazon SQS as an event source (Einführung maximaler Gleichzeitigkeit von AWS Lambda bei Verwendung von Amazon SQS als Ereignisquelle)
- AWS Lambda: Maximum Concurrency (AWS Lambda: Maximale Gleichzeitigkeit)

Zugehörige Beispiele:

- The three most important AWS WAF rate-based rules (Die drei wichtigsten ratenbasierten Regeln in AWS WAF)
- Java Bucket4j
- Python Token-Bucket
- Node-Token-Bucket
- .NET System Threading Rate Limiting (Ratenbegrenzung für .NET-System-Threading)

Zugehörige Videos:

- Implementing GraphQL API security best practices with AWS AppSync (Implementierung von bewährten Sicherheitsmethoden für GraphQL API mit AWS AppSync)

Zugehörige Tools:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF

REL05-BP03 Steuern und Einschränken 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: Typische Komponenten in einem verteilten Softwaresystem sind Server, Load Balancer, Datenbanken und DNS-Server. 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.
- Wiederholungsversuche werden implementiert, ohne ihre Auswirkungen zu testen, oder es wird davon ausgegangen, dass Wiederholungsversuche bereits in ein SDK integriert sind, ohne Wiederholungsszenarien 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 Best Practice: 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.

Mit AWS SDKs werden Wiederholungen und exponentielles Backoff standardmäßig implementiert. Verwenden Sie diese integrierten AWS-Implementierungen, sofern dies in Ihrem Workload erforderlich 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 der vorhandenen SDKs bewusst, die bewährte Wiederholungsstrategien mit exponentiellem Backoff und Jitter für die Sprache Ihrer Wahl implementieren, und nutzen Sie eher diese, anstatt eigene Wiederholungsimplementierungen zu schreiben.
- Überprüfen Sie, 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 von AWS-Service-APIs die AWS SDKs und AWS CLI und machen Sie sich mit den Konfigurationsoptionen für Wiederholungsversuche vertraut. 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 aller Reaktionen als idempotent
- REL05-BP02 Drosselung von Anfragen
- REL05-BP04 Schnelles Scheitern und Begrenzen von Warteschlangen
- REL05-BP05 Festlegen von Client-Zeitüberschreitungen
- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

Zugehörige Dokumente:

- Error Retries and Exponential Backoff in AWS (Fehlerwiederholungen und exponentielles Backoff in AWS)
- Die Amazon Builders' Library: Timeouts, Wiederholungen und Backoff mit Jitter
- Exponentielles Backoff und Jitter
- Making retries safe with idempotent APIs (Sichere Wiederholungsversuche mit idempotenten APIs)

Zugehörige Beispiele:

- Spring Retry (Spring-Wiederholung)
- Resilience4j Retry (Resilience4j-Wiederholung)

Zugehörige Videos:

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

Zugehörige Tools:

- AWS SDKs und Tools: Wiederholungsverhalten
- AWS Command Line Interface: AWS CLI-Wiederholungen

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 Best Practice: 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 Implementieren lose gekoppelter Abhängigkeiten
- REL05-BP02 Drosselung von Anfragen
- REL05-BP03 Steuern und Einschränken 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
- Schnell scheitern
- Wie kann ich einen zunehmenden Rückstand an Nachrichten in meiner Amazon SQS-Warteschlange verhindern?
- Elastic Load Balancing: Zonenverschiebung
- Amazon Route 53 Application Recovery Controller: Routingsteuerung für Traffic-Failover

Zugehörige Beispiele:

- Muster der Unternehmensintegration: Channel für unzustellbare Nachrichten

Zugehörige Videos:

- AWS re:Invent 2022 – Operating highly available Multi-AZ applications (AWS re:Invent 2022 – Betrieb hochverfügbarer Multi-AZ Anwendungen)

Zugehörige Tools:

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

REL05-BP05 Festlegen von Client-Zeitüberschreitungen

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 bei fehlender Befolgung dieser Best Practice: 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 bewährte Entwurfsmuster wie erneute Versuche und Unterbrecher, um Zeitüberschreitungen problemlos zu behandeln und Ansätze für schnelles Scheitern zu unterstützen. AWS SDKs und AWS CLI ermöglichen die Konfiguration von Zeitüberschreitungen sowohl für Verbindungen als auch für Anfragen sowie für erneute Versuche mit exponentiellem Backoff und Jitter. AWS Lambda -Funktionen unterstützen die Konfiguration von Zeitüberschreitungen. Mit AWS Step Functionskönnen Sie Low-Code-Unterbrecher erstellen, die die Vorteile vorgefertigter Integrationen mit AWS-Services und SDKs nutzen. 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 Anrufe mit einem AWS SDK tätigt, finden Sie in der Dokumentation die sprachspezifische Zeitüberschreitungskonfiguration.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++
- Wenn Sie AWS SDKs oder AWS CLI-Befehle in Ihrem Workload verwenden, konfigurieren Sie die Standardwerte für Zeitüberschreitungen durch Festlegen der AWS -Standardeinstellungen für die Konfiguration für `connectTimeoutInMillis` und `tlsNegotiationTimeoutInMillis`.
- Wenden Sie die Befehlszeilenoptionen `cli-connect-timeout` und `cli-read-timeout` an, um einmalige AWS CLI-Befehle an AWS-Services zu steuern.
- 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-Erkennung von Unregelmäßigkeiten für Aufruffehlerraten, Service-Level-Ziele für Latenz und Latenzausreißer, um Einblicke in den Umgang mit zu aggressiven oder toleranten Zeitüberschreitungen zu erhalten.
- Konfigurieren Sie Zeitüberschreitungen für Lambda-Funktionen.
- API Gateway-Clients müssen bei der Verarbeitung von Zeitüberschreitungen eigene erneute Versuche implementieren. API Gateway unterstützt eine Integrationszeitüberschreitung zwischen 50 Millisekunden und 29 Sekunden für Downstream-Integrationen und versucht es nicht erneut, wenn bei Integrationsanfragen Zeitüberschreitungen auftreten.
- 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.
- Verwenden Sie AWS Step Functions, um Low-Code-Unterbrecher für Remote-Serviceaufrufe zu erstellen, insbesondere beim Aufrufen nativer AWS SDKs und unterstützter Step Functions-Integrationen, um Ihren Workload zu vereinfachen.

Ressourcen

Zugehörige bewährte Methoden:

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

Zugehörige Dokumente:

- AWS SDK: Wiederholungen und Zeitüberschreitungen
- Die Amazon Builders' Library: Timeouts, Wiederholungen und Backoff mit Jitter
- Amazon API Gateway-Kontingente und wichtige Hinweise
- AWS Command Line Interface: Befehlszeilenoptionen
- AWS SDK for Java 2.x: Konfigurieren von API-Timeouts
- AWS Botocore mit dem Konfigurationsobjekt und der Konfigurationsreferenz
- AWS SDK for .NET: Wiederholungen und Zeitüberschreitungen

- AWS Lambda: Konfigurieren von Lambda-Funktionsoptionen

Zugehörige Beispiele:

- Verwenden des Unterbrechermusters mit AWS Step Functions und Amazon DynamoDB
- Martin Fowler: CircuitBreaker

Zugehörige Tools:

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

REL05-BP06 Erstellen zustandsloser Anwendungen

Services 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. Amazon ElastiCache oder Amazon DynamoDB sind gute Ziele für den ausgelagerte Zustand.

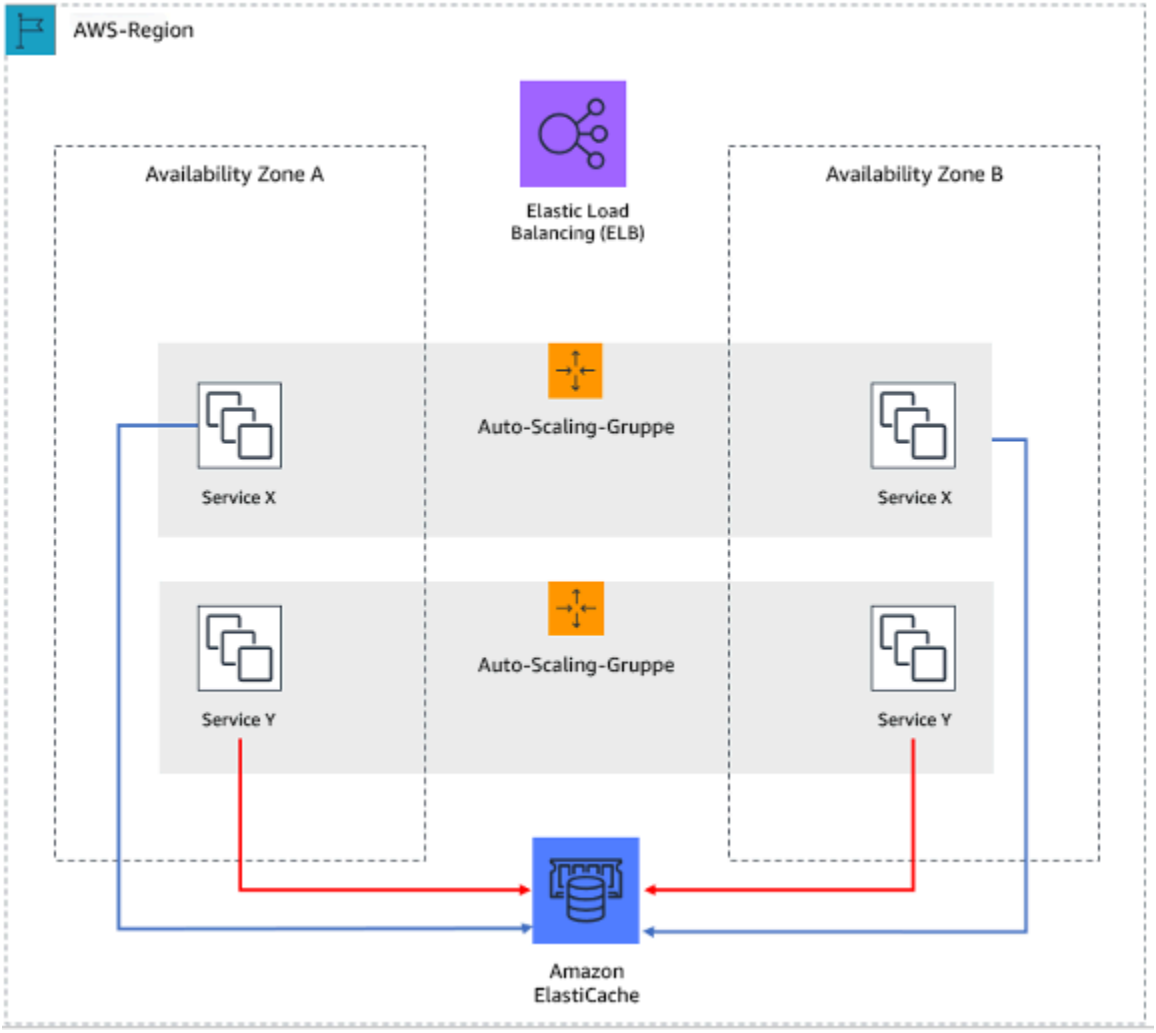


Abbildung 7: In dieser zustandslosen Webanwendung wird der Sitzungsstatus in Amazon ElastiCache ausgelagert.

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 eine Anwendung als zustandslos entwickelt wurde, können Sie serverlose Compute-Services wie AWS Lambda oder AWS Fargate verwenden.

Neben dem Serverersatz besteht ein weiterer Vorteil zustandsloser Anwendungen darin, dass sie horizontal skaliert werden können, da alle verfügbaren Compute-Ressourcen (z. B. EC2-Instances und AWS Lambda-Funktionen) jede Anfrage bearbeiten können.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Erstellen Sie zustandslose Anwendungen. Zustandslose Anwendungen ermöglichen eine horizontale Skalierung und sind gegenüber dem Ausfall eines einzelnen Knotens tolerant.
  - Entfernen Sie Zustände, die tatsächlich in Anfrageparametern gespeichert werden können.
  - Nachdem Sie untersucht haben, ob der Zustand erforderlich ist, verschieben Sie die gesamte Zustandsverfolgung in einen ausfallsicheren Multizonen-Cache oder Datenspeicher wie Amazon ElastiCache, Amazon RDS, Amazon DynamoDB oder in die verteilte Datenlösung eines Drittanbieters. Speichern Sie nicht verlagerbare Zustände in ausfallsicheren Datenspeichern.
    - Manche Daten (wie Cookies) können in Headern oder Abfrageparametern übergeben werden.
    - Entfernen Sie Zustände, die sich schnell in Anfragen übergeben lassen.
    - Einige Daten sind möglicherweise nicht für jede Anfrage erforderlich, sondern können bei Bedarf abgerufen werden.
    - Entfernen Sie asynchron abrufbare Daten.
    - Wählen Sie einen Datenspeicher, der die Anforderungen eines erforderlichen Zustands erfüllt.
    - Ziehen Sie für nichtrelationale Daten eine NoSQL-Datenbank in Erwägung.

Ressourcen

Ähnliche 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

REL05-BP07 Implementieren von Nothebeln

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 über die ordnungsgemäße Funktionsminderung finden Sie unter REL05-BP01 Implementieren einer ordnungsgemäßen Funktionsminderung, um harte Abhängigkeiten in weiche zu ändern.

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 (Jeder Tag kann ein Prime Day sein: Wie die Amazon.com-Suche mit Hilfe von Chaos Engineering über 84.000 Anfragen pro Sekunde bewältigt).
  - 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 Verhindern von bimodalem Verhalten mithilfe statischer Stabilität.
- 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 einer ordnungsgemäßen Funktionsminderung, um harte Abhängigkeiten in weiche zu ändern

- REL05-BP02 Drosselung von Anfragen
- REL11-BP05 Verhindern von bimodalem Verhalten mithilfe statischer Stabilität

Zugehörige Dokumente:

- Automating safe, hands-off deployments (Automatisierung sicherer, vollautomatischer Bereitstellungen)
- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second (Jeder Tag kann ein Prime Day sein: Wie die Amazon.com-Suche mit Hilfe von Chaos Engineering über 84.000 Anfragen pro Sekunde bewältigt)

Zugehörige Videos:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability (AWS re:Invent 2020: Zuverlässlichkeit, Konsistenz und Vertrauen durch Unveränderlichkeit)

Änderungsverwaltung

Änderungen an Ihrem Workload oder der Umgebung müssen vorausgesehen und berücksichtigt werden, um einen zuverlässigen Betrieb der Workload zu erreichen. Zu diesen Änderungen gehören durch äußere Faktoren hervorgerufene Änderungen (z. B. Bedarfsspitzen) sowie interne Änderungen wie Funktionsbereitstellungen und Sicherheitspatches.

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

Themen

Überwachen von Workload-Ressourcen

Protokolle und Metriken sind wertvolle Tools, um einen Einblick in den Zustand Ihrer Workloads zu gewinnen. Sie können Ihre Workload so konfigurieren, dass Protokolle und Metriken überwacht und bei Über- oder Unterschreiten von Schwellenwerten oder wichtigen Ereignissen Benachrichtigungen gesendet werden. Die Überwachung ermöglicht Ihrem Workload, zu erkennen, wenn niedrige Leistungsschwellenwerte überschritten werden oder Ausfälle auftreten, sodass er als Reaktion automatisch wiederhergestellt werden kann.

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 Kennzahlen 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 vier spezifischen Phasen:

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

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 Analysen
- REL06-BP06 Regelmäßiges Durchführen von Prüfungen
- 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 Ihrer Workload 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 zugrunde liegen.

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. Algorithmen zur Erkennung von Anomalien berücksichtigen saisonale Schwankungen und Trendänderungen 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-Canariesgenannt, 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. Zu erkennen, dass eine Workload keine Geschäftsergebnisse erzielt, ermöglicht es Ihnen, schnell einen Systemausfall zu deklarieren und das System nach einem Vorfall wiederherzustellen.

Gängige Antimuster:

- Es werden nur externe Schnittstellen zur 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 Einführung 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, die CloudWatch-Metriken veröffentlichen 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. Dies
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 Transaktionenkönnen Sie Probleme erkennen, bevor Ihre Kunden dies tun.

Ressourcen

Relevante bewährte Methoden:

- REL11-BP03 Automatisieren der Reparatur auf allen Ebenen

Relevante Dokumente:

- Getting started with your AWS Health Dashboard – Your account health (Erste Schritte mit Ihrem AWS Health-Dashboard – Der Zustand Ihres Kontos)
- AWS-Services, die CloudWatch-Metriken veröffentlichen
- Zugriffsprotokolle für Ihren Network Load Balancer
- Zugriffsprotokolle für Ihre Application Load Balancer
- Zugriff auf Amazon CloudWatch Logs für AWS Lambda
- Protokollierung von Amazon S3-Serverzugriffen
- Aktivieren Sie Zugriffsprotokolle für Ihren Classic Load Balancer.
- Exportieren von Protokolldaten zu Amazon S3
- Installieren des CloudWatch-Agenten
- Veröffentlichen benutzerdefinierter Metriken
- Verwenden von Amazon CloudWatch-Dashboards
- Verwenden von Amazon CloudWatch-Metriken
- Verwenden von Synthetic Monitoring
- Was sind Amazon CloudWatch Logs?

Benutzerhandbücher:

- Erstellen eines Trails
- Überwachen von Arbeitsspeicher- und Datenträgermetriken für Amazon EC2 Linux-Instances
- Verwenden von CloudWatch Logs mit Container-Instances
- VPC Flow Logs
- Was ist Amazon DevOps Guru?
- Was ist AWS X-Ray?

Ähnliche Blogs:

- Debugging mit Amazon CloudWatch Synthetics und AWS X-Ray

Ähnliche Beispiele und Workshops:

- AWS Well-Architected Labs: Operational Excellence - Dependency Monitoring (AWS Well-Architected Labs: Operative Exzellenz – Überwachung von Abhängigkeiten)
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen
- Workshop zur Beobachtbarkeit

REL06-BP02 Definieren und Berechnen von Metriken (Aggregierung)

Speichern Sie Protokolldaten und wenden Sie gegebenenfalls Filter an, um Metriken zu berechnen. Dazu gehören z. B. die Anzahl eines bestimmten Protokollereignisses oder die Latenz, die aus den Zeitstempeln des Protokollereignisses berechnet wird.

Amazon CloudWatch und Amazon S3 dienen als primäre Aggregierungs- und Speicherebenen. Bei einigen Services wie AWS Auto Scaling und Elastic Load Balancing werden Standardkennzahlen für die CPU-Last oder die durchschnittliche Anfragelatenz eines Clusters oder einer Instance bereitgestellt. Für Streaming-Services wie VPC Flow Logs und AWS CloudTrail werden Ereignisdaten an CloudWatch Logs weitergeleitet und Sie müssen Filter definieren und anwenden, um Metriken aus diesen Ereignisdaten zu extrahieren. Auf diese Weise erhalten Sie Zeitreihendaten, die als Eingaben für CloudWatch-Alarme dienen können, die Sie zum Auslösen von Warnungen definieren.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Definieren und berechnen Sie Metriken (Aggregierung). Speichern Sie Protokolldaten und wenden Sie gegebenenfalls Filter an, um Metriken zu berechnen. Dazu gehören z. B. die Anzahl eines bestimmten Protokollereignisses oder die Latenz, die aus den Zeitstempeln des Protokollereignisses berechnet wird.
  - Metrikfilter definieren die Begriffe und Muster, die in Protokolldaten zu suchen sind, wenn diese an CloudWatch Logs gesendet werden. CloudWatch Logs verwendet diese Metrikfilter, um Protokolldaten in numerische CloudWatch-Metriken umzuwandeln, die Sie grafisch darstellen oder für die Sie einen Alarm einrichten können.
    - Suchen und Filtern von Protokolldaten
  - Verwenden Sie einen vertrauenswürdigen Drittanbieter für die Protokollaggregierung.
    - Befolgen Sie die Anweisungen des Drittanbieters. Die meisten Produkte von Drittanbietern lassen sich in CloudWatch und Amazon S3 integrieren.
  - Einige AWS-Services können Protokolle direkt in Amazon S3 veröffentlichen. Wenn die Speicherung von Protokollen in Amazon S3 die wichtigste Anforderung ist, kann der Protokoll-Service die Protokolle direkt an Amazon S3 senden, ohne dass eine zusätzliche Infrastruktur eingerichtet werden muss.
    - Senden von Protokollen direkt an Amazon S3

Ressourcen

Relevante Dokumente:

- Amazon CloudWatch Logs Insights-Beispielabfragen
- Debugging mit Amazon CloudWatch Synthetics und AWS X-Ray
- Workshop zur Beobachtbarkeit
- Suchen und Filtern von Protokolldaten
- Senden von Protokollen direkt an Amazon S3
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen

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 Best Practice: 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 durchführt oder ein OpsItem-Element oder einen Vorfall in AWS Systems Manager erstellen.

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 Konfigurieren von 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.

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 Streamsund 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. Amazon SNS Die typische Latenz 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 S3und 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-Themazu 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. 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-Erkenntnisse
- Verwenden von Amazon CloudWatch-Alarmen
- Verwenden von Amazon CloudWatch-Dashboards
- Using Amazon CloudWatch metrics (Verwenden von Amazon CloudWatch-Metriken)
- Einrichtung von Amazon SNS-Benachrichtigungen
- CloudWatch-Erkennung von Unregelmäßigkeiten
- CloudWatch Logs-Datenschutz
- Amazon EventBridge
- Amazon Simple Notification Service

Zugehörige Videos:

- re:Invent 2022 observability videos
- AWS re:Invent 2022 – Observability best practices at Amazon (AWS re:Invent 2022 – Bewährte Überwachungsmethoden bei Amazon)

Zugehörige Beispiele:

- Workshop zur Beobachtbarkeit
- Amazon EventBridge bis AWS Lambda mit Feedbacksteuerung durch Amazon CloudWatch-Alarme

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.

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, wenn diese bewährte Methode nicht eingeführt wird: 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 Mit Runbooks arbeiten. Ausführliche Informationen zum Anzeigen von Runbook-Inhalten finden Sie unter Runbook-Inhalt anzeigen.
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 durchgeführt werden sollen, wenn ein Ereignis einer Regel entspricht. Zu den Aktionen, die automatisch ausgelöst werden können, gehören der Aufruf einer AWS Lambda-Funktion, der Aufruf des Amazon EC2 `Run Command`, die Weiterleitung des Ereignisses an Amazon Kinesis Data Streams und die Anzeige von Automatisieren von Amazon EC2 mit 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-Automatisierung
- Erstellen einer EventBridge-Regel, die durch ein Ereignis aus einer AWS-Ressource ausgelöst wird
- Workshop zur Beobachtbarkeit
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen
- Was ist Amazon DevOps Guru?
- Arbeiten mit Automation-Dokumenten (Playbooks)

Zugehörige Videos:

- AWS re:Invent 2022 - Observability best practices at Amazon (AWS re:Invent 2022: Bewährte Überwachungsmethoden bei Amazon)
- AWS re:Invent 2020: Automate anything with AWS Systems Manager(AWS re:Invent 2020: Automatiserung mit AWS Systems Manager)
- Introduction to AWS Resilience Hub(Einführung in AWS Resilience Hub)
- Create Custom Ticket Systems for Amazon DevOps Guru Notifications (Benutzerdefinierte Ticketsysteme für x Benachrichtigungen erstellen Amazon DevOps Guru)
- Enable Multi-Account Insight Aggregation with Amazon DevOps Guru (Aktivieren der Erkenntnisaggregierung bei mehreren Konten mithilfe von Amazon DevOps Guru)

Zugehörige Beispiele:

- Workshops zur Zuverlässigkeit
- Amazon CloudWatch- und Systems Manager-Workshop

REL06-BP05 Analysen

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 und dennoch leistungsstarke Abfragesprache, mit der Sie Protokolldaten analysieren können. Amazon CloudWatch Logs unterstützt auch Abonnements, mit denen Daten nahtlos nach 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 unterstütz. Unter Unterstützte SerDes- und Datenformate 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 S3 Glacier weiterleiten und nach Ablauf der Aufbewahrungsperiode die Speicherung 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, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Mit CloudWatch Logs Insights können Sie Protokolldaten in Amazon CloudWatch Logs interaktiv durchsuchen und analysieren.
  - Analysieren von Protokolldaten mit CloudWatch Logs Insights
  - Amazon CloudWatch Logs Insights-Beispielabfragen
- Verwenden Sie Amazon CloudWatch Logs, um Protokolle an Amazon S3 zu senden, wo Sie sie nutzen oder Amazon Athena verwenden können, um die Abfrage der Daten nutzen können.
  - Wie analysiere ich meine Amazon S3-Serverzugriffsprotokolle mit Athena?

- Erstellen Sie eine S3-Lebenszyklusrichtlinie für Ihren Bucket mit den Serverzugriffsprotokollen. Konfigurieren Sie die Richtlinie so, dass Protokolldateien regelmäßig entfernt werden. Dies reduziert die Datenmenge, die Athena für die einzelnen Abfragen analysiert.
  - Wie erstelle ich eine Lebenszyklusrichtlinie für einen S3-Bucket?

Ressourcen

Relevante Dokumente:

- Amazon CloudWatch Logs Insights-Beispielabfragen
- Analysieren von Protokolldaten mit CloudWatch Logs Insights
- Debugging mit Amazon CloudWatch Synthetics und AWS X-Ray
- Wie erstelle ich eine Lebenszyklusrichtlinie für einen S3-Bucket?
- Wie analysiere ich meine Amazon S3-Serverzugriffsprotokolle mit Athena?
- Workshop zur Beobachtbarkeit
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen

REL06-BP06 Regelmäßiges Durchführen von Prüfungen

Prüfen Sie regelmäßig, wie die Workload-Überwachung implementiert ist, und aktualisieren Sie sie auf Grundlage wichtiger Ereignisse und Änderungen.

Eine effektive Überwachung basiert auf wichtigen Geschäftsmetriken. Stellen Sie sicher, dass diese Metriken in Ihrer Workload berücksichtigt werden, wenn sich geschäftliche Prioritäten ändern.

Durch die Prüfung Ihrer Überwachung stellen Sie sicher, dass Sie wissen, wann eine Anwendung die eigenen Verfügbarkeitsziele erfüllt. Für die Durchführung von Ursachenanalysen ist es erforderlich, bei Ausfällen ermitteln zu können, was passiert ist. AWS bietet Services, mit denen Sie den Status Ihrer Services während eines Vorfalls nachverfolgen können.

- Amazon CloudWatch Logs: Sie können Ihre Protokolle in diesem Service speichern und die Inhalte überprüfen.
- Amazon CloudWatch Logs Insights: Ein vollständig verwalteter Service, mit dem Sie umfangreiche Protokolle innerhalb von Sekunden analysieren können. Es bietet Ihnen schnelle, interaktive Abfragen und Visualisierungen.

- AWS Config: Sie können sehen, welche AWS-Infrastruktur zu verschiedenen Zeitpunkten verwendet wurde.
- AWS CloudTrail: Mit diesem Service können Sie erkennen, welche AWS-APIs zu welchem Zeitpunkt und durch welchen Prinzipal aufgerufen wurden.

Bei AWS werden wöchentliche Meetings abgehalten, um die Produktionsleistung zu prüfen und Erkenntnisse mit anderen Teams zu teilen. Da es so viele Teams in AWS gibt, haben wir Das Rad entwickelt, um zufällig eine zu überprüfende Workload auszuwählen. Der Aufbau einer Struktur mit regelmäßigen Überprüfungen der betrieblichen Leistung und mit Wissensaustausch verbessert Ihre Fähigkeit, höhere Leistungen bei Ihren Betriebsteams zu erzielen.

Gängige Antimuster:

- Es werden nur Standardmetriken erfasst.
- Es wird eine Überwachungsstrategie festgelegt, aber nie überprüft.
- Bei Bereitstellung größerer Änderungen wird die Überwachung nicht erörtert.

Vorteile der Einführung dieser bewährten Methode: Durch die regelmäßige Prüfung der Überwachung können Sie mögliche Probleme vorhersehen, statt nur zu reagieren, wenn ein Problem tatsächlich auftritt.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Erstellen Sie mehrere Dashboards für die Workload. Ein übergeordnetes Dashboard mit den wichtigsten Geschäftsmetriken ist unverzichtbar. Es sollte zudem die technischen Metriken enthalten, die Sie für den prognostizierten Zustand der Workload bei variabler Nutzung als die relevantesten eingestuft haben. Dashboards für verschiedene Anwendungsebenen und Abhängigkeiten, die untersucht werden können, sind ebenfalls empfehlenswert.
  - Verwenden von Amazon CloudWatch-Dashboards
- Planen und prüfen Sie die Workload-Dashboards regelmäßig. Führen Sie regelmäßige Untersuchungen der Dashboards durch. Was die Gründlichkeit der Untersuchungen angeht, sind unterschiedliche Intervalle denkbar.
  - 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: ansteigende Latenz, Nachlassen der primären Geschäftsfunktion und zunehmende Anzahl von Reaktionen auf Fehler.

- Spüren Sie Ausreißer/Anomalien in den Metriken auf. Ausreißer sind anhand von Durchschnitts- oder Mittelwerten oder Anomalien nicht unbedingt erkennbar. Sehen Sie sich die höchsten und niedrigsten Werte in einem bestimmten Zeitraum an und untersuchen Sie die Ursachen für die extremen Werte. Beseitigen Sie nach und nach die Ursachen und legen Sie dabei einen engeren Maßstab für die Definition von Extremwerten an. So können Sie die Beständigkeit der Workload-Leistung weiter erhöhen.
- 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.

Ressourcen

Ähnliche Dokumente:

- Amazon CloudWatch Logs Insights-Beispielabfragen
- Debugging mit Amazon CloudWatch Synthetics und AWS X-Ray
- Workshop zur Beobachtbarkeit
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen
- Verwenden von Amazon CloudWatch-Dashboards

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 bei fehlender Befolgung dieser Best Practice: 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 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 Management Console.
- Instrumentenanwendungen AWS Distro for OpenTelemetry 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-Downstream-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 Relicoder 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 mit Amazon CloudWatch Synthetics und AWS X-Ray
- Die Amazon Builders' Library: Verteilte Systeme instrumentieren, um betriebliche Transparenz zu erzielen
- Integration von AWS X-Ray in andere AWS-Dienste
- AWS Distro for OpenTelemetry und AWS X-Ray
- Amazon CloudWatch: Synthetische Überwachung verwenden
- Amazon CloudWatch: CloudWatch RUM verwenden
- Amazon CloudWatch Synthetics Canary und Amazon CloudWatch-Alarm einrichten
- Verfügbarkeit und mehr: Verdeutlichung und Verbesserung der Ausfallsicherheit bei verteilten Systemen in AWS

Zugehörige Beispiele:

- Workshop zur Beobachtbarkeit

Zugehörige Videos:

- AWS re:Invent 2022: Kontenübergreifendes Überwachen Ihrer Anwendungen
- Überwachen Ihrer AWS-Anwendungen

Zugehörige Tools:

- AWS X-Ray
- Amazon CloudWatch

- Amazon Route 53

Entwerfen einer Workload, die sich an Bedarfsänderungen anpasst

Ein skalierbarer Workload bietet die Elastizität, Ressourcen automatisch entsprechend dem aktuellen Bedarf hinzuzufügen oder zu entfernen.

Bewährte Methoden

- REL07-BP01 Automatisches Abrufen und 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 Durchführen von Lasttests für die Workload

REL07-BP01 Automatisches Abrufen und Skalieren von Ressourcen:

Wenn Sie beeinträchtigte Ressourcen ersetzen oder Ihre Workload skalieren, können Sie den Prozess mithilfe von verwalteten AWS-Services wie Amazon S3 und AWS Auto Scaling automatisieren. Sie können die Skalierung auch mit Tools von Drittanbietern und AWS SDKs automatisieren.

Zu den verwalteten AWS-Services gehören Amazon S3, Amazon CloudFront, AWS Auto Scaling, AWS Lambda, Amazon DynamoDB, AWS Fargate und Amazon Route 53.

Mit AWS Auto Scaling können Sie beeinträchtigte Instances erkennen und ersetzen. Außerdem können Sie Skalierungspläne für Ressourcen erstellen, unter anderem für Amazon EC2 -Instances und Spot-Flotten, Amazon ECS -Aufgaben, Amazon DynamoDB -Tabellen und -Indizes sowie für Amazon Aurora -Replicas.

Bei der Skalierung von EC2-Instances sollten Sie mehrere Availability Zones nutzen (mindestens drei) und Kapazität hinzufügen oder entfernen, um ein Gleichgewicht über diese Availability Zones hinweg zu gewährleisten. ECS-Aufgaben oder Kubernetes-Pods (bei Verwendung von Amazon Elastic Kubernetes Service) sollten ebenfalls über mehrere Availability Zones hinweg verteilt werden.

Bei Verwendung von AWS Lambda werden Instances automatisch skaliert. Jedes Mal, wenn eine Ereignisbenachrichtigung für Ihre Funktion eingeht, ermittelt AWS Lambda schnell freie Kapazität innerhalb seiner Compute-Flotte und führt Ihren Code bis zur zugeteilten Gleichzeitigkeit aus. Sie

müssen sicherstellen, dass die erforderliche Gleichzeitigkeit auf dem spezifischen Lambda und in Ihrem Service Quotas konfiguriert ist.

Amazon S3 wird automatisch skaliert, um hohe Anfrageraten zu verarbeiten. Beispielsweise kann Ihre Anwendung mindestens 3 500 PUT/COPY/POST/DELETE- oder 5 500 GET/HEAD-Anfragen pro Sekunde pro Präfix in einem Bucket erreichen. Für die Anzahl der Präfixe in einem Bucket gibt es keine Beschränkungen. Sie können Ihre Lese- oder Schreibleistung erhöhen, indem Sie Lesevorgänge parallelisieren. Wenn Sie beispielsweise 10 Präfixe in einem Amazon S3-Bucket erstellen, können Sie die Leseleistung auf 55 000 Leseanfragen pro Sekunde skalieren, um die Lesevorgänge zu parallelisieren.

Konfigurieren und nutzen Sie Amazon CloudFront oder ein vertrauenswürdiges Content Delivery Network (CDN). Ein CDN kann Antwortzeiten für Endbenutzer verkürzen und Anfragen für Inhalte aus dem Cache verarbeiten. Dadurch wird die Notwendigkeit zur Skalierung Ihrer Workload verringert.

Gängige Antimuster:

- Es werden Auto-Scaling-Gruppen für die automatisierte Reparatur implementiert, aber keine Elastizität.
- Als Reaktion auf stark ansteigenden Datenverkehr wird automatisch skaliert.
- Es werden hochgradig zustandsbehaftete Anwendungen bereitgestellt, wodurch die Option der Elastizität entfällt.

Vorteile der Einführung dieser bewährten Methode: Durch die Automatisierung entfällt die Gefahr manueller Fehler bei der Bereitstellung und Außerbetriebnahme von Ressourcen. Durch die Automatisierung entfällt das Risiko von Kostenüberschreitungen und Dienstverweigerungen (Denial of Service) aufgrund der langsamen Reaktion auf Bedürfnisse bezüglich der Bereitstellung oder Außerbetriebnahme von Ressourcen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Konfigurieren und nutzen Sie AWS Auto Scaling. Hiermit erfolgt eine Überwachung der Anwendungen und eine automatische Anpassung der Kapazität, um eine stabile, vorhersehbare Leistung zu möglichst niedrigen Kosten aufrechtzuerhalten. Mit AWS Auto Scaling lässt sich die Anwendungsskalierung für mehrere Ressourcen in mehreren Services einrichten.
  - Was ist AWS Auto Scaling?

    - Konfigurieren Sie Auto Scaling nach Bedarf in Ihren Amazon EC2-Instances und Spot-Flotten, Amazon ECS-Aufgaben, Amazon DynamoDB-Tabellen und -Indizes, Amazon Aurora-Replikaten und AWS Marketplace-Appliances.
        - Automatische Verwaltung der Durchsatzkapazität mit DynamoDB Auto Scaling
            - Legen Sie über die Service-API Alarme, Skalierungsrichtlinien sowie Aufwärm- und Abkühlungszeiten fest.
- Nutzen Sie Elastic Load Balancing. Load Balancer können die Last nach Pfaden oder Netzwerkkonnektivität verteilen.
    - Was ist Elastic Load Balancing?
        - Application Load Balancers kann Lasten nach Pfaden verteilen.
            - Was ist ein Application Load Balancer?
                - Konfigurieren Sie einen Application Load Balancer, um Datenverkehr basierend auf dem Pfad unter dem Domänennamen auf verschiedene Workloads zu verteilen.
                - Mit Application Load Balancers können Sie Lasten entsprechend dem AWS Auto Scaling verteilen, um den Bedarf zu verwalten.
                    - Nutzen eines Load Balancer mit einer Auto-Scaling-Gruppe
        - Network Load Balancer können Lasten nach Verbindungen verteilen.
            - Was ist ein Network Load Balancer?
                - Konfigurieren Sie einen Network Load Balancer, um Datenverkehr auf verschiedene Workloads mit TCP zu verteilen oder einen konstanten Satz von IP-Adressen für die Workload festzulegen.
                - Mit Network Load Balancern können Sie Lasten entsprechend dem AWS Auto Scaling verteilen, um den Bedarf zu verwalten.
- Nutzen Sie einen hochverfügbaren DNS-Anbieter. Mithilfe von DNS-Namen können Ihre Benutzer anstelle von IP-Adressen Namen eingeben, um auf Ihre Workloads zuzugreifen. Diese Informationen werden innerhalb einer definierten Reichweite (meist weltweit) für Benutzer der Workload verteilt.
    - Nutzen Sie Amazon Route 53 oder einen vertrauenswürdigen DNS-Anbieter.
        - Was ist Amazon Route 53?
    - Mit Route 53 können Sie Ihre CloudFront-Verteilungen und Load Balancer verwalten.
        - Ermitteln Sie die zu verwaltenden Domänen und Subdomänen.
        - Erstellen Sie entsprechende Datensätze mithilfe von ALIAS- oder CNAME-Datensätzen.

        - Arbeiten mit Datensätzen
- Nutzen Sie das globale AWS-Netzwerk, um den Pfad von den Benutzern zu Ihren Anwendungen zu optimieren. AWS Global Accelerator überwacht kontinuierlich den Zustand der Anwendungsendpunkte und leitet den Datenverkehr in weniger als 30 Sekunden an fehlerfreie Endpunkte um.
    - Bei AWS Global Accelerator handelt es sich um einen Service, der die Verfügbarkeit und Leistung der Anwendungen bei lokalen oder weltweiten Benutzern verbessert. Er stellt statische IP-Adressen bereit, die als fester Einstiegspunkt zu den Anwendungsendpunkten in einer einzelnen oder in mehreren AWS-Regionen fungieren, z. B. Application Load Balancers, Network Load Balancer oder Amazon EC2-Instances.
        - Was ist AWS Global Accelerator?
- Konfigurieren und nutzen Sie Amazon CloudFront oder ein vertrauenswürdiges Content Delivery Network (CDN). Ein Content Delivery Network kann Antwortzeiten für Endbenutzer verkürzen und Anfragen für Inhalte verarbeiten, die zu einer unnötigen Skalierung Ihrer Workloads führen könnten.
    - Was ist Amazon CloudFront?
        - Konfigurieren Sie Amazon CloudFront-Verteilungen für Ihre Workloads oder verwenden Sie das CDN eines Drittanbieters.
            - Sie können festlegen, dass die Workloads nur über CloudFront zugänglich sind. Legen Sie hierfür die IP-Bereiche für CloudFront in den Sicherheitsgruppen oder Zugriffsrichtlinien der Endpunkte fest.

Ressourcen

Relevante Dokumente:

- APN-Partner: Partner, die Ihnen beim Erstellen automatisierter Datenverarbeitungslösungen helfen können
- 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?
- Arbeiten mit Datensätzen

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 Management Console 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. o 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, sogenannte Auto Scaling-Gruppen. 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?
    - Bei der automatischen Skalierung von Amazon DynamoDB wird der Application Auto Scaling-Service genutzt, um die bereitgestellte Durchsatzkapazität in Ihrem Auftrag dynamisch an die Muster des tatsächlichen Datenverkehrs anzupassen. So kann eine Tabelle oder ein GSI die bereitgestellte Lese- und Schreibkapazität erhöhen, um einen plötzlichen Anstieg des Datenverkehrs ohne Drosselung zu bewältigen. Weitere Informationen finden Sie unter Automatische Verwaltung der Durchsatzkapazität mit DynamoDB-Auto-Scaling.

Ressourcen

Zugehörige bewährte Methoden:

- REL07-BP01 Automatisches Abrufen und 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

Skalieren Sie Ressourcen proaktiv, um den Bedarf zu erfüllen und Auswirkungen auf die Verfügbarkeit zu vermeiden.

Viele AWS-Services werden automatisch dem Bedarf entsprechend skaliert. Wenn Sie Amazon EC2-Instances oder Amazon ECS-Cluster verwenden, können Sie die automatische Skalierung dieser Instances auf der Grundlage von Nutzungsmetriken konfigurieren, die dem Bedarf Ihrer Workload entsprechen. Für Amazon EC2 können Sie die durchschnittliche CPU-Auslastung, die Anzahl der Load Balancer-Anfragen oder die Netzwerkbandbreite verwenden, um EC2-Instances zu skalieren. Für Amazon ECS können Sie die durchschnittliche CPU-Auslastung, die Anzahl der Load-Balancer-Anfragen und die Speichernutzung verwenden, um ECS-Aufgaben auf- oder abzuskalieren. Mit Target Auto Scaling auf AWS fungiert der Autoscaler wie ein Haushaltsthermostat, der Ressourcen hinzufügt oder entfernt, um den von Ihnen angegebenen Zielwert (z. B. 70 % CPU-Auslastung) beizubehalten.

AWS Auto Scaling kann auch Predictive Auto Scalingdurchführen. Dabei wird Machine Learning verwendet, um die bisherige Workload jeder Ressource zu analysieren und regelmäßig die zukünftige Last für die nächsten zwei Tage zu prognostizieren.

Das Gesetz von Little hilft beim Berechnen der Anzahl von Compute-Instances, die Sie benötigen (EC2-Instances, gleichzeitige Lambda-Funktionen usw.).

$L = \lambda W$

L = Anzahl der Instances (oder mittlere Gleichzeitigkeit im System)

λ = mittlere Rate des Eingangs von Anfragen (Anfrage/Sekunde)

W = mittlere Zeit, die jede Anfrage im System verbringt (Sekunden)

Wenn beispielsweise bei 100 RPS die Verarbeitung jeder Anfrage 0,5 Sekunden dauert, benötigen Sie 50 Instances, um mit dem Bedarf Schritt zu halten.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Rufen Sie Ressourcen ab, wenn Sie feststellen, dass für eine Workload mehr Ressourcen benötigt werden. Skalieren Sie Ressourcen proaktiv, um den Bedarf zu erfüllen und Auswirkungen auf die Verfügbarkeit zu vermeiden.
  - Berechnen Sie, wie viele Rechenressourcen Sie benötigen (Gleichzeitigkeit der Datenverarbeitung), um eine bestimmte Anfragerate zu verarbeiten.
    - Berichte über das Gesetz von Little
  - Wenn Sie über ein Verlaufsmuster für die Nutzung verfügen, richten Sie die geplante Skalierung für Amazon EC2 ein.
    - Geplante Skalierung für Amazon EC2 Auto Scaling
  - Verwenden Sie die vorausschauende Skalierung von AWS.
    - Prädiktive Skalierung für EC2, unterstützt von Machine Learning

Ressourcen

Relevante Dokumente:

- AWS Auto Scaling: Funktionsweise von Skalierungsplänen
- AWS Marketplace: Für Auto Scaling geeignete Produkte
- Automatische Verwaltung der Durchsatzkapazität mit DynamoDB Auto Scaling
- Prädiktive Skalierung für EC2, unterstützt von Machine Learning
- Geplante Skalierung für Amazon EC2 Auto Scaling
- Berichte über das Gesetz von Little
- Was ist Amazon EC2 Auto Scaling?

REL07-BP04 Durchführen von Lasttests für die Workload

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

Es ist wichtig, regelmäßige Lasttests durchzuführen. Mit diesen Tests können Sie die Belastungsgrenze Ihrer Workload ermitteln und deren Leistung prüfen. AWS erleichtert das Einrichten temporärer Testumgebungen, die den Umfang Ihrer Produktions-Workload 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 lokalen 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.

Gängige Antimuster:

- Es werden Lasttests für Bereitstellungen durchgeführt, die nicht mit der Konfiguration der Produktionsumgebung übereinstimmen.
- Lasttests werden nur für einzelne Teile, nicht aber für die gesamte 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 Einführung 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.
  - Verteilte Lasttests auf AWS: Simulation Tausender 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

Relevante Dokumente:

- Verteilte Lasttests auf AWS: Simulation Tausender verbundener Benutzer

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 diese Änderungen nicht kontrolliert stattfinden, ist es schwierig, ihre Auswirkungen vorherzusagen oder daraus entstehende Probleme zu beheben.

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.

Gängige Antimuster:

- Durchführen ungeplanter Änderungen an der Konfiguration in der Produktion.
- Überspringen von Schritten in Ihrem Plan, um schneller bereitzustellen, was dann jedoch zum Fehlschlagen der Bereitstellung führt.
- Vornehmen von Änderungen, ohne die Umkehrung der Änderung zu testen.

Vorteile der Einführung dieser bewährten Methode: Die effektive Änderungsplanung erhöht Ihre Fähigkeit, die Änderung erfolgreich auszuführen, da Sie sich über alle 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.
  - AWS-Well-Architected-Framework: Konzepte: Runbook
- 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 zur Versionskontrolle eine Quellkontrolle wie AWS CodeCommit oder das vertrauenswürdige Tool eines Drittanbieters.
    - Was ist AWS CodeCommit?

Ressourcen

Relevante Dokumente:

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

Ähnliche Beispiele:

- Automating operations with Playbooks and Runbooks (Vorgänge mit Playbooks und Runbooks automatisieren)

REL08-BP02 Integrieren von Funktionstests in die Bereitstellung

Funktionstests werden im Rahmen der automatisierten Bereitstellung ausgeführt. Wenn die Erfolgskriterien nicht erfüllt sind, wird die Pipeline angehalten oder rückgängig gemacht.

Diese Tests werden in einer Vorproduktionsumgebung ausgeführt, die vor der Produktion in der Pipeline bereitgestellt wird. Idealerweise erfolgt dies im Rahmen einer Bereitstellungspipeline.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Integrieren Sie Funktionstests in Ihre Bereitstellung. Funktionstests werden im Rahmen der automatisierten Bereitstellung ausgeführt. Wenn die Erfolgskriterien nicht erfüllt sind, wird die Pipeline angehalten oder rückgängig gemacht.
  - Rufen Sie AWS CodeBuild während der „Testaktion“ Ihrer in AWS CodePipeline modellierten Software-Release-Pipelines auf. Mit dieser Funktion können Sie ganz einfach verschiedene Tests für Ihren Code ausführen, z. B. Komponententests, statische Code-Analysen und Integrationstests.
    - AWS CodePipeline fügt Unterstützung für Komponententests und angepasste Integrationstests mit AWS CodeBuild hinzu.
  - Verwenden Sie AWS Marketplace-Lösungen, um als Teil Ihrer Softwarebereitstellungs-Pipeline automatisierte Tests auszuführen.
    - Automatisierung von Softwaretests

Ressourcen

Relevante Dokumente:

- AWS CodePipeline fügt Unterstützung für Komponententests und angepasste Integrationstests mit AWS CodeBuild hinzu.
- Automatisierung von Softwaretests
- Was ist AWS CodePipeline?

REL08-BP03 Integrieren von Ausfallsicherheitstests in die Bereitstellung

Ausfallsicherheitstests (unter Anwendung der Grundlagen des Chaos-Engineering) werden als Teil der automatisierten Bereitstellungs-Pipeline in einer Vorproduktionsumgebung ausgeführt.

Diese Tests werden in einer Vorproduktionsumgebung in der Pipeline bereitgestellt und ausgeführt. Sie sollten auch in der Produktion ausgeführt werden, aber im Rahmen von Ernstfallübungen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Integrieren Sie Ausfallsicherheitstests in Ihre Bereitstellung. Verwenden Sie Chaos-Engineering, die Disziplin des Experimentierens an einer Workload, um Vertrauen in die Fähigkeit der Workload aufzubauen, turbulente Bedingungen in der Produktion zu bewältigen.
  - Ausfallsicherheitstests schleusen Fehler oder die Verschlechterung von Ressourcen ein, um zu bewerten, ob Ihre Workload mit der vorgesehenen Resilienz reagiert.
    - Well-Architected Lab: Level 300: Testen auf Resilienz von EC2 RDS and S3
  - Diese Tests können regelmäßig für automatisierte Bereitstellungs-Pipelines in Vorproduktionsumgebungen ausgeführt werden.
  - Sie sollten auch in der Produktion als Teil geplanter Ernstfallübungen ausgeführt werden.
  - Entwickeln Sie unter Verwendung von Grundsätzen des Chaos-Engineering Hypothesen zur Leistung Ihrer Workload bei verschiedenen Beeinträchtigungen. Testen Sie dann Ihre Hypothesen mithilfe von Resilienztests.
    - Grundlagen des Chaos-Engineering

Ressourcen

Relevante Dokumente:

- Grundlagen des Chaos-Engineering
- Was ist AWS Fault Injection Simulator?

Ähnliche Beispiele:

- Well-Architected Lab: Level 300: Testen auf Resilienz von EC2 RDS and S3

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 Automatisieren von Änderungen und Automating safe, hands-off deployments (Automatisierung sicherer, vollautomatischer Bereitstellungen).

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.

Blau/Grün-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 Amazon Route 53 durchzuführen. Sie können AWS CodeDeploy und AWS Elastic Beanstalk mit einer Bereitstellungskonfiguration konfigurieren, die eine Blau/Grün-Bereitstellung ermöglicht.

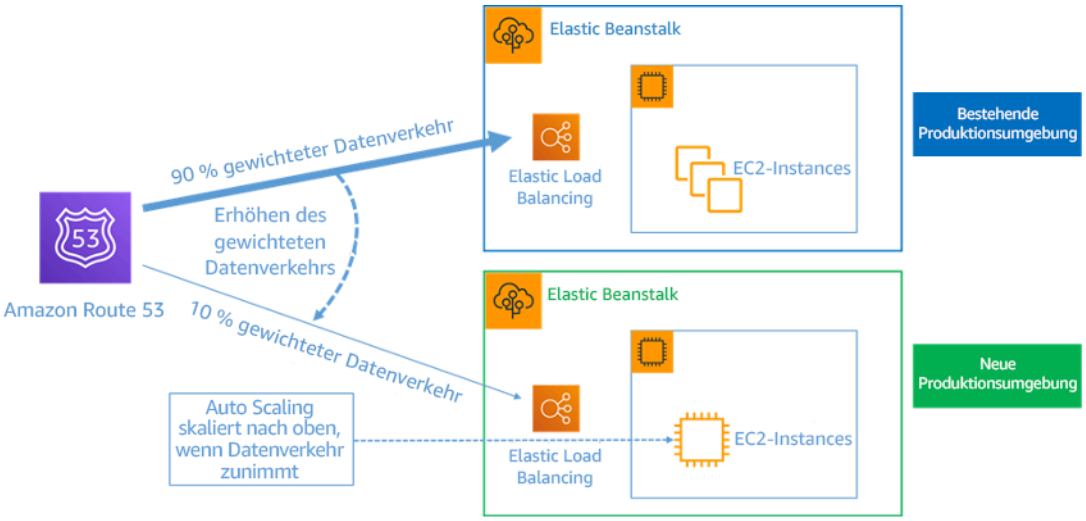


Abbildung 8: Blau/Grün-Bereitstellung mit AWS Elastic Beanstalk und Amazon Route 53

Abweichungserkennung

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 (Einführung in DevOps in 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 verfassen Ihre Ressourcen in JSON oder YAML mit einem beliebigen Code-Editor wie AWS Cloud9, 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 sich mit anderen AWS-Services integrieren und ist eine Erweiterung von AWS 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 AWS 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-Bereitstellung als Release
    - Erstellen Sie eine Pipeline mit Canary-Bereitstellungen für Amazon ECS mit AWS App Mesh
  - Blau/Grün-Bereitstellungen: Das Whitepaper Blau/Grün-Bereitstellungen in AWS beschreibt Beispieltechniken zur Umsetzung von Blau/Grün-Bereitstellungsstrategien.
- Erkennen Sie Konfigurations- oder Zustandsabweichungen. Weitere Informationen finden Sie unter Erkennen von nicht verwalteten Konfigurationsänderungen an Stacks und Ressourcen.

Ressourcen

Zugehörige bewährte Methoden:

- REL08-BP05 Automatisieren von Änderungen

Zugehörige Dokumente:

- Automating safe, hands-off deployments (Automatisierung sicherer, vollautomatischer Bereitstellungen)
- Nutzung von AWS CloudFormation zur Erstellung einer unveränderlichen Infrastruktur bei Nubank
- Infrastruktur als Code
- Implementieren eines Alarms zur automatischen Erkennung von Abweichungen in AWS CloudFormation-Stacks

Zugehörige Videos:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability (AWS re:Invent 2020: Zuverlässlichkeit, Konsistenz und Vertrauen durch Unveränderlichkeit)

REL08-BP05 Automatisieren von Änderungen

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

Änderungen an Produktionssystemen gehören in vielen Unternehmen 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. Mit AWS CodePipeline können Sie die erforderlichen Schritte für die Freigabe Ihrer Workload verwalten. Dies umfasst einen Bereitstellungsstatus in AWS CodeDeploy, um die Bereitstellung von Anwendungscode für Amazon EC2-Instances, On-Premise-Instances, serverlose Lambda-Funktionen oder Amazon ECS-Services zu automatisieren.

> **Empfehlung**
>
> Obwohl die gängige Meinung vorherrscht, dass es sinnvoll ist, Menschen bei den komplexesten betrieblichen Abläufen in die Vorgänge zu integrieren, empfehlen wir, die komplexesten Abläufe aus genau diesem Grund zu automatisieren.

Gängige Antimuster:

- Manuelles Durchführen von Änderungen.
- Überspringen von Schritten in Ihrer Automatisierung durch Notfallarbeitsabläufe.
- Entspricht nicht Ihren Plänen.

Vorteile der Einführung dieser bewährten Methode: Die Verwendung der Automatisierung zum Bereitstellen aller Änderungen verringert das Risiko menschlicher Fehler und ermöglicht die Durchführung von Tests vor Produktionsänderung, um sicherzustellen, dass Ihre Pläne vollständig sind.

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.
  - Die Amazon Builders' Library: Rollback-Sicherheit bei Bereitstellungen gewährleisten
  - Die Amazon Builders' Library: Schneller mit kontinuierlicher Bereitstellung

- Verwenden Sie AWS CodePipeline oder das vertrauenswürdige Produkt eines Drittanbieters), um Ihre Pipelines zu definieren und auszuführen.
  - Legen Sie fest, dass die Pipeline startet, sobald in Ihrem Code-Repository eine Änderung festgeschrieben wird.
    - Was ist AWS CodePipeline?
  - Verwenden Sie Amazon Simple Notification Service (Amazon SNS) und Amazon Simple Email Service (Amazon SES), um Benachrichtigungen bezüglich Pipeline-Problemen zu senden, oder integrieren Sie diese in ein Team-Chat-Tool wie Amazon Chime.
    - Was ist Amazon Simple Notification Service?
    - Was ist Amazon SES?
    - Was ist Amazon Chime?
    - Automatisieren Sie Chat-Nachrichten mit Webhooks.

Ressourcen

Relevante Dokumente:

- APN-Partner: Partner, die Sie beim Erstellen automatisierter Bereitstellungslösungen unterstützen können
- AWS Marketplace: Produkte zur Automatisierung Ihrer Bereitstellungen
- Automatisieren Sie Chat-Nachrichten mit Webhooks.
- Die Amazon Builders' Library: Rollback-Sicherheit bei Bereitstellungen gewährleisten
- Die Amazon Builders' Library: Schneller mit kontinuierlicher Bereitstellung
- Was ist AWS CodePipeline?
- Was ist CodeDeploy?
- AWS Systems Manager Patch Manager
- Was ist Amazon SES?
- Was ist Amazon Simple Notification Service?

Relevante Videos:

- AWS Summit 2019: CI/CD on AWS (AWS Summit 2019: CI/CD auf 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-Premise-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 Ihren 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 Ihren Workload verhindern können.

Themen

Daten sichern

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

Bewährte Methoden

- 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

Informieren Sie sich über die Backup-Funktionen der vom Workload genutzten Daten-Services und Ressourcen und nutzen Sie diese. 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: Foundational

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 mit AWS Storage Gateway oder AWS DataSync in der AWS Cloud gesichert werden. Mit Amazon S3-Buckets können Sie diese Daten auf speichern. Amazon S3 bietet mehrere Speicherebenen wie Amazon S3 Glacier oder S3 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 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. https://aws.amazon.com/disaster-recovery/ übernimmt die automatisierte sekundengenaue Replikation von Daten in einer . 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.

Grad des Aufwands für den Implementierungsplan: moderat.

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
- Backups von Amazon EFS
- Backups von Amazon FSx für Windows-Dateiserver
- Backup und Wiederherstellung für ElastiCache for Redis
- Erstellen eines DB-Cluster-Snapshots in Neptune
- Erstellen eines DB-Snapshots
- Erstellen einer EventBridge-Regel, die nach einem Zeitplan ausgelöst wird

- Regionsübergreifende Replikation mit Amazon S3
- EFS-zu-EFS AWS Backup
- Exportieren von Protokolldaten zu Amazon S3
- Verwaltung des Objektlebenszyklus
- On-Demand-Sicherung und Wiederherstellung in DynamoDB
- Zeitpunktbezogene Wiederherstellung für DynamoDB
- Mit Amazon OpenSearch Service Index-Snapshots arbeiten
- Was ist AWS Elastic Disaster Recovery?

Zugehörige Videos:

- AWS re: Ivent 2021 – Backup, disaster recovery, and ransomware protection with AWS (AWS re:Invent 2021 – Backup, Notfallwiederherstellung und Ransomware-Schutz mit AWS)
- AWS Backup Demo: Cross-Account and Cross-Region Backup (AWS Backup Demo: Konto- und regionsübergreifendes Backup)
- AWS re:Invent 2019: Ausführliche Beschreibung von AWS Backup, mit Rackspace (STG341)

Zugehörige Beispiele:

- Well-Architected Lab: Implementieren einer bidirektionalen Cross-Region Replication (CRR, regionsübergreifende Replikation) für Amazon S3
- Well-Architected Lab: Testen von Backup und Wiederherstellung von Daten
- Well-Architected Lab: Backup and Restore with Failback for Analytics Workload (Well-Architected Lab: Backups und Wiederherstellung mit Failback für Analytics-Workload)
- Well-Architected Lab: Notfallwiederherstellung – Backup und Wiederherstellung

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.

Typische Anti-Muster:

- Derselbe Zugriff auf die Sicherungen und die automatisierte Wiederherstellung wie auf die Daten.

- Keine Verschlüsselung der Sicherungen.

Vorteile der Nutzung dieser bewährten Methode: Die Absicherung Ihrer Backups verhindert die Manipulation der Daten und die Verschlüsselung der Daten verhindert den Zugriff auf diese Daten, wenn sie versehentlich offengelegt werden.

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 gespeicherte Daten. 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 sind 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.

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.

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 AWS KMS gespeichert wurden.
- DynamoDB-Verschlüsselung im Ruhezustand
- Verschlüsseln von Amazon RDS-Ressourcen
- Encrypting Data and Metadata in Amazon EFS (Verschlüsseln von Daten und Metadaten in Amazon EFS)
- Verschlüsselung für Backups in AWS
- Verwalten verschlüsselter Tabellen

- Sicherheitssäule – AWS Well-Architected Framework
- Was ist AWS Elastic Disaster Recovery?

Zugehörige Beispiele:

- Well-Architected Lab: Implementieren einer bidirektionalen Cross-Region Replication (CRR, regionsübergreifende Replikation) für Amazon S3

REL09-BP03 Automatische Daten-Backups

Sie können die Backups so konfigurieren, dass sie automatisch nach Zeitplan, der auf dem Recovery Point Objective (RPO) basiert, oder bei Änderungen am Datensatz durchgeführt werden. Kritische Datasets, 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 (for 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. Amazon S3 Sobald die Versionsverwaltung aktiviert ist, erfolgt sie 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 vor Ort mithilfe des AWS Storage Gateway.

Zusätzlich zum Versioning 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 finden 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 finden 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 (Testen von Backup und Wiederherstellung von Daten). 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 auszuführen.

Grad des Aufwands für den Implementierungsplan: niedrig

Ressourcen

Zugehörige Dokumente:

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

Zugehörige Videos:

- AWS re:Invent 2019: Ausführliche Beschreibung von AWS Backup, mit Rackspace (STG341)

Zugehörige Beispiele:

- Well-Architected Lab: Testen von Backup und Wiederherstellung von Daten

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

Überprüfen Sie mit einem Wiederherstellungstest, ob sich mit Ihren Sicherungsverfahren das RTO und das RPO einhalten lassen.

Angestrebtes Ergebnis: Daten aus Backups werden regelmäßig mit genau definierten Mechanismen wiederhergestellt, um zu überprüfen, ob eine Wiederherstellung innerhalb des festgelegten Recovery Time Objectives (RTO) für den 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 den 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: Das Testen der Wiederherstellung der Backups stellt sicher, dass die Daten bei Bedarf wiederhergestellt werden können, ohne dass Sie sich Sorgen um fehlende oder beschädigte Daten machen müssen, dass die Wiederherstellung innerhalb des RTOs für den Workload möglich ist und dass jeder Datenverlust innerhalb des RPOs für den Workload liegt.

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 den 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 Point-in-Time-Wiederherstellung. 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 RPOs für den 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 die Datenquellen, von denen derzeit ein Backup erstellt wird, und wo diese Backups gespeichert werden. Eine Anleitung zur Implementierung finden Sie unter REL09-BP01 Ermitteln und Sichern aller zu sichernden Daten oder Reproduzieren der Daten aus Quellen.
2. Etablieren von Kriterien zur Datenvalidierung für jede Datenquelle. 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. Etablieren des RTO und RPO für die Wiederherstellung der Daten basierend auf der Wichtigkeit der Daten. Eine Anleitung zur Implementierung finden Sie unter REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten:.
4. Bewerten Sie die Funktion zur Datenwiederherstellung. Prüfen Sie Ihre Sicherungs- und Wiederherstellungsstrategie, um festzustellen, ob sie Ihre RTO und RPO erfüllen kann, und passen Sie die Strategie bei Bedarf an. Mit dem AWS Resilience Hub können Sie eine Bewertung Ihres Workloads vornehmen. 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 durch, indem Sie die derzeit in der Produktion für die Wiederherstellung von Daten verwendeten Prozesse verwenden. 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 z. B. einen verwalteten Service wie AWS Backup verwenden, reicht es vielleicht aus, das Backup in einer neuen Ressource wiederherzustellen. Wenn Sie AWS Elastic Disaster Recovery verwendet haben, können Sie einen Recovery-Drill starten.
6. Validieren Sie die Datenwiederherstellung aus der wiederhergestellten Ressource anhand der Kriterien, die Sie zuvor für die Validierung der Daten 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 die benötigte Zeit für die Wiederherstellung und vergleichen Sie sie 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 haben einen Zeitstempel. Sie finden diese Informationen in AWS CloudTrail. 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 Sie Services wie Amazon DynamoDB nutzen, 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. Innerhalb von AWS Backup werden Backup- und Wiederherstellungsaktionen als Jobs bezeichnet. Diese Jobs

enthalten als Teil ihrer Metadaten Zeitstempelinformationen, die zur Messung der für die Wiederherstellung benötigten Zeit verwendet werden können.

8. Benachrichtigen Sie die Stakeholder, wenn die Validierung der Daten fehlschlägt oder wenn die für die Wiederherstellung benötigte Zeit den festgelegten RTO für den Workload überschreitet. Bei der Implementierung einer entsprechenden Automatisierung, wie in dieser Übung, können Services wie Amazon Simple Notification Service (Amazon SNS) genutzt werden, um Push-Benachrichtigungen wie E-Mails oder SMS an Stakeholder zu senden. Diese Nachrichten können auch in Messaging-Anwendungen wie Amazon Chime, Slack oder Microsoft Teams veröffentlicht werden. Sie können zudem verwendet werden, um Aufgaben als OpsItems mit AWS Systems Manager OpsCenter zu erstellen.
9. Lassen Sie diesen Prozess regelmäßig automatisch ausführen. 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 auszulösen, wie im folgenden Architekturdiagramm abgebildet. Informieren Sie sich darüber, wie Sie die Validierung der Datenwiederherstellung mit AWS Backup automatisieren. Darüber hinaus bietet diese Well-Architected-Übung eine praxisorientierte Anleitung zur Automatisierung mehrerer der hier beschriebenen Schritte.

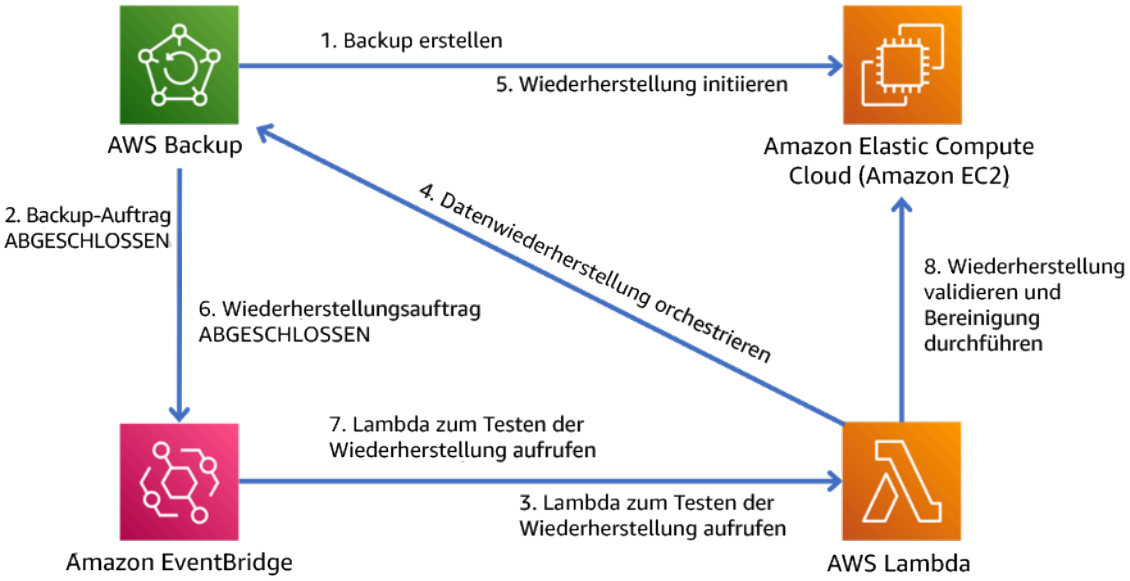


Abbildung 9. Ein automatisierter Sicherungs- und Wiederherstellungsprozess

Aufwandsniveau für den Implementierungsplan: Mäßig bis hoch, abhängig von der Komplexität der Validierungskriterien.

Ressourcen

Zugehörige Dokumente:

- Automatisieren der Datenwiederherstellung mit AWS Backup
- 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-Sicherung und Wiederherstellung in DynamoDB
- Was ist AWS Backup?
- Was ist AWS Step Functions?
- Was ist AWS Elastic Disaster Recovery?
- AWS Elastic Disaster Recovery

Zugehörige Beispiele:

- Well-Architected Lab: Testen von Backup und Wiederherstellung von Daten

Schützen von Workloads durch Fehlerisolierung

Fehlerisolierte Grenzen beschränken die Auswirkungen eines Ausfalls innerhalb eines Workloads auf eine begrenzte Anzahl von Komponenten. Komponenten außerhalb der Grenze sind vom Ausfall nicht betroffen. Wenn Sie mehrere fehlerisolierte Grenzen verwenden, können Sie die Auswirkungen auf Ihren Workload einschränken.

Bewährte Methoden

- REL10-BP01 Bereitstellen des Workloads an mehreren Standorten
- REL10-BP02 Auswählen der geeigneten Standorte für Ihre Multi-Standort-Bereitstellung
- 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. Die Standorte können so vielfältig wie nötig sein.

Eins der grundlegenden Prinzipien für das Servicedesign in AWS ist die Vermeidung von Single Points of Failure in der zugrunde liegenden physischen Infrastruktur. Dies treibt uns an, Software und Systeme zu entwickeln, die mehrere Availability Zones verwenden und Schutz beim Ausfall einer einzelnen Region bieten. Außerdem sollen Systeme gegen den Ausfall einzelner Compute-Knoten, einzelner Speicher-Volumes oder einzelner Instances einer Datenbank geschützt sein. Bei der Entwicklung eines Systems, das auf redundanten Komponenten basiert, muss gewährleistet sein, dass die Komponenten unabhängig voneinander betrieben werden und im Falle von AWS-Regionen autonom sind. Die Vorteile theoretischer Verfügbarkeitsberechnungen mit redundanten Komponenten sind nur anwendbar, wenn diese Voraussetzung erfüllt ist.

Availability Zones (AZs)

AWS-Regionen bestehen aus mehreren voneinander unabhängigen Availability Zones. Die einzelnen Availability Zones sind durch eine signifikante physische Distanz voneinander getrennt, um korrelierte Fehlerszenarios aufgrund von Umweltgefahren wie Feuer, Überflutungen und Tornados zu vermeiden. Jede Availability Zone verfügt außerdem über eine unabhängige physische Infrastruktur: eigene Verbindungen zur Stromversorgung, unabhängige Backup-Stromquellen, unabhängige mechanischen Services und unabhängige Netzwerkkonnektivität innerhalb der Availability Zone und darüber hinaus. Durch dieses Design bleiben Fehler in einem dieser Systeme auf die jeweils betroffene AZ beschränkt. Trotz ihrer geografischen Verteilung befinden sich Availability Zones in demselben regionalen Bereich, wodurch Netzwerke mit hohem Durchsatz und geringer Latenz ermöglicht werden. Die gesamte AWS-Region (über alle Availability Zones, die aus mehreren physisch unabhängigen Rechenzentren bestehen) kann wie ein logisches Bereitstellungsziel für Ihren Workload behandelt werden. Dies umfasst auch die Möglichkeit zum synchronen Replizieren von Daten (z. B. zwischen Datenbanken). So können Sie Availability Zones in einer Aktiv-Aktiv- oder einer Aktiv-Standby-Konfiguration nutzen.

Availability Zones sind voneinander unabhängig. Daher erhöht sich die Workload-Verfügbarkeit, wenn in der Architektur des Workloads mehrere Zonen verwendet werden. Einige AWS-Services (darunter auch die Amazon EC2-Instance-Datenebene) werden als strikte zonale Services bereitgestellt, die von denselben Fehlern betroffen sind wie die Availability Zone, in der sie sich befinden. Amazon EC2-Instances in den anderen AZs sind hingegen nicht betroffen und weiterhin funktionsfähig. Wenn entsprechend ein Fehler in einer Availability Zone zum Ausfall einer Amazon Aurora-Datenbank

führt, kann eine Auslese-Replikat-Aurora-Instance in einer nicht betroffenen AZ automatisch zur primären Instance hochgestuft werden. Regionale AWS-Services wie Amazon DynamoDB wiederum verwenden intern mehrere Availability Zones in einer Aktiv-Aktiv-Konfiguration, um die Verfügbarkeitsdesignziele für den jeweiligen Service zu erfüllen, ohne dass Sie die AZ-Platzierung konfigurieren müssen.

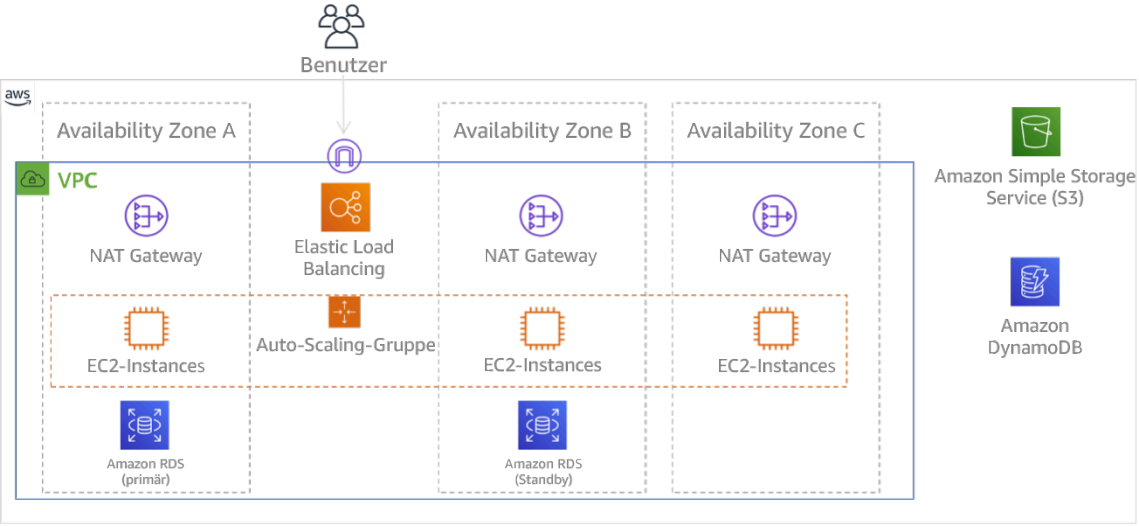


Abbildung 9: Mehrstufige Architektur, die in drei Availability Zones bereitgestellt wird. Amazon S3 und Amazon DynamoDB nutzen immer automatisch mehrere AZs. Auch der ELB wird in allen drei Zonen bereitgestellt.

Während Amazon EBS-Steuerebenen in der Regel die Möglichkeit bieten, Ressourcen innerhalb der gesamten Region (also in mehreren Availability Zones) zu verwalten, haben bestimmte Steuerebenen (wie AWS und Amazon EC2) die Fähigkeit, Ergebnisse in eine einzelne Availability Zone zu filtern. Wenn dies erledigt ist, wird die Anfrage nur in der angegebenen Availability Zone verarbeitet; dies reduziert die Wahrscheinlichkeit von Ausfällen in anderen Availability Zones. Dieses AWS CLI-Beispiel veranschaulicht das Abrufen von Amazon EC2-Instance-Informationen ausschließlich aus der Availability Zone „us-east-2c“:

```
AWS ec2 describe-instances --filters Name=availability-zone,Values=us-east-2c
```

AWS Local Zones

AWS Local Zones verhalten sich ähnlich wie Availability Zones innerhalb ihrer jeweiligen AWS-Region. Sie können als Platzierungsstandort für zonale AWS-Ressourcen wie Subnetze und EC2-Instances ausgewählt werden. Das Besondere daran ist, dass sie sich nicht in der zugehörigen AWS-Region befinden, sondern in der Nähe großer Ballungsräume, Industrie- und IT-Zentren, in denen

derzeit keine AWS-Region vorhanden ist. Sie sorgen dennoch für eine sichere Verbindung mit hoher Bandbreite zwischen lokalen Workloads in der lokalen Zone und Workloads in der AWS-Region. Sie sollten AWS Local Zones verwenden, um Workloads mit Anforderungen an eine geringe Latenz näher bei Ihren Benutzern bereitzustellen.

Amazon Global Edge Network

Amazon Global Edge Network besteht aus Edge-Standorten in Städten auf der ganzen Welt. Amazon CloudFront nutzt dieses Netzwerk, um Inhalte mit geringerer Latenz für Endbenutzer bereitzustellen. Mit AWS Global Accelerator können Sie Ihre Workload-Endpunkte an diesen Edge-Standorten erstellen, um ein Onboarding in das globale AWS-Netzwerk in der Nähe Ihrer Benutzer zu ermöglichen. Amazon API Gateway können Sie Edge-optimierte API-Endpunkte mithilfe einer CloudFront-Verteilung verwenden, um den Client-Zugriff über den nächstgelegenen Edge-Standort zu erleichtern.

AWS-Regionen

AWS-Regionen sind autonom konzipiert. Daher können Sie dedizierte Kopien von Services für jede Region bereitstellen, um einen multiregionalen Ansatz zu verwenden.

Ein multiregionaler Ansatz wird häufig für Strategien der Notfallwiederherstellung eingesetzt, um Wiederherstellungsziele zu erfüllen, falls einmalige Ereignisse mit großer Reichweite auftreten. Siehe Planung der Notfallwiederherstellung für weitere Informationen zu diesen Strategien. Hier liegt der Schwerpunkt allerdings auf der Verfügbarkeit, wobei versucht wird, ein mittleres Betriebszeitziel über einen längeren Zeitraum zu erreichen. Wenn eine hohe Verfügbarkeit angestrebt wird, ist eine multiregionale Architektur normalerweise Aktiv-Aktiv konzipiert. Dabei sind die einzelnen Servicekopien (in den jeweiligen Regionen) aktiv (und bearbeiten Anfragen).

> **Empfehlung**
>
> Die Verfügbarkeitsziele für die meisten Workloads können mithilfe einer Multi-AZ-Strategie innerhalb einer einzelnen AWS-Region erfüllt werden. Ziehen Sie multiregionale Architekturen nur in Betracht, wenn für Workloads extreme Verfügbarkeitsanforderungen gelten oder andere Unternehmensziele eine solche Architektur erforderlich machen.

AWS bietet Ihnen die Möglichkeit, Services regionsübergreifend zu betreiben. AWS stellt beispielsweise eine fortlaufende asynchrone Datenreplikation mit Amazon S3-Replikation (Amazon Simple Storage Service), Amazon RDS-Lesereplikaten (u. a. Aurora-Lesereplikaten) und globalen

Amazon DynamoDB-Tabellen bereit. Bei der fortlaufenden Replikation sind Versionen Ihrer Daten für die fast sofortige Nutzung in jeder aktiven Region verfügbar.

Mit AWS CloudFormation können Sie Ihre Infrastruktur definieren und einheitlich in AWS-Konten und AWS-Regionen bereitstellen. AWS CloudFormation StackSets erweitern diese Funktionen, indem Sie AWS CloudFormation-Stacks mit nur einem Vorgang in verschiedenen Konten und Regionen erstellen, aktualisieren oder löschen können. Bei Amazon EC2-Instance-Bereitstellungen wird ein AMI (Amazon Machine Image) verwendet, um Informationen wie die Hardwarekonfiguration und installierte Software bereitzustellen. Sie können eine Amazon EC2 Image Builder-Pipeline implementieren, die die benötigten AMIs erstellt, und diese in Ihre aktiven Regionen kopieren. Diese goldenen AMIs enthalten alles, was Sie zum Bereitstellen und Skalieren von Workloads in neuen Regionen benötigen.

Zum Weiterleiten von Datenverkehr ermöglichen sowohl Amazon Route 53 als auch AWS Global Accelerator das Definieren von Richtlinien, die angeben, welche Benutzer zu welchem aktiven regionalen Endpunkt geleitet werden. Mit Global Accelerator legen Sie für den Datenverkehr einen Prozentwert fest, der an die einzelnen Anwendungsendpunkte geleitet wird. Route 53 unterstützt diesen Ansatz mit Prozentwerten sowie eine Vielzahl weiterer Richtlinien, u. a. auf Grundlage der geografischen Nähe oder der Latenz. Global Accelerator nutzt automatisch das umfassende Netzwerk von AWS-Edge-Servern, um Datenverkehr an den Backbone des AWS-Netzwerks zu senden, sobald dies möglich ist. Dies führt zu einer geringeren Latenz bei Abfragen.

Alle diese Funktionen sind so konzipiert, dass die Autonomie der einzelnen Regionen erhalten wird. Es gibt nur sehr wenige Ausnahmen von diesem Ansatz, darunter unsere Services für eine weltweite Edge-Lieferung (z. B. Amazon CloudFront und Amazon Route 53) und die Steuerebene für den AWS Identity and Access Management-Service (IAM). Die meisten Services werden vollständig innerhalb einer einzigen Region betrieben.

On-Premises-Rechenzentrum

Für Workloads, die in einem On-Premises-Rechenzentrum ausgeführt werden, sollten Sie nach Möglichkeit eine hybride Umgebung erstellen. AWS Direct Connect bietet eine dedizierte Netzwerkverbindung zwischen Ihrem Standort und AWS, sodass eine Ausführung in beiden Umgebungen möglich ist.

Außerdem haben Sie die Möglichkeit, AWS-Infrastruktur und -Services mit AWS Outposts lokal auszuführen. AWS Outposts ist ein vollständig verwalteter Service, der die AWS-Infrastruktur, AWS-Services, APIs und Tools auf Ihr Rechenzentrum erweitert. Die gleiche Hardwareinfrastruktur, die in der AWS Cloud verwendet wird, wird dafür in Ihrem Rechenzentrum installiert. AWS Outposts werden

dann mit der nächstgelegenen AWS-Region verbunden. Anschließend können Sie AWS Outposts verwenden, um Workloads mit geringer Latenz oder lokalen Datenverarbeitungsanforderungen zu unterstützen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Verwenden Sie mehrere Availability Zones und AWS-Regionen. Verteilen Sie die Workload-Daten und -Ressourcen über mehrere Availability Zones oder ggf. über mehrere AWS-Regionen. Die Standorte können so vielfältig wie nötig sein.
  - Regionale Services werden von Haus aus in Availability Zones bereitgestellt.
    - Dazu gehören Amazon S3, Amazon DynamoDB und AWS Lambda (wenn keine VPC-Verbindung vorhanden ist).
  - Stellen Sie Ihre Container-, Instance- und funktionsbasierten Workloads in mehreren Availability Zones bereit. Verwenden Sie Multi-AZ-Datenspeicher, einschließlich Cache. Nutzen Sie EC2 Auto Scaling, die ECS-Aufgabenplatzierung, ElastiCache-Cluster sowie bei Ausführung in Ihrer VPC AWS Lambda-Funktionen.
    - Verwenden Sie für die Bereitstellung von Auto-Scaling-Gruppen Subnetze in getrennten Availability Zones.
      - Beispiel: Verteilen von Instances in Availability Zones
      - Strategien zur Aufgabenplatzierung mit Amazon ECS
      - Konfigurieren einer AWS Lambda-Funktion für den Zugriff auf Ressourcen in einer Amazon VPC
      - Auswählen von Regionen und Availability Zones
    - Verwenden Sie für die Bereitstellung von Auto-Scaling-Gruppen Subnetze in getrennten Availability Zones.
      - Beispiel: Verteilen von Instances in Availability Zones
    - Verwenden Sie ECS-Parameter für die Platzierung von Aufgaben unter Angabe von DB-Subnetzgruppen.
      - Strategien zur Aufgabenplatzierung mit Amazon ECS
    - Nutzen Sie Subnetze in mehreren Availability Zones, wenn Sie eine in Ihrem VPC auszuführende Funktion konfigurieren.
      - Konfigurieren einer AWS Lambda-Funktion für den Zugriff auf Ressourcen in einer Amazon VPC

    - Verwenden Sie mehrere Availability Zones mit ElastiCache-Clustern.
        - Auswählen von Regionen und Availability Zones
- Wenn Ihr Workload für mehrere Regionen bereitgestellt werden muss, sollten Sie sich für eine Strategie mit mehreren Regionen entscheiden. Die meisten Zuverlässigkeitsanforderungen können mithilfe einer Multi-Availability-Zone-Strategie innerhalb einer einzelnen AWS-Region erfüllt werden. Verwenden Sie eine Multi-Regionen-Strategie, wenn notwendig, um Ihre Geschäftsanforderungen zu erfüllen.
    - AWS re:Invent 2018: Architekturmuster für Aktiv-Aktiv-Anwendungen in mehreren Regionen (ARC209-R2)
        - Ein Backup in einer anderen AWS-Region kann zusätzliche Gewissheit bieten, dass Daten verfügbar sind, wenn sie benötigt werden.
        - Für einige Workloads gibt es gesetzliche Anforderungen, die eine Multi-Region-Strategie erfordern.
- Evaluieren Sie AWS Outposts für Ihren Workload. Wenn Ihre Workload eine niedrige Latenz für Ihr Rechenzentrum vor Ort erfordert oder lokale Datenverarbeitungsanforderungen hat. Führen Sie anschließend AWS-Infrastruktur und -Services On-Premises mit AWS Outposts aus.
    - Was ist AWS Outposts?
- Ermitteln Sie, ob AWS Local Zones Sie bei der Bereitstellung von Services für Ihre Benutzer unterstützt. Wenn Sie Anforderungen an eine geringe Latenz haben, prüfen Sie, ob sich AWS Local Zones in der Nähe Ihrer Benutzer befindet. Wenn dies der Fall ist, stellen Sie damit Workloads näher an diesen Benutzern bereit.
    - AWS Local Zones – häufig gestellte Fragen

Ressourcen

Ähnliche Dokumente:

- Globale AWS-Infrastruktur
- AWS Local Zones – häufig gestellte Fragen
- Strategien zur Aufgabenplatzierung mit Amazon ECS
- Auswählen von Regionen und Availability Zones
- Beispiel: Verteilen von Instances in Availability Zones
- Globale Tabellen: Multiregionale Replikation mit DynamoDB
- Verwenden von Amazon Aurora Global Databases

- Blog-Reihe: Creating a Multi-Region Application with AWS Services (Erstellen einer Multi-Region-Anwendung mit AWS-Services)
- Was ist AWS Outposts?

Relevante Videos:

- AWS re:Invent 2018: Architekturmuster für Aktiv-Aktiv-Anwendungen in mehreren Regionen (ARC209-R2)
- AWS re:Invent 2019: Innovation und Betrieb der globalen Netzwerkinfrastruktur von AWS (NET339)

REL10-BP02 Auswählen der geeigneten Standorte für Ihre Multi-Standort-Bereitstellung

Gewünschtes Ergebnis

Für eine hohe Verfügbarkeit stellen Sie Ihre Workload-Komponenten (falls möglich) immer in mehreren Availability Zone (AZ) bereit, wie in Abbildung 10 dargestellt. Überdenken Sie bei Workloads mit extremen Anforderungen an die Ausfallsicherheit die Optionen für eine Multi-Region-Architektur genau.

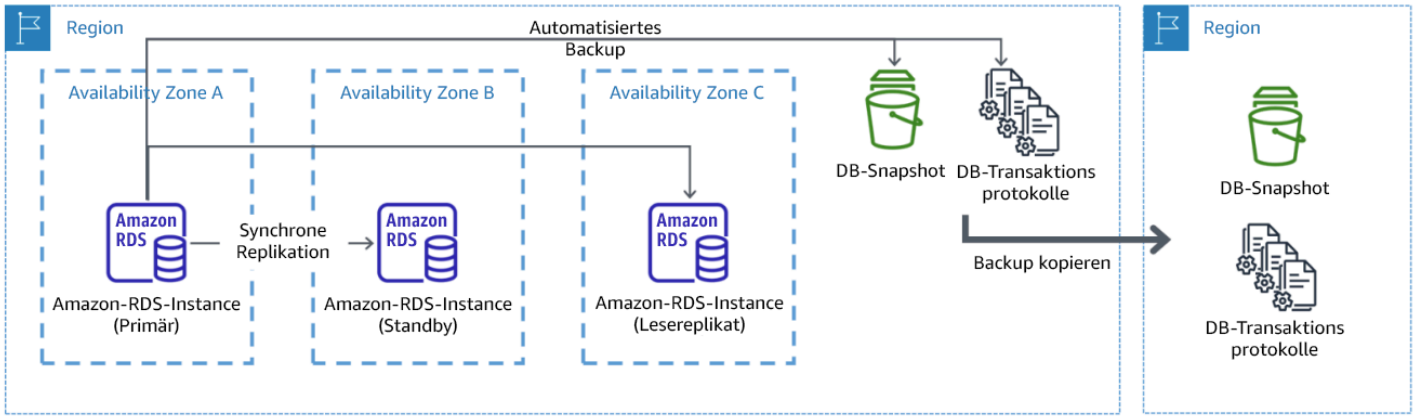


Abbildung 10: Resiliente Multi-AZ-Datenbankbereitstellung mit Backup in einer anderen AWS-Region

Gängige Antimuster

- Entscheidung für das Design einer Multi-Region-Architektur, wenn eine Multi-AZ-Architektur für die Anforderungen ausreichend wäre.

- Fehlende Berücksichtigung der Abhängigkeiten zwischen Anwendungskomponenten, wenn diese Komponenten unterschiedliche Anforderungen im Bezug auf Ausfallsicherheit und mehrere Standorte aufweisen.

Vorteile der Einführung dieser bewährten Methode:

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.

- Der Unterschied zwischen einer Verfügbarkeit von 99,5 % und 99,99 % beträgt über 3,5 Stunden pro Monat. Die erwartete Verfügbarkeit eines Workloads kann nur „four nines“ (d. h. 99,99 %) erreichen, wenn er sich in mehreren AZs befindet.
- Indem Sie einen Workload in mehreren AZs ausführen, können Sie Fehler bei der Stromversorgung, Kühlung, im Netzwerk sowie die meisten Naturkatastrophen wie Feuer und Überflutung isolieren.
- Wenn Sie eine Multi-Region-Strategie für Ihren Workload implementieren, ist er 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.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Bei einer Unterbrechung oder dem teilweisen Ausfall einer Availability Zone hilft die Implementierung eines hoch verfügbaren Workloads in mehreren Availability Zones innerhalb einer einzelnen AWS-Region, die Folgen von Naturkatastrophen oder technischen Problemen zu begrenzen. Jede AWS-Region besteht aus mehreren Availability Zones, die von Fehlern in den jeweils anderen Zonen isoliert sind und die eine deutliche Distanz aufweisen. In Bezug auf Notfallereignisse, bei denen das Risiko des Ausfalls mehrerer, voneinander weit entfernter Availability-Zone-Komponenten besteht, sollten Sie Optionen für die Notfallwiederherstellung implementieren. So können Sie Fehler eingrenzen, die sich auf eine ganze Region auswirken. Bei Workloads, für die eine extreme

Ausfallsicherheit erforderlich ist (kritische Infrastruktur, gesundheitsbezogene Anwendungen, Infrastruktur von Finanzsystemen usw.) wird möglicherweise eine Multi-Region-Strategie benötigt.

Implementierungsschritte

1. Analysieren Sie Ihren Workload und bestimmen Sie, ob die Anforderungen an die Ausfallsicherheit mit einem Multi-AZ-Ansatz erfüllt werden (eine AWS-Region) oder ob ein Multi-Region-Ansatz erforderlich ist. Das Implementieren einer Multi-Region-Architektur, um diese Anforderungen zu erfüllen, führt zu einer höheren Komplexität. Betrachten Sie daher Ihren Anwendungsfall und wägen Sie die Anforderungen sorgfältig ab. Die Anforderungen an die Ausfallsicherheit können fast immer auch mit einer AWS-Region erfüllt werden. Berücksichtigen Sie bei der Entscheidung, ob Sie mehrere Regionen verwenden möchten, die folgenden möglichen Anforderungen:
   a. Notfallwiederherstellung (Disaster Recovery, DR): Bei einer Unterbrechung oder dem teilweisen Ausfall einer Availability Zone hilft die Implementierung eines hoch verfügbaren Workloads in mehreren Availability Zones innerhalb einer einzelnen AWS-Region, die Folgen von Naturkatastrophen oder technischen Problemen zu begrenzen. In Bezug auf Notfallereignisse, bei denen das Risiko des Ausfalls mehrerer, voneinander weit entfernter Availability Zone-Komponenten besteht, sollten Sie eine Notfallwiederherstellung in mehreren Regionen implementieren. So können Sie die Risiken durch Naturkatastrophen oder technische Fehler eingrenzen, die sich auf eine ganze Region auswirken.
   b. Hohe Verfügbarkeit (High Availability, HA): Mit einer Multi-Region-Architektur (mit mehreren AZs in jeder Region) kann eine höhere Verfügbarkeit als „four 9’s“ (> 99,99 %) erreicht werden.
   c. Stack-Lokalisierung: Beim Bereitstellen eines Workloads für Benutzer weltweit können Sie lokalisierte Stacks in verschiedenen AWS-Regionen bereitstellen, um die Benutzer in diesen Regionen zu versorgen. Die Lokalisierung kann Sprache, Währung und die gespeicherten Datentypen umfassen.
   d. Nähe zu den Benutzern: Wenn Sie einen Workload für Benutzer weltweit bereitstellen, können Sie die Latenz reduzieren, indem Sie Stacks in AWS-Regionen in der Nähe der Endbenutzer bereitstellen.
   e. Datenresidenz: Für einige Workloads gelten Anforderungen an die Datenresidenz, d. h. die Daten von bestimmten Nutzern müssen innerhalb der Grenzen eines bestimmten Landes gespeichert werden. Abhängig von der jeweiligen Regelung können Sie einen ganzen Stack oder nur die Daten in der AWS-Region innerhalb dieser Landesgrenzen bereitstellen.
2. Im Folgenden finden Sie einige Bespiele für Multi-AZ-Funktionen, die von AWS-Services bereitgestellt werden:

a. Um Workloads mit EC2 oder ECS zu schützen, stellen Sie einen Elastic Load Balancer vor den Datenverarbeitungsressourcen bereit. Elastic Load Balancing bietet so die Lösung, um Instances in fehlerhaften Zonen zu erkennen und den Datenverkehr zu fehlerfreien Zonen zu leiten.

   i. Erste Schritte mit Application Load Balancers

   ii. Erste Schritte mit Network Load Balancers

b. Bei EC2-Instances, auf denen kommerzielle Standardsoftware ohne Unterstützung für Load Balancing ausgeführt wird, können Sie eine gewisse Fehlertoleranz durch die Implementierung einer Methodologie für die Multi-AZ-Notfallwiederherstellung erreichen.

   i. the section called “REL13-BP02: Verwenden von definierten Wiederherstellungsstrategien, um die Wiederherstellungsziele zu erreichen”

c. Stellen Sie für Amazon ECS-Aufgaben den Service gleichmäßig auf drei AZs verteilt bereit, um eine ausgeglichene Verteilung von Verfügbarkeit und Kosten zu erreichen.

   i. Bewährte Methoden für die Amazon ECS-Verfügbarkeit | Container

d. Wenn Sie nicht mit Aurora Amazon RDS arbeiten, können Sie Multi-AZ als Konfigurationsoption auswählen. Beim Ausfall der primären Datenbank-Instance stuft Amazon RDS automatisch eine Standby-Datenbank hoch, sodass sie Datenverkehr in einer anderen Availability Zone empfangen kann. Außerdem können Multi-Region-Lesereplikate erstellt werden, um die Ausfallsicherheit zu steigern.

   i. Amazon RDS-Multi-AZ-Bereitstellungen

   ii. Erstellen eines Lesereplikats in einer anderen AWS-Region

3. Im Folgenden finden Sie einige Bespiele für Multi-Region-Funktionen, die von AWS-Services bereitgestellt werden:

   a. Für Amazon S3-Workloads, bei denen Multi-AZ-Verfügbarkeit automatisch vom Service bereitgestellt wird, erwägen Sie Multi-Region-Zugriffspunkte, wenn eine Multi-Region-Bereitstellung benötigt wird.

      i. Multi-Region-Zugriffspunkte in Amazon S3

   b. Wenn bei DynamoDB-Tabellen Multi-AZ-Verfügbarkeit automatisch vom Service bereitgestellt wird, können Sie vorhandene Tabellen problemlos in globale Tabellen konvertieren, um mehrere Regionen nutzen zu können.

      i. Konvertieren von Amazon DynamoDB-Tabellen für eine Region in globale Tabellen

c. Wenn Ihr Workload hinter Application Load Balancers oder Network Load Balancers liegt, verwenden Sie AWS Global Accelerator, um die Verfügbarkeit Ihrer Anwendung zu verbessern, indem Sie Datenverkehr zu mehreren Regionen mit fehlerfreien Endpunkten leiten.
   i. Endpunkte für Standard-Accelerators in AWS Global Accelerator – AWS Global Accelerator (amazon.com)
d. Erwägen Sie bei Anwendungen, die AWS EventBridge nutzen, die Verwendung von regionsübergreifenden Buses, um Ereignisse an ausgewählte Regionen weiterzuleiten.
   i. Senden und Empfangen von Amazon EventBridge-Ereignissen zwischen AWS-Regionen
e. Ziehen Sie bei Amazon Aurora-Datenbanken globale Aurora-Datenbanken in Erwägungen, die mehrere AWS-Regionen umfassen können. Vorhandene Cluster können ebenfalls geändert werden, um neue Regionen hinzuzufügen.
   i. Erste Schritte mit globalen Amazon Aurora-Datenbanken
f. Wenn Ihr Workload AWS Key Management Service-Verschlüsselungsschlüssel (AWS KMS) umfasst, überlegen Sie, ob Multi-Region-Schlüssel für Ihre Anwendung geeignet sind.
   i. Multi-Region-Schlüssel in AWS KMS
g. Weitere Funktionen von AWS-Services finden Sie in dieser Blog-Reihe zum Erstellen einer Multi-Region-Anwendung mit AWS-Services

Grad des Aufwands für den Implementierungsplan: Mittel bis hoch

Ressourcen

Ähnliche Dokumente:

- Erstellen einer Multi-Region-Anwendung mit AWS-Services
- Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active (Architektur für die Notfallwiederherstellung (Disaster Recovery, DR) in AWS, Teil IV: Multi-Site Aktiv-Aktiv)
- Globale AWS-Infrastruktur
- AWS Local Zones – häufig gestellte Fragen
- Architektur für die Notfallwiederherstellung in AWS, Teil I: Strategien für die Wiederherstellung in der Cloud
- Die Notfallwiederherstellung in der Cloud unterscheidet sich
- Globale Tabellen: Multiregionale Replikation mit DynamoDB

Relevante Videos:

- AWS re:Invent 2018: Architekturmuster für Aktiv-Aktiv-Anwendungen in mehreren Regionen (ARC209-R2)
- Auth0: multiregionale Architektur mit hoher Verfügbarkeit, die auf mehr als 1,5 Milliarden Anmeldungen pro Monat mit automatisiertem Failover skaliert werden kann.

Ähnliche Beispiele:

- Architektur für die Notfallwiederherstellung in AWS, Teil I: Strategien für die Wiederherstellung in der Cloud
- DTCC erzielt Resilienz weit über das hinaus, was On-Premises möglich wäre
- Expedia Group nutzt eine Architektur mit mehreren Regionen und Availability Zones und einem proprietären DNS-Service, um den Anwendungen Resilienz hinzuzufügen.
- Uber: Notfallwiederherstellung für multiregionales Kafka
- Netflix: Aktiv-Aktiv für multiregionale Resilienz
- Entwicklung von Data Residency für Atlassian Cloud
- Intuit TurboTax wird über zwei Regionen ausgeführt

REL10-BP03 Automatisierte Wiederherstellung für Komponenten, die auf einen einzelnen Standort beschränkt sind

Wenn Komponenten des Workloads nur in einer einzigen Availability Zone oder in einem On-Premises-Rechenzentrum ausgeführt werden können, implementieren Sie die Möglichkeit, den Workload innerhalb Ihrer definierten Wiederherstellungsziele komplett neu aufzusetzen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: mittel

Implementierungsleitfaden

Wenn die bewährte Methode zur Bereitstellung des Workloads 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 werden, das wiederum über mehrere Availability Zones oder AWS-Regionen repliziert werden kann.

Ähnlich wie bei Amazon Redshift wird Ihr Cluster standardmäßig in einer zufällig ausgewählten Availability Zone innerhalb der ausgewählten AWS-Region bereitgestellt. Alle Cluster-Knoten 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 Ihren Workload in einer anderen 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 Anforderungen an eine einzelne Instance-IP-Adresse, private IP-Adresse, elastische IP-Adresse und Instance-Metadaten stellen.
     - 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 einzige Instance-IP-Adresse, eine private IP-Adresse, eine elastische IP-Adresse und Instance-Metadaten erfordern.
     - Automatic Recovery sendet Benachrichtigungen zum Wiederherstellungsstatus an ein SNS-Thema, wenn der Instance-Fehler erkannt wird.

- Verwenden Sie Lebenszyklusereignisse von Amazon EC2-Instances oder Amazon ECS-Ereignissen, um das Self-Healing zu automatisieren, wenn eine 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 zustandsbasierte Workloads, die auf einen einzigen Standort beschränkt sind, mit AWS Elastic Disaster Recovery.

Ressourcen

Zugehörige Dokumente:

- Amazon ECS-Ereignisse
- Amazon EC2 Auto Scaling-Lebenszyklus-Hooks
- Stellen Sie Ihre Instance wieder her.
- 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 eines Workloads auf eine begrenzte Anzahl von Komponenten zu beschränken.

Gewünschtes Ergebnis: Eine zellenbasierte Architektur verwendet mehrere isolierte Instances eines Workloads, 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 einem 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 treten viele häufige Fehlerarten innerhalb einer Zelle selbst auf, was eine zusätzliche Fehlerisolierung ermöglicht. Diese Fehlergrenzen bieten Schutz vor Fehlern, die sich sonst nur schwer eindämmen lassen, wie z. B. eine erfolglose Codebereitstellung oder Anfragen, die beschädigt sind oder einen bestimmten Fehlermodus auslösen (Poison Pill Requests).

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 eines Workloads auf eine begrenzte Anzahl von Komponenten. Komponenten außerhalb der Grenze sind vom Ausfall nicht betroffen. Wenn Sie mehrere fehlerisolierte Grenzen verwenden, können Sie die Auswirkungen auf Ihren 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 Ihres Workloads ausweiten.

Der gesamte Workload wird durch einen Partitionsschlüssel in Zellen unterteilt. Dieser Schlüssel muss mit dem Grain des Service übereinstimmen, d. h. mit der logischen Art und Weise, in der der 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.

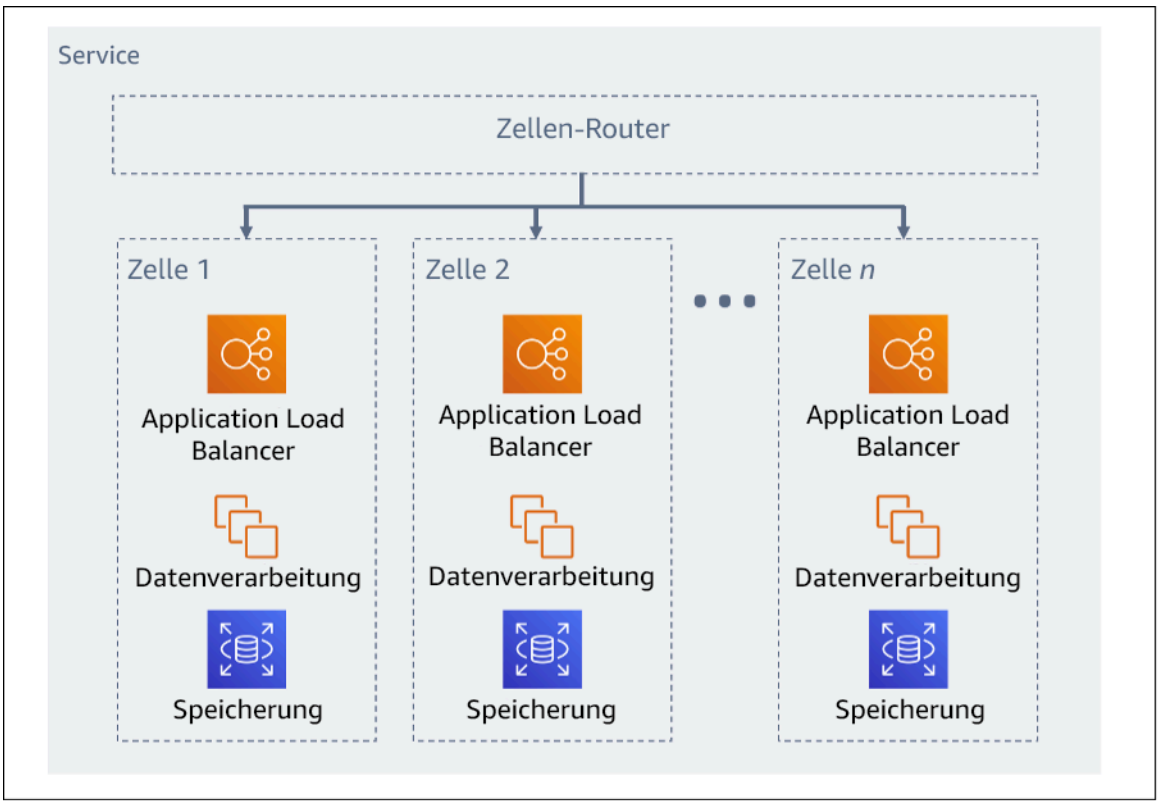


Abbildung 11: Zellenbasierte Architektur

Implementierungsschritte

Bei der Entwicklung einer zellenbasierten Architektur sind mehrere Designüberlegungen zu berücksichtigen:

1. Partitionsschlüssel: Bei der Wahl des Schlüssels für die Partitionierung sollten Sie besonders sorgfältig vorgehen.
   - Er sollte mit der Struktur des Service übereinstimmen oder mit der natürlichen Art und Weise, wie der Workload eines Service mit minimalen zellenübergreifenden Interaktionen unterteilt werden kann. Beispiele sind `Kunden-ID` oder `Ressourcen-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 Lebenszyklus ihrer Ressourcen nur mit einer einzigen 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 Ihrem 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. Routing-Schicht: Die Routing-Schicht ist eine gemeinsame Komponente von Zellen und kann daher nicht dieselbe Strategie der Segmentierung wie bei Zellen nutzen.
   - 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 Reliability, constant work, and a good cup of coffee (Zuverlässigkeit, konstante Arbeit und eine gute Tasse Kaffee) erläutert, führen einfache Designs und konstante Arbeitsmuster zu zuverlässigen Systemen und verringern die Antifragilität.
4. Zellengröße: Zellen sollten eine maximale Größe haben und nicht darüber hinaus wachsen 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 Durchführen von Lasttests für die Workload
   - Der gesamte Workload sollte durch Hinzufügen zusätzlicher Zellen wachsen, sodass der Workload mit der steigenden Nachfrage skalieren kann.
5. Multi-AZ oder Multi-Region-Strategien: Es sollten mehrere Schichten zur Ausfallsicherheit genutzt werden, um sich gegen verschiedene Fehlerbereiche zu schützen.
   - 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 Ihren Workload implementieren, ist er 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 finden Sie unter REL10-BP02 Auswählen der geeigneten Standorte für Ihre Multi-Standort-Bereitstellung.

6. Code-Bereitstellung: Eine gestaffelte Strategie für die Bereitstellung von Code 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 Details finden Sie unter Automatisierung sicherer, vollautomatischer Bereitstellungen.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: hoch

Ressourcen

Zugehörige bewährte Methoden:

- REL07-BP04 Durchführen von Lasttests für die Workload
- REL10-BP02 Auswählen der geeigneten Standorte für Ihre Multi-Standort-Bereitstellung

Zugehörige Dokumente:

- Reliability, constant work, and a good cup of coffee (Zuverlässigkeit, konstante Arbeit und ein ordentlicher Kaffee)
- AWS and Compartmentalization (Segmentierung mit AWS)
- Workload-Isolation mit Shuffle Sharding
- Automatisierung sicherer, vollautomatischer Bereitstellungen

Zugehörige Videos:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: Details und Strategien: Wie man die Kontrolle über große und kleine Systeme übernimmt)
- AWS re:Invent 2018: So minimiert AWS den Wirkungsradius von Fehlern (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 (AWS Summit ANZ 2021 – Alles schlägt fehl, immer wieder: Design für Ausfallsicherheit)

Zugehörige Beispiele:

- Well-Architected Lab: Fehlerisolierung mit Shuffle Sharding

Entwerfen von Workloads, die Komponentenausfälle verkraften

Workloads, die eine hohe Verfügbarkeit und eine niedrige mittlere Wiederherstellungszeit (Mean Time To Recovery, MTTR) benötigen, müssen auf Resilienz ausgelegt sein.

Bewährte Methoden

- 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 Ihres Workloads 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 eines Workloads 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 des Workloads 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 bei fehlender Befolgung dieser Best Practice: Hoch

Implementierungsleitfaden

Identifizieren Sie alle Workloads, die für die Überwachung überprüft werden sollen. Sobald Sie alle zu überwachenden Komponenten des Workloads 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 Fehlernberücksichtigen. Weitere Details zu grauen Fehlern finden Sie unter Graue Fehler im Whitepaper „Advanced Multi-AZ Resilience Patterns“ (Erweiterte Multi-AZ Resilience-Muster).

Implementierungsschritte

- 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.
  - Bestimmen Sie, ob eine detaillierte Überwachung für EC2-Instances und Auto Scaling notwendig ist. Eine detaillierte Überwachung liefert Metriken in einminütigen Intervallen, die Standardüberwachung liefert Metriken in fünfminütigen Intervallen.
  - Bestimmen Sie, ob eine erweiterte Überwachung für RDS erforderlich ist. Die erweiterte Überwachung verwendet einen Agenten auf RDS-Instances, um nützliche Informationen über verschiedene Prozesse oder Threads zu erhalten.
  - Bestimmen Sie die Anforderungen an die Überwachung von kritischen Serverless-Komponenten für Lambda, API Gateway, Amazon EKS, Amazon ECS, und alle Arten von Load Balancernberücksichtigen.
  - Ermitteln Sie die Überwachungsanforderungen von Speicherkomponenten für Amazon S3, Amazon FSx, Amazon EFSund Amazon EBS.
- Erstellen Sie benutzerdefinierte Metriken, um geschäftliche Key Performance Indicators (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 auf Fehler mithilfe von Benutzer-Canarys. Tests für synthetische Transaktionen (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, die das Benutzererlebnis nachverfolgen. Wenn Sie das Kundenerlebnis instrumentieren können, können Sie die Verschlechterung des Kundenerlebnisses feststellen.
- Legen Sie Alarme fest, um zu erkennen, wenn ein Teil Ihres Workloads 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 Tracing-Überwachung 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 Überwachungssysteme (mit CloudWatch oder X-Ray) Dashboards und einer Datenerfassung in einer eigenen Region und einem eigenen Konto.
- Erstellen Sie eine Integration zur Amazon Health Aware Überwachung, um die Überwachung von AWS-Ressourcen zu ermöglichen, bei denen es zu Leistungseinbußen kommen könnte. Für geschäftskritische Workloads bietet diese Lösung Zugriff auf proaktive und Echtzeitbenachrichtigungen für AWS-Services.

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-Canaries.
- Aktivieren oder deaktivieren Sie die detaillierte Überwachung für Ihre Instance
- Erweiterte Überwachung
- Überwachen ihrer Auto Scaling-Gruppe und Instances mit Amazon CloudWatch
- Veröffentlichen benutzerdefinierter Metriken
- Verwenden von Amazon CloudWatch-Alarmen
- Verwenden von CloudWatch-Dashboards
- Using Cross Region Cross Account CloudWatch Dashboards (Verwenden von konto- und regionenübergreifenden Amazon CloudWatch-Dashboards)
- Using Cross Region Cross Account X-Ray Tracing (Verwenden der konto- und regionenübergreifenden Amazon CloudWatch-Nachverfolgung)

- Verstehen der Verfügbarkeit
- Implementing Amazon Health Aware (AHA) (Implementierung von Amazon Health Aware (AHA))

Zugehörige Videos:

- Mitigating gray failures (Beheben von grauen Fehlern)

Zugehörige Beispiele:

- Well-Architected Lab: Level 300: Implementieren von Zustandsprüfungen und Verwalten von Abhängigkeiten zur Verbesserung der Zuverlässigkeit
- 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 Domänen.
- 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 bei fehlender Befolgung dieser Best Practice: Hoch

Implementierungsleitfaden

AWS-Services, wie z. B. Elastic Load Balancing und Amazon EC2 Auto Scalinghelfen, 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, Route 53 Route 53 ARC, CloudFront und AWS Global Accelerator können beim Routing des Datenverkehrs über AWS-Regionen helfen.

AWS-Services wie Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge oder 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 oder Amazon ECS ist Multi-AZ eine Konfigurationsoption. AWS kann den Datenverkehr zur intakten Instance umleiten, wenn ein Failover eingeleitet wird. Diese Failover-Aktion kann von AWS oder auf Wunsch des Kunden durchgeführt werden.

Für Amazon EC2-Instances, Amazon Redshift, Amazon ECS-Aufgaben oder Amazon EKS-Pods wählen Sie aus, in welchen Availability Zones sie bereitgestellt werden sollen. Für einige Designs bietet Elastic Load Balancing die Lösung, Instances in fehlerhaften Zonen zu erkennen und den Datenverkehr in die intakten Zonen zu routen. Elastic Load Balancing kann den Datenverkehr auch zu Komponenten in Ihrem On-Premises-Rechenzentrum routen.

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 for VPCs oder CloudFront eine Möglichkeit bieten. Sie können Internetdomänen definieren und Routing-Richtlinien einschließlich Zustandsprüfungen zuweisen, 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 geroutet 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 – Nutzen der Fehlerisolierung zum Schutz Ihres Workloads

Zugehörige Dokumente:

- Einstellen von RTO- und RPO-Zielen

- Einrichten von Route 53 ARC mit Application Load Balancers
- Failover mit gewichtetem Route 53-Routing
- DR mit Route 53 ARC
- EC2 mit automatischer Skalierung
- EC2-Bereitstellungen – Multi-AZ
- ECS-Bereitstellungen – Multi-AZ
- Datenverkehr umleiten mit Route 53 ARC
- 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 regionsübergreifende Replikation für EFS und Failover
- Regionsübergreifende EFS-Replikation und Failover
- Netzwerk-Failover
- S3-Endpunkt-Failover mit MRAP
- Erstellen einer regionsübergreifenden Replikation für S3
- Failover-Region API Gateway mit Route 53 ARC
- Failover mit Global Accelerator über mehrere Regionen
- Failover mit DRS
- Erstellen von Mechanismen für die Notfallwiederherstellung mit Amazon Route 53

Zugehörige Beispiele:

- Notfallwiederherstellung auf AWS
- Elastische Notfallwiederherstellung auf 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 Computing-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, einen exponentiellen Backoff mit Jitter durchzuführen. Die Fähigkeit zum Neustart ist eine Funktion, die in wiederherstellungsorientierten Computing- 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 bei fehlender Befolgung dieser Best Practice: 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 Computing-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-Compute-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. Beispiele für Services sind Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK und Amazon OpenSearch Service. Viele dieser Services können mit zusätzlichen Funktionen 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 lassen sich Ereignisse wie CloudWatch-Alarme oder Statusänderungen in anderen AWS-Services überwachen und filtern. Auf der Grundlage von Ereignisinformationen kann er dann AWS Lambda, Systems Manager Automation oder andere Ziele aufrufen, um eine angepasste Abhilfelogik für Ihren Workload auszuführen. Amazon EC2 Auto Scaling kann so konfiguriert werden, dass der Status der EC2-Instance überprüft wird. 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.
- Für die bereits erwähnten Computing-Instances verwenden Sie Load Balancing und wählen Sie den entsprechenden von Load-Balancer-Typ aus.
- Erwägen Sie die Reparatur für Amazon RDS. Bei Standby-Instances konfigurieren Sie den automatischen Failover auf die Standby-Instance. Bei Amazon RDS-Lesereplikaten ist ein automatisierter Workflow erforderlich, um ein Lesereplikat zur primären Instance zu machen.
- Implementieren Sie die automatische Wiederherstellung auf EC2-Instances, die Anwendungen bereitstellen, die nicht an mehreren Standorten bereitgestellt werden können und die einen Neustart bei Fehlern tolerieren können. 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 Einbindungspunkte für Amazon Elastic File System oder Dateisysteme für Lustre und Windows. Mit AWS OpsWorkskönnen 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.
- Amazon EventBridge kann verwendet werden, um Ereignisse zu überwachen und zu filtern, wie CloudWatch-Alarme oder Zustandsänderungen in anderen AWS-Services. Auf der Grundlage von Ereignisinformationen kann es dann AWS Lambda (oder andere Ziele) aufrufen, um eine angepasste Wiederherstellungslogik für Ihren Workload auszuführen.

Ressourcen

Zugehörige bewährte Methoden:

- Definition der Verfügbarkeit

- REL11-BP01 Überwachen aller Komponenten der Workload auf Fehler

Zugehörige Dokumente:

- So funktioniert AWS Auto Scaling
- Automatische Wiederherstellung mit Amazon EC2
- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- Was ist Amazon FSx for Lustre?
- Was ist Amazon FSx for 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-Alarmen
- Amazon RDS-Failover
- SSM – Systems Manager-Automatisierung
- Bewährte Methoden für eine widerstandsfähige Architektur

Zugehörige Videos:

- Automatically Provision and Scale OpenSearch Service (OpenSearch automatisch Bereitstellen und skalieren)
- Automatischer Failover mit Amazon RDS

Zugehörige Beispiele:

- Workshop zu Auto Scaling
- 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 Computing-Instance, Erstellen von Block-Speicher und Beschreiben von Warteschlangen-Services. Wenn Sie Computing-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 bei fehlender Befolgung dieser Best Practice: Mittel: Bei bestimmten Arten von Service-Störungen sind die 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.

Anleitung zur Umsetzung

Um die Aktionen auf der Datenebene zu begrenzen, bewerten Sie für jeden Service, welche Aktionen zur Wiederherstellung des Services erforderlich sind.

Nutzen Sie Amazon Route 53 Application Recovery Controller, um den DNS-Datenverkehr zu verlagern. Diese Funktionen 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 sind global verteilt und für ein Service Level Agreement (SLA) mit einer Verfügbarkeit von 100 % entworfen worden.

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.

Verwenden Sie für Amazon EC2 Designs mit statischer Stabilität, um Aktionen auf der Steuerebene zu begrenzen. Zu den Aktionen auf der Steuerebene gehört das Hochskalieren von Ressourcen einzeln oder über Auto Scaling-Gruppen (ASG). 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 im Dokument 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 jeden Workload, der 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)
- Migrieren Sie zu Lambda und seinen Skalierungsmethoden (Datenebene) oder Amazon EC2 und ASG (Steuerebene)
- 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. Mit der 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.

- Route 53 Record change (control plane) or Route 53 ARC (data plane) (Route 53-Datensatzänderung (Steuerebene) oder Route 53 ARC (Datenebene))
- Route 53 Health checks for more automated updates (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 Routing des Datenverkehrs zur sekundären Region (Aktion auf 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-Ausführungen (aufgeteilt in die Steuerebene und die Datenebene)
- AWS Elemental MediaStore-Datenebene
- Entwickeln hoch resilienter Anwendungen mit Amazon Route 53 Application Recovery Controller, Teil 1: Stack für eine einzelne Region
- Entwickeln hoch resilienter Anwendungen mit Amazon Route 53 Application Recovery Controller, Teil 2: Stack für mehrere Regionen
- Erstellen von Mechanismen für die Notfallwiederherstellung mit Amazon Route 53
- Was ist Route 53 Application Recovery Controller?
- Kubernetes-Steuerebene und -Datenebene

Zugehörige Videos:

- Back to Basics – Using Static Stability (Zurück zu den Basics – Verwendung statischer Stabilität)
- Building resilient multi-site workloads using AWS global services (Aufbau belastbarer Workloads an mehreren Standorten mit globalen AWS-Services)

Zugehörige Beispiele:

- Vorstellung von Amazon Route 53 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 Route 53 Application Recovery Controller, Teil 1: Stack für eine einzelne Region
- Entwickeln hoch resilienter Anwendungen mit Amazon Route 53 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 der 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 der Workload nur in einem einzigen Modus ausgeführt wird.

Gewünschtes Ergebnis: Workloads zeigen im Normalmodus und im Fehlermodus kein bimodales Verhalten.

Typische Anti-Muster:

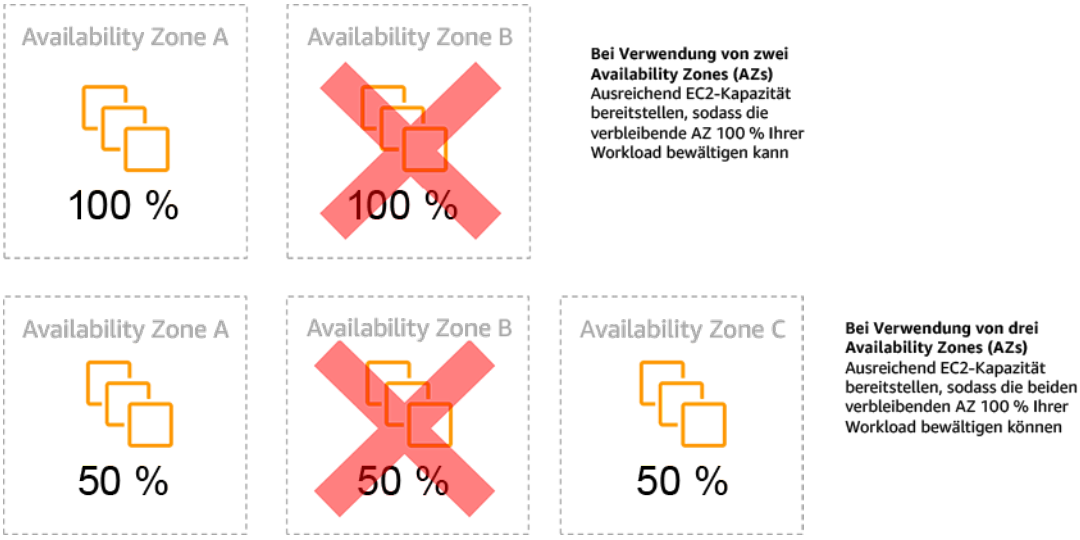
- Es wird davon ausgegangen, dass Ressourcen unabhängig vom Umfang des Fehlers immer bereitgestellt werden können.
- Während eines Fehlers wird versucht, dynamisch Ressourcen zu erwerben.
- Es werden keine ausreichenden Ressourcen für Zonen oder Regionen bereitgestellt, bis ein Fehler auftritt.
- Statische 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 bei fehlender Befolgung dieser Best Practice: Mittel

Implementierungsleitfaden

Bimodales Verhalten bedeutet, dass ein 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 Elastic Load Balancing-Designs genügend Instances in jeder Availability Zone bereitstellen, so dass die Verarbeitung der Workload auch beim Entfernen einer Availability Zone gewährleistet ist. Anschließend sollten Sie die beeinträchtigten Instances mithilfe von Elastic Load Balancing oder Amazon Route 53-Zustandsprüfungen entlasten. 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 in Availability Zones

Dies muss gegen die Kosten für dieses Modell und den geschäftlichen Nutzen der Aufrechterhaltung des Workloads 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 Ihren Workload gegeneinander abwägen. Ansätze mit statischer Stabilität gelten für eine Vielzahl von Architekturen, darunter Computing-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 Ihres Workloads 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 Ihres Workloads erheblich ändern und führt wahrscheinlich zu Fehlern.

Bewerten Sie kritische Workloads, um festzustellen, für welche Workloads diese Art von Resilienzentwurf 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
- 存储: Amazon S3 (eine Zone), Amazon EFS (Bereitstellungen), Amazon FSx (Bereitstellungen)
- Load Balancer: Unter 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 Route 53 Application Recovery Controller
- Konfigurieren Sie Datenbank-Read-Replikate, um den Verlust einer einzelnen primären Instance oder einer Read Replica 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 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, einen 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 durch Availability Zones
- Fault Isolation Boundaries (Grenzen für die Fehlerisolierung)
- Statische Stabilität mithilfe von Availability Zones
- Multi-Zone RDS (RDS für mehrere Zonen)
- Minimierung der Abhängigkeiten bei der Planung der Notfallwiederherstellung
- Regionsübergreifendes DNS-Routing
- MRAP-Amazon S3-Routing mit mehreren Regionen
- AWS Global Accelerator
- Route 53 ARC
- Amazon S3 mit einer Zone
- Cross Zone Load Balancing (Zonenübergreifender Lastenausgleich)

Zugehörige Videos:

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328) (Statische Stabilität in AWS: AWS re:Invent 2019: Einführung der Amazon Builders' Library (DOP328))

Zugehörige Beispiele:

- Die Amazon Builders' Library: Statische Stabilität durch Availability Zones

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 Ihr 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.
- Nicht berücksichtigt werden die grauen Fehler bei der Gestaltung von Überwachung und Alarmen.
- 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 bei fehlender Befolgung dieser Best Practice: 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. `ALARM AKTIV` oder `0K`) 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 des Workloads nicht ordnungsgemäß funktioniert.
- Alarme für die Infrastruktur geben an, wann Ressourcen skaliert werden sollen. Alarme können visuell in Dashboards angezeigt werden, Warnungen per Amazon SNS oder E-Mail senden und mit Auto Scaling die Ressourcen für einen Workload hoch- oder herunterskalieren.
- 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 direkt eine Amazon SNS-API aufrufen, um Benachrichtigungen zu senden und alle Automatisierungen zur Behebung oder Kommunikation zu verknüpfen.
- Setzen Sie Amazon Health Aware Überwachung, um die Überwachung von AWS-Ressourcen zu ermöglichen, bei denen es zu Leistungseinbußen kommen könnte. Für geschäftskritische Workloads bietet diese Lösung Zugriff auf proaktive und Echtzeitbenachrichtigungen für AWS-Services.

Ressourcen

Zugehörige bewährte Methoden für Well-Architected:

- Definition der Verfügbarkeit

Zugehörige Dokumente:

- Erstellen eines CloudWatch-Alarms auf der Basis eines statischen Schwellenwerts
- Was ist Amazon EventBridge?
- Was ist Amazon Simple Notification Service?
- Veröffentlichen benutzerdefinierter Metriken
- Using Amazon CloudWatch Alarms (Verwenden von Amazon CloudWatch-Alarmen)
- Amazon Health Aware (AHA)
- 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)

Entwerfen Sie Ihr Produkt zur Erfüllung der Verfügbarkeitsziele und der Uptime-SLAs (Service Level Agreements). 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 Ziel für die Verfügbarkeit und eine SLA für Leistungsmetrik, die überwacht und aufrechterhalten werden können, um die Geschäftsziele zu erreichen.

Typische Anti-Muster:

- Entwurf und Bereitstellung von Workloads ohne Einstellung 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 von Schlüsselzielen für die Ausfallsicherheit hilft Ihnen, 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.

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 des Workloads festgelegt oder ermittelt werden. Sie sollten in der Entwurfsphase festgelegt werden und als Leitlinien für verschiedene Entscheidungen und Abwägungen dienen.

- Jeder Workload sollte seine 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. Jeder Workload sollte seine Abhängigkeiten und deren SLAs berücksichtigen. Wählen Sie im Allgemeinen Abhängigkeiten mit Verfügbarkeitszielen aus, die den Zielen Ihres Workloads entsprechen oder höher sind.
- Ziehen Sie eine lose Kopplung in Betracht, damit Ihr 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 einem Workload zu erhöhen.
- Die Überwachbarkeit 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 eines Workloads ist eine der Grundlagen für jedes effektive Design. Diese Entwürfe müssen Kompromisse in Bezug auf Designkomplexität, Service-Abhängigkeiten, Leistung, Skalierung und Kosten berücksichtigen.

Implementierungsschritte

- Überprüfen und dokumentieren Sie den Workload-Entwurf unter Berücksichtigung der folgenden Fragen:
  - Wo werden die Steuerebenen im Workload verwendet?

  - Wie implementiert der Workload die Ausfallsicherheit?
  - Wie sehen die Entwurfsmuster 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 der 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 Segmentierung Ihres Workloads
- 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-BP05 Testen der Ausfallsicherheit mit Chaos-Engineering
- REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten:
- Grundlegendes zum Workload-Status

Zugehörige Dokumente:

- Availability with redundancy (Verfügbarkeit mit Redundanz)
- Zuverlässigkeitssäule – Verfügbarkeit
- Measuring availability (Messung der Verfügbarkeit)
- AWS Fault Isolation Boundaries (AWS-Grenzen für die Fehlerisolierung)

- Modell der geteilten Verantwortung für Ausfallsicherheit
- Statische Stabilität mithilfe von Availability Zones
- AWS Service Level Agreements (SLAs)
- Guidance for Cell-based Architecture on AWS (Leitfaden für eine zellenbasierte Architektur auf AWS)
- AWS-Infrastruktur
- Advanced Multi-AZ Resiliance Patterns whitepaper (Whitepaper: Fortschrittliche Multi-AZ-Resiliance-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 Ihr Workload funktionale und nicht funktionale Anforderungen erfüllt, da Fehler oder Leistungsengpässe die Zuverlässigkeit Ihres Workloads beeinträchtigen können. Ermitteln Sie anhand von Tests der Ausfallsicherheit Ihres Workloads latente Bugs, die erst während der Produktion auftreten. Führen Sie diese Tests regelmäßig durch.

Bewährte Methoden

- REL12-BP01 Untersuchen von Fehlern mit Playbooks:
- REL12-BP02 Durchführen von Analysen nach Vorfällen
- REL12-BP03 Testen funktionaler Anforderungen
- REL12-BP04 Testen von Skalierungs- und Leistungsanforderungen
- REL12-BP05 Testen der Ausfallsicherheit mit Chaos-Engineering
- REL12-BP06 Regelmäßiges Abhalten von Gamedays

REL12-BP01 Untersuchen von Fehlern mit Playbooks:

Ermöglichen 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.

Gängige Antimuster:

- Planen der Bereitstellung eines Workloads, ohne die Prozesse für die Diagnose von Problemen oder die 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 Einführung dieser bewährten Methode: Durch das Erfassen 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 von Probleme mit Playbooks. Playbooks sind dokumentierte Prozesse für die Untersuchung von Problemen. Durch die Dokumentation der Prozesse in Playbooks schaffen Sie

die Voraussetzung für eine einheitliche und schnelle Reaktion auf Fehlerszenarien. 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 von 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 ausgelöst oder im Rahmen von Playbook-Aktivitäten ausgeführt werden. Sie können auch als Antwort auf identifizierte Ereignisse die Ausführung eines Playbooks auslösen.
  - Automatisieren Sie Ihre operativen Playbooks mit AWS Systems Manager
  - AWS Systems Manager Befehl ausführen
  - AWS Systems Manager Automation
  - Was ist AWS Lambda?
  - Was ist Amazon EventBridge?
  - Verwenden von Amazon CloudWatch Alarmen

Ressourcen

Zugehörige Dokumente:

- AWS Systems Manager Automation
- AWS Systems Manager Befehl ausführen
- Automatisieren Sie Ihre operativen Playbooks mit AWS Systems Manager
- Verwenden von Amazon CloudWatch Alarmen
- Verwenden von Canaries (Amazon CloudWatch Synthetics)
- Was ist Amazon EventBridge?
- Was ist AWS Lambda?

Ähnliche Beispiele:

- Automatisieren von Vorgängen mit Playbooks und 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 ein wiederholtes Auftreten nach Möglichkeit 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 verfolgen einen konsistenten und vereinbarten Ansatz für die Analyse nach einem Vorfall. Einer dieser Mechanismen ist der COE-Prozess (Correction of Error, Fehlerkorrektur). 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

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?
    - Aktionspunkte
    - Verwandte Artikel
- 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:

- Darum sollten Sie eine Fehlerkorrektur (COE) entwickeln

Zugehörige Videos:

- Amazon’s approach to failing successfully (Amazons Ansatz zum erfolgreichen Scheitern)
- AWS re:Invent 2021 - Amazon Builders’ Library: Operational Excellence at Amazon (Die Amazon Builders' Library: Operative Exzellenz von Amazon)

REL12-BP03 Testen funktionaler Anforderungen

Verwenden Sie Techniken wie Komponenten- und Integrationstests, mit denen die erforderliche Funktionalität validiert wird.

Im Idealfall sollten diese Tests automatisch als Teil von Build- und Bereitstellungsaktionen ausgeführt werden. Mit AWS CodePipeline übergeben Entwickler beispielsweise Änderungen an ein Quell-Repository, in dem CodePipeline die Änderungen automatisch erkennt. Diese Änderungen werden vorgenommen und Tests werden ausgeführt. Nachdem die Tests abgeschlossen sind, wird der erstellte Code für 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.

Außerdem zeigen frühere Erfahrungen, dass synthetische Transaktionstests (auch bekannt als Canary-Tests, aber nicht zu verwechseln mit Canary-Bereitstellungen), die ausgeführt werden können und das Kundenverhalten simulieren, zu den wichtigsten Testprozessen gehören. Führen Sie diese Tests für Ihre Workload-Endpunkte konstant von verschiedenen Remote-Standorten aus. Mit Amazon CloudWatch Synthetics können Sie Canaries erstellen, um Ihre Endpunkte und APIs zu überwachen.

Risikostufe, falls diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Testen funktionaler Anforderungen: Dazu gehören Komponenten- und Integrationstests, mit denen die erforderliche Funktionalität validiert wird.

- Verwenden von AWS CodeBuild mit CodePipeline zum Testen von Code und zum Ausführen von Builds
- AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild (AWS CodePipeline fügt Unterstützung für Komponententests und angepasste Integrationstests mit AWS CodeBuild hinzu)
- Kontinuierliche Bereitstellung und kontinuierliche Integration
- Using synthetic monitoring (Amazon CloudWatch Synthetics) (Verwenden von synthetischer Überwachung (Amazon CloudWatch Synthetics))
- Automatisierung von Softwaretests

Ressourcen

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Implementierung einer Continuous Integration-Pipeline unterstützen können
- AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild (AWS CodePipeline fügt Unterstützung für Komponententests und angepasste Integrationstests mit AWS CodeBuild hinzu)
- AWS Marketplace: Für die kontinuierliche Integration geeignete Produkte
- Kontinuierliche Bereitstellung und kontinuierliche Integration
- Automatisierung von Softwaretests
- Verwenden von AWS CodeBuild mit CodePipeline zum Testen von Code und zum Ausführen von Builds
- Using synthetic monitoring (Amazon CloudWatch Synthetics) (Verwenden von synthetischer Überwachung (Amazon CloudWatch Synthetics))

REL12-BP04 Testen von Skalierungs- und Leistungsanforderungen

Verwenden Sie Techniken wie Lasttests, um zu überprüfen, ob die Workload die Skalierungs- und Leistungsanforderungen erfüllt.

In der Cloud können Sie bei Bedarf eine Testumgebung für Ihren Workload in Produktionsumgebungen erstellen. Wenn Sie diese Tests auf einer herunterskalierten Infrastruktur

ausführen, müssen Sie die Ergebnisse auf den Maßstab der Produktionsumgebung hochrechnen. Last- und Leistungstests können auch in der Produktion durchgeführt werden. Achten Sie dabei darauf, Benutzer nicht zu beeinträchtigen und Ihre Testdaten mit Tags zu versehen, sodass sie nicht mit Benutzerdaten vermischt werden und Nutzungsstatistiken oder Produktionsberichte verfälschen.

Stellen Sie mit Tests sicher, dass Ihre Basisressourcen, Skalierungseinstellungen, Servicekontingente und die Ausfallsicherheit unter Auslastung wie erwartet funktionieren.

Risikostufe, wenn diese Best Practice nicht eingeführt wird: Hoch

Implementierungsleitfaden

- Testen Sie Skalierungs- und Leistungsanforderungen. Führen Sie Lasttests durch, um zu prüfen, ob der Workload die Skalierungs- und Leistungsanforderungen erfüllt.
  - Verteilte Lasttests auf AWS: Simulation Tausender verbundener Benutzer
  - Apache JMeter
    - Stellen Sie Ihre Anwendung in einer Umgebung bereit, die mit Ihrer Produktionsumgebung identisch ist, und führen Sie einen Lasttest durch.
      - Erstellen Sie auf Grundlage von "Infrastructure as Code"-Konzepten eine Umgebung, die Ihrer Produktionsumgebung möglichst ähnlich ist.

Ressourcen

Zugehörige Dokumente:

- Verteilte Lasttests auf AWS: Simulation Tausender verbundener Benutzer
- Apache JMeter

REL12-BP05 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 Chaosexperimenten 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 Ihres Workloads gewinnen Sie die nötige Zuversicht, dass die Wiederherstellungsverfahren Ihres resilienten Entwurfs im Fall eines realen Fehlers funktionieren.

Risikostufe bei fehlender Befolgung dieser Best Practice: 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 Ihr 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 Ihr 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.

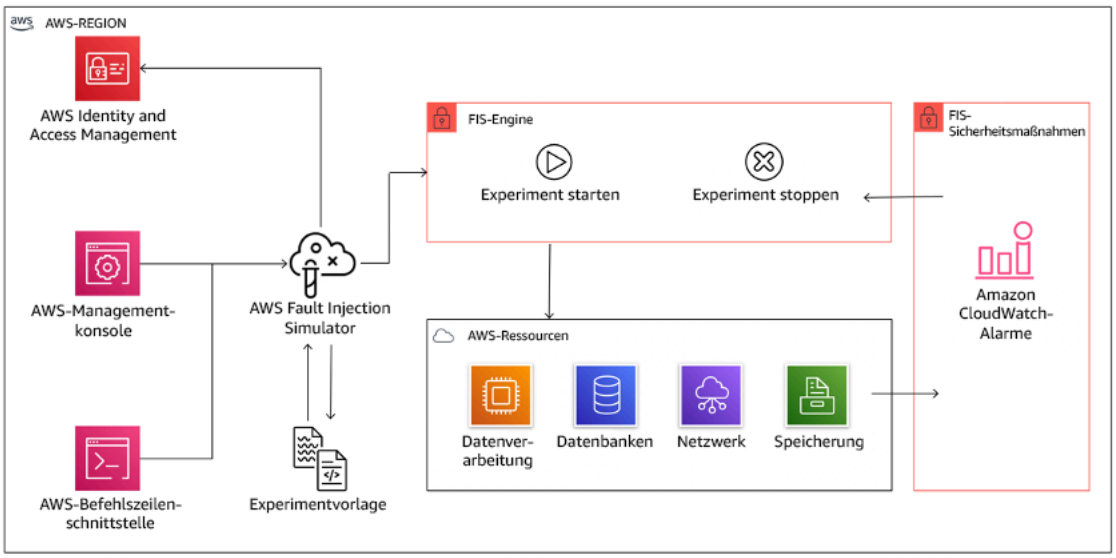
Zur Validierung von Fallback- oder Failover-Mechanismen für externe Abhängigkeiten, die bei zeitweisen Netzwerkdisruptionen ausgelöst werden, sollten Ihre Komponenten diese Ereignisse durch das Blockieren des Zugriffs auf externe Anbieter über einen bestimmten Zeitraum simulieren, der von wenigen Sekunden 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. Der Service unterstützt die gleichzeitige Injektion von Fehlern in verschiedene Arten von Ressourcen, 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.



Diagramm, das die Integration von AWS Fault Injection Service in AWS-Ressourcen zeigt, 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 Meshund Litmus Chaossowie kommerzielle Optionen wie Gremlin. Zur Erweiterung der Art der Fehler, die in AWS injiziert werden können, kann AWS FIS in Chaos Mesh und Litmus Chaos integriert werden.So können Sie Fehlerinjektions-Workflows über verschiedene Tools hinweg koordinieren. 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

- Ermitteln Sie die Fehler, mit denen experimentiert werden soll.

  Bewerten Sie das Design Ihres Workloads in Bezug auf die Resilienz. Diese Designs (anhand der Best Practices des Well-Architected Frameworkerstellt) berücksichtigen Risiken im Zusammenhang

mit kritischen Abhängigkeiten, früheren Ereignissen, bekannten Problemen und Compliance-Anforderungen. Listen Sie die einzelnen Elemente des Designs auf, die Resilienz zeigen sollen, und die Fehler, denen es standhalten soll. Weitere Informationen zur Erstellung dieser Listen finden Sie im Whitepaper zur Überprüfung der betrieblichen Bereitschaft. Dieses Whitepaper führt Sie durch die Entwicklung eines Prozesses zur Verhinderung der Wiederholung früherer Vorfälle. 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 Ihren Workload bereit. FMEA wird von Adrian Cockcroft in Failure Modes and Continuous Resilience(Fehlerarten und kontinuierliche Resilienz) detaillierter beschrieben.

- 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 den Workload insgesamt.

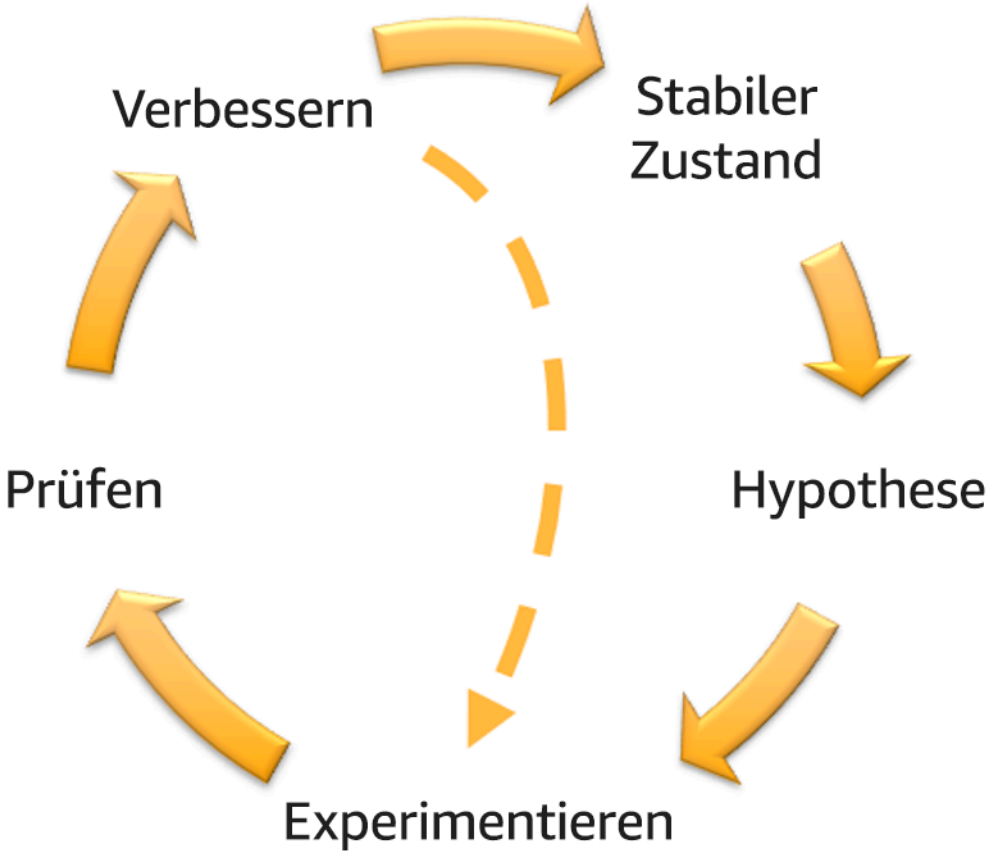
  Analysieren Sie hinsichtlich der Häufigkeit eines bestimmten Fehlers frühere Daten für den betreffenden Workload, wenn verfügbar. Wenn keine Daten verfügbar sind, verwenden Sie Daten zu anderen Workloads, die in einer ähnlichen Umgebung ausgeführt werden.

  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 des Workloads berücksichtigen. Beispielsweise ist für einen Workload, der 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.

- Für jedes von Ihnen ausgeführte Experiment sollten Sie sich am Schwungrad für Chaos-Engineering und kontinuierliche Resilienz orientieren.



Schwungrad für Chaos-Engineering und kontinuierliche Resilienz unter Verwendung der wissenschaftlichen Methode von Adrian Hornsby.

- Definieren Sie den Steady-State als die messbare Ausgabe eines Workloads, der ein normales Verhalten zeigt.

  Ihr Workload befindet sich im Steady-State, wenn er zuverlässig und wie erwartet ausgeführt wird. Daher sollten Sie die Integrität Ihres Workloads überprüfen, bevor Sie den Steady-State definieren. Steady-State bedeutet nicht notwendigerweise, dass sich ein Fehler nicht auf den 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.

- Formulieren Sie eine Hypothese dazu, wie der Workload auf den Fehler reagieren wird.

  Eine gute Hypothese basiert darauf, wie der Workload den Fehler voraussichtlich bewältigt, um den Steady-State zu wahren. Die Hypothese besagt, dass bei einem Fehler eines spezifischen Typs das System oder der Workload weiter im Steady-State bleiben, da der 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 *(spezifischer Fehler)* auftritt, wird der *Workload* (Name des Workloads) *(Maßnahmen zur Bewältigung beschreiben),* um die Auswirkungen auf *geschäftliche oder technische Metriken einzudämmen*.

  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 Traffic 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 der 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).

- Führen Sie das Experiment aus, indem Sie den Fehler injizieren.

  Ein Experiment sollte grundsätzlich nicht zu einem Ausfall führen und vom Workload toleriert werden. Wenn Sie wissen, dass der 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 Ereignisse, die Ihnen bekannt sind, die Sie jedoch nicht vollständig verstehen. Unbekannt-unbekannte Ereignisse sind Ereignisse, die Sie weder kennen noch vollständig verstehen. Wenn Sie Experimente für einen Workload ausführen, von dem Sie wissen, dass er 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 Ihr Workload das Experiment überstehen sollte, können Sie das Experiment starten. Für die Injektion von Fehlern gibt es verschiedene Optionen. Für AWS-Workloads stellt AWS FIS zahlreiche vordefinierte Fehlersimulationen bereit, die als Aktionenbezeichnet 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 des Workloads 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 Ausführung 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 Experimenten unter Verwendung von AWS FIS finden Sie im Resilient and Well-Architected Apps with Chaos Engineering Lab. Darüber hinaus analysiert AWS Resilience Hub Ihren 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 realen Bedingungen ausgeführt werden. Dabei sollten nach Möglichkeit Canary-Bereitstellungen verwendet werden, die sowohl ein

Kontrollsystem als auch ein Experimentsystem bereitstellen. 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 Traffic 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 Produktions-Traffic 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 des Workloads 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 den Workload und die zugrunde liegenden Systeme wieder in den ursprünglichen, gut funktionierenden Zustand zurückversetzen. Häufig führt das resiliente Design des betreffenden Workloads eine Selbstreparatur durch. Einige Fehlerdesigns oder fehlgeschlagenen Experimente können Ihren Workload jedoch in einem nicht erwarteten Fehlerzustand zurücklassen. Nach dem Ende des Experiments müssen Sie dies erkennen und den 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.

- Prüfen Sie die Hypothese.

  Grundlagen des Chaos-Engineering stellt die folgende Anleitung für die Verifizierung des Steady-State Ihres Workloads bereit:

  Konzentrieren Sie sich auf die messbare Ausgabe des Systems und nicht auf die internen Attribute des Systems. 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 des Workloads 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 Ihres Workloads direkt zugreift, eine synthetische Überwachung einrichten (auch als Benutzer-Canary bezeichnet).

- 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 Best Practices der AWS Well-Architected-Säule „Zuverlässigkeit“ an. Zusätzliche Anleitungen und Ressourcen finden Sie in der AWS Builder’s Library.Diese Bibliothek enthält Artikel zur Verbesserung von Zustandsprüfungen oder zur Nutzung von Wiederholungen mit Backoff im Anwendungscodeund mehr.

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 der Workload im Steady-State und der Zyklus wird fortgesetzt.

- 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 einen Workload zutrifft, sollte das Experiment automatisiert werden, um innerhalb Ihrer CI/CD-Pipeline kontinuierlich als Regression ausgeführt zu werden. Informationen hierzu finden Sie in diesem Blog, der die Ausführung von AWS FIS-Experimenten mit AWS CodePipelinebeschreibt. Dieses Lab für wiederholte AWS FIS-Experimente in einer CI/CD-Pipeline ermöglicht Ihnen de Sammlung praktischer Erfahrungen.

  Fehlerinjektionsexperimente sind auch Bestandteil von Gamedays (siehe REL12-BP06 Regelmäßiges Abhalten 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.

- Erfassen und speichern Sie die Ergebnisse der Experimente.

  Die Ergebnisse von Fehlerinjektionsexperimenten müssen erfasst und gespeichert werden. Erfassen Sie dabei alle notwendigen Daten (wie 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 Bestandteil 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 Beispiele:

- Well-Architected Lab: Level 300: Testen auf Resilienz von Amazon EC2, Amazon RDS und Amazon S3
- Chaos Engineering in AWS (Lab)
- Resilient and Well-Architected Apps with Chaos Engineering Lab
- Serverless-Chaos (Lab)
- Messen und Verbessern der Resilienz Ihrer Anwendung mit AWS Resilience Hub (Lab)

Zugehörige Tools:

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

REL12-BP06 Regelmäßiges Abhalten von Gamedays

Nutzen Sie Gamedays, um Ihre Verfahren für Reaktionen auf Ereignisse und Fehler unter möglichst produktionsnahen Bedingungen (einschließlich Produktionsumgebungen) regelmäßig mit den Personen zu testen, die auch in tatsächlichen Fehlerszenarien beteiligt sind. Bei Gamedays werden Vorkehrungen getroffen, die sicherstellen, das sich Produktionsereignisse nicht auf Benutzer auswirken.

Bei Gamedays wird ein Fehler oder Ereignis simuliert, um Systeme, Prozesse und die Reaktion von Teams zu testen. Dabei sollen die auszuführenden Aktionen vom Team wie im Fall eines außergewöhnlichen Ereignisses tatsächlich ausgeführt werden. So können Sie nachvollziehen, wo nachgebessert werden kann. Zudem üben Sie dabei ein, wie Ihre Organisation mit Ereignissen umgeht. Gamedays sollten regelmäßig ausgeführt werden, damit die Reaktion für Ihr Team zu einem Reflex wird.

Nachdem Sie Ihre Maßnahmen für Ausfallsicherheit implementiert und in Umgebungen abseits der Produktion getestet haben, können Sie an einem Gameday feststellen, ob in der Produktion alles wie geplant funktioniert. An einem Gameday, insbesondere am ersten, werden alle Entwickler und Betriebsteams miteinbezogen und über Zeitpunkt sowie Ablauf des Tests informiert. Die Runbooks müssen vorhanden sein. Simulierte Ereignisse, auch potenzielle Ausfallereignisse, werden wie vorgeschrieben in den Produktionssystemen ausgeführt und deren Auswirkungen werden bewertet. Wenn alle Systeme wie vorgesehen funktionieren, erfolgen Erkennung und Selbstreparatur mit minimalen oder gar keinen Auswirkungen. Wenn jedoch negative Auswirkungen festgestellt werden, wird ein Rollback des Tests durchgeführt und die Workload-Probleme werden bei Bedarf manuell behoben (gemäß Runbook). Da Gamedays oft in der Produktion stattfinden, sollten alle Vorkehrungen getroffen werden, um Kunden vor Beeinträchtigungen der Verfügbarkeit zu schützen.

Gängige Antimuster:

- Die eigenen Verfahren werden dokumentiert, jedoch nie trainiert.
- Entscheidungsträger werden bei den Tests außen vorgelassen.

Vorteile der Einführung dieser Best Practice: Die regelmäßige Durchführung von Gamedays sorgt dafür, dass bei einem tatsächlichen Vorfall alle Mitarbeiter die Richtlinien und Verfahren befolgen. Außerdem wird überprüft, ob diese Richtlinien und Verfahren geeignet sind.

Risikostufe, falls diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Planen Sie Gamedays, um Ihre Runbooks und Playbooks regelmäßig zu trainieren. An Gamedays sollten alle Mitarbeiter beteiligt werden, die von Produktionsunterbrechungen betroffen sein können: Geschäftsinhaber, Entwickler, Produktionsmitarbeiter und die Teams, die auf Vorfälle reagieren.
  - Führen Sie Ihre Last- oder Leistungstests durch und schleusen Sie anschließend Fehler ein.
  - Prüfen Sie die Runbooks auf Anomalien und suchen Sie nach Möglichkeiten zur Ausführung der Playbooks.
    - Optimieren Sie bei Abweichungen die Runbooks oder ändern Sie das Verhalten. Ermitteln Sie bei Ausführung eines Playbooks das Runbook, das hätte verwendet werden sollen, oder erstellen Sie ein neues.

Ressourcen

Zugehörige Dokumente:

- Was ist AWS GameDay?

Zugehörige Videos:

- AWS re:Invent 2019: Verbesserung der Ausfallsicherheit mit Chaos-Engineering (DOP309-R1)

  Zugehörige Beispiele:

- AWS Well-Architected Labs: Testen der Ausfallsicherheit

Planung der Notfallwiederherstellung

Backups 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 Ziele 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 Disruptionen und die Kosten von Wiederherstellungen sind ebenfalls wichtige Faktoren bei der Ermittlung des Unternehmenswerts, den Notfallwiederherstellungen von Workloads bieten.

Die Verfügbarkeit und Notfallwiederherstellung stützen sich auf dieselben Best Practices, z. B. die Überwachung auf Fehler, die Bereitstellung mehrerer Standorte und das automatische Failover. Allerdings liegt der Fokus der Verfügbarkeit auf den Komponenten der Workload, während er bei der Notfallwiederherstellung auf separaten Kopien der gesamten Workload liegt. Die Notfallwiederherstellung hat andere Ziele als die Verfügbarkeit. Der Fokus liegt auf der Zeit, die nach einem Notfall für die Wiederherstellung benötigt wird.

Bewährte Methoden

- 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:

Für die Workload gelten ein Recovery Time Objective (RTO, Wiederherstellungsdauer) und ein Recovery Point Objective (RPO, Wiederherstellungszeitpunkt).

Die Wiederherstellungsdauer 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.

Der Wiederherstellungszeitpunkt ist die maximal zulässige Zeitspanne seit dem letzten Wiederherstellungspunkt. Damit wird festgelegt, was als akzeptabler Datenverlust zwischen dem letzten Wiederherstellungspunkt und der Service-Unterbrechung gilt.

RTO- und RPO-Werte sind wichtige Überlegungen bei der Auswahl einer geeigneten Notfallwiederherstellungsstrategie (Disaster Recovery, DR) für Ihre Workload. Diese Ziele werden vom Unternehmen festgelegt und dann von den technischen Teams zur Auswahl und Umsetzung einer DR-Strategie verwendet.

Gewünschtes Ergebnis:

Jeder Workload sind ein RTO und ein RPO zugewiesen, die auf der Grundlage der geschäftlichen Auswirkungen definiert werden. Die Workload wird einer vordefinierten Stufe zugewiesen, die die Serviceverfügbarkeit und den akzeptablen Datenverlust mit einem entsprechenden RTO und RPO definiert. Wenn eine solche Einstufung nicht möglich ist, kann die Zuweisung individuell pro Workload erfolgen, mit der Absicht, zu einem späteren Zeitpunkt Stufen zu erstellen. RTO und RPO werden als eine der Hauptüberlegungen für die Auswahl einer Notfallwiederherstellungsstrategie für die Workload verwendet. Weitere Überlegungen bei der Auswahl einer DR-Strategie sind Kostenbeschränkungen, Abhängigkeiten von der Workload und betriebliche Anforderungen.

Bei der RTO sind die Auswirkungen anhand der Dauer eines Ausfalls zu verstehen. Ist sie linear oder gibt es nichtlineare Auswirkungen? (Beispiel: Nach vier Stunden wird eine Fertigungsstraße bis zum Beginn der nächsten Schicht stillgelegt.)

Eine Matrix der Notfallwiederherstellung wie die folgende kann Ihnen helfen zu verstehen, wie die Kritikalität der Workload mit den Wiederherstellungszielen zusammenhängt. (Beachten Sie, dass die tatsächlichen Werte für die X- und Y-Achsen an die Bedürfnisse Ihres Unternehmens angepasst werden sollten.)

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

Abbildung 16: Matrix der Notfallwiederherstellung

Gängige Antimuster:

- Keine definierten Wiederherstellungsziele.
- Auswählen beliebiger Wiederherstellungsziele.
- Auswählen von Wiederherstellungszielen, die zu lasch sind und die Geschäftsziele nicht erfüllen.
- Kein Verständnis des Auswirkung von Ausfallzeiten und Datenverlust.
- Auswahl unrealistischer Wiederherstellungsziele, wie z. B. Null-Zeit bis zur Wiederherstellung und Null-Datenverlust, die für Ihre Workload-Konfiguration möglicherweise nicht erreicht werden können.
- Auswählen von Wiederherstellungszielen, die strikter sind als die tatsächlichen Geschäftsziele. Dies erzwingt Implementierungen für die Notfallwiederherstellung, die kostspieliger und komplizierter sind als die Anforderungen der Workload.
- Auswahl von Wiederherstellungszielen, die mit denen einer abhängigen Workloads unvereinbar sind.
- Ihre Wiederherstellungsziele berücksichtigen nicht die Einhaltung gesetzlicher Vorschriften.
- RTO und RPO sind für eine Workload definiert, aber nie getestet.

Vorteile der Einführung dieser bewährten Methode: Die Wiederherstellungsziele für Dauer und Datenverlust sind als Orientierungshilfe für die Implementierung der Notfallwiederherstellung erforderlich.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Hoch

Implementierungsleitfaden

Bei der gegebenen Workload müssen Sie die Auswirkungen von Ausfallzeiten und Datenverlusten auf Ihr Unternehmen verstehen. Die Auswirkungen werden in der Regel mit zunehmender Ausfallzeit oder Datenverlust größer, aber die Form dieses Anstiegs kann je nach Art der Workload unterschiedlich sein. So können Sie z. B. Ausfallzeiten bis zu einer Stunde ohne größere Beeinträchtigung tolerieren, danach steigen die Auswirkungen jedoch schnell an. Die Auswirkungen auf das Unternehmen zeigen sich in vielen Formen, darunter monetäre Kosten (z. B. entgangene Einnahmen), Kundenvertrauen (und Auswirkungen auf den Ruf), betriebliche Probleme (z. B. fehlende Gehaltsabrechnungen oder verringerte Produktivität) und gesetzliche Risiken. Führen Sie die folgenden Schritte aus, um diese Auswirkungen zu verstehen und RTO und RPO für Ihre Workload festzulegen.

Implementierungsschritte

1. Bestimmen Sie die Interessengruppen Ihres Unternehmens für diese Workload und arbeiten Sie mit ihnen zusammen, um diese Schritte umzusetzen. Die Wiederherstellungsziele für eine Workload sind eine geschäftliche Entscheidung. Die technischen Teams arbeiten dann mit den Business-Stakeholdern zusammen, um anhand dieser Ziele eine DR-Strategie auszuwählen.

   > **Note**
   >
   > Für die Schritte 2 und 3 können Sie Folgendes verwenden: the section called “Implementierungsarbeitsblatt”.

2. Sammeln Sie die notwendigen Informationen, um eine Entscheidung zu treffen, indem Sie die folgenden Fragen beantworten.
3. Gibt es in Ihrem Unternehmen Kategorien oder Stufen der Kritikalität für die Auswirkungen von Workloads?
   a. Falls zutreffend, ordnen Sie diese Workload einer Kategorie zu.
   b. Falls nicht zutreffend, richten Sie diese Kategorien ein. Legen Sie fünf oder weniger Kategorien fest und verfeinern Sie die Spanne der angestrebten Wiederherstellungszeit für jede Kategorie. Zu den Beispielkategorien gehören: kritisch, hoch, mittel, niedrig. Um zu verstehen, wie sich Workloads den Kategorien zuordnen lassen, sollten Sie prüfen, ob die Workload unternehmenskritisch, geschäftswichtig oder nicht geschäftsrelevant ist.
   c. Legen Sie RTO und RPO für die Workload je nach Kategorie fest. Wählen Sie immer eine Kategorie, die strikter ist (niedrigere RTO- und RPO-Werte) als die bei der Eingabe dieses Schritts berechneten Rohwerte. Wenn dies zu einer unangemessen großen Veränderung des Wertes führt, sollten Sie eine neue Kategorie anlegen.
4. Weisen Sie auf der Grundlage dieser Antworten der Workload RTO- und RPO-Werte zu. Dies kann direkt geschehen oder durch Zuweisung der Workload zu einer vordefinierten Serviceebene.
5. Dokumentieren Sie den Notfallwiederherstellungsplan (Disaster Recovery Plan, DRP) für diese Workload, der Teil der Unternehmensstrategie ist. Betriebskontinuitätsplan (BCP)an einem Ort, der für das Workload-Team und die Stakeholder zugänglich ist
   a. Halten Sie die RTO- und RPO-Werte sowie die zur Ermittlung dieser Werte verwendeten Informationen fest. Geben Sie eine Strategie zur Bewertung der Auswirkungen der Workload auf das Unternehmen an.
   b. Erfassen Sie neben RTO und RPO auch andere Metriken, die Sie für Notfallwiederherstellungsziele verfolgen oder zu verfolgen planen

   c. Sie fügen diesem Plan Details zu Ihrer DR-Strategie und Ihrem Runbook hinzu, wenn Sie diese erstellen.

6. Indem Sie die Kritikalität der Workload in einer Matrix wie der in Abbildung 15 nachschlagen, können Sie damit beginnen, vordefinierte Serviceebenen für Ihr Unternehmen festzulegen.
7. Nachdem Sie eine DR-Strategie (oder einen Machbarkeitsnachweis für eine DR-Strategie) gemäß implementiert haben,the section called “REL13-BP02: Verwenden von definierten Wiederherstellungsstrategien, um die Wiederherstellungsziele zu erreichen”testen Sie diese Strategie, um die tatsächliche RTC (Recovery Time Capability) und RPC (Recovery Point Capability) der Workload zu bestimmen. Wenn diese nicht den angestrebten Wiederherstellungszielen entsprechen, arbeiten Sie entweder mit Ihren Stakeholdern zusammen, um diese Ziele anzupassen, oder nehmen Sie Änderungen an der DR-Strategie vor, um die Zielvorgaben zu erreichen.

Primäre Fragen

1. Wie lange kann die Workload maximal ausfallen, bevor es zu schwerwiegenden Auswirkungen auf das Unternehmen kommt?
   a. Bestimmen Sie die monetären Kosten (direkte finanzielle Auswirkungen) für das Unternehmen pro Minute, wenn die Workload unterbrochen wird.
   b. Bedenken Sie, dass die Auswirkungen nicht immer linear sind. Die Auswirkungen können zunächst begrenzt sein und dann ab einem kritischen Zeitpunkt rasch zunehmen.
2. Wie groß ist die maximale Datenmenge, die verloren gehen kann, bevor es zu schwerwiegenden Auswirkungen auf das Unternehmen kommt?
   a. Berücksichtigen Sie diesen Wert für Ihren wichtigsten Datenspeicher. Identifizieren Sie die jeweilige Kritikalität für andere Datenspeicher.
   b. Können Workload-Daten bei Verlust wiederhergestellt werden? Wenn dies aus betrieblicher Sicht einfacher ist als Backup und Wiederherstellung, dann wählen Sie das RPO auf der Grundlage der Kritikalität der Ursprungsdaten, die zur Wiederherstellung der Workload-Daten verwendet werden.
3. Wie lauten die Wiederherstellungsziele und Verfügbarkeitserwartungen von Workloads, von denen dieser abhängt (Downstream), oder von Workloads, die von diesem abhängen (Upstream)?
   a. Wählen Sie Wiederherstellungsziele, die es dieser Workload ermöglichen, die Anforderungen der vorgelagerten Abhängigkeiten zu erfüllen

b. Wählen Sie Wiederherstellungsziele, die angesichts der Wiederherstellungsmöglichkeiten der nachgelagerten Abhängigkeiten erreichbar sind. Unkritische nachgelagerte Abhängigkeiten (die Sie „umgehen“ können) können ausgeschlossen werden. Oder arbeiten Sie mit kritischen, nachgelagerten Abhängigkeiten zusammen, um deren Wiederherstellungsmöglichkeiten zu verbessern.

Weitere Fragen

Überlegen Sie sich, wie diese Fragen auf diese Workload zutreffen könnten:

4. Haben Sie unterschiedliche RTO und RPO je nach Art des Ausfalls (Region vs. Region)? AZ, etc.)?
5. Gibt es einen bestimmten Zeitpunkt (Saisonabhängigkeit, Verkaufsveranstaltungen, Produkteinführungen), zu dem sich Ihr RTO/RPO ändern kann? Wenn ja, was ist die unterschiedliche Messung und die zeitliche Begrenzung?
6. Wie viele Kunden sind von einer Unterbrechung der Workload betroffen?
7. Welche Auswirkungen hat es auf den Ruf, wenn die Workload unterbrochen wird?
8. Welche anderen betrieblichen Auswirkungen können auftreten, wenn die Workload unterbrochen wird? Zum Beispiel Auswirkungen auf die Produktivität der Mitarbeiter, wenn die E-Mail-Systeme nicht verfügbar sind oder wenn die Lohnbuchhaltungssysteme keine Transaktionen übermitteln können.
9. Wie stimmen RTO und RPO der Workload mit der DR-Strategie der Geschäftsbereiche und des Unternehmens überein?
10. Gibt es interne vertragliche Verpflichtungen für die Erbringung einer Dienstleistung? Gibt es Strafen für die Nichteinhaltung dieser Vorgaben?
11. Welche rechtlichen oder Compliance-Bedingungen gelten für die Daten?

Implementierungsarbeitsblatt

Sie können dieses Arbeitsblatt für die Implementierungsschritte 2 und 3 verwenden. Sie können dieses Arbeitsblatt an Ihre speziellen Bedürfnisse anpassen, 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

Grad des Aufwands für den Implementierungsplan: Niedrig

Ressourcen

Ähnliche bewährte Methoden:

- the section called “REL09-BP04 Verifizieren der Sicherungsintegrität und -verfahren durch regelmäßiges Wiederherstellen der Daten”
- the section called “REL13-BP02: Verwenden von definierten Wiederherstellungsstrategien, um die Wiederherstellungsziele zu erreichen”
- the section called “REL13-BP03 Testen der Implementierung der Notfallwiederherstellung zur Validierung:”

Zugehörige Dokumente:

- AWS Architecture Blog: Notfallwiederherstellungsserie

- Notfallwiederherstellung von Workloads auf AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)
- Verwalten von Ausfallsicherheit mit AWS Resilience Hub
- APN-Partner: Partner, die Sie bei der Notfallwiederherstellung unterstützen können
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte

Relevante Videos

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2) (Architekturmuster für Aktiv-Aktiv-Anwendungen in mehreren Regionen)
- Notfallwiederherstellung von Workloads auf 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 jeden Workload gibt es eine definierte und implementierte Notfallwiederherstellungsstrategie, die dem Workload das Erreichen der Notfallwiederherstellungsziele ermöglicht. 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, den Workload auszuführen. Die häufigsten Wiederherstellungsziele sind RTO und RPO, wie besprochen in REL13-BP01 Definieren von Wiederherstellungszielen bei Ausfällen und Datenverlusten:.

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 werden nach zunehmenden Kosten und zunehmender Komplexität und abnehmender RTO und RPO aufgelistet. Die Wiederherstellungsregion verweist auf eine andere AWS-Region als die für Ihren Workload verwendete primäre Region.

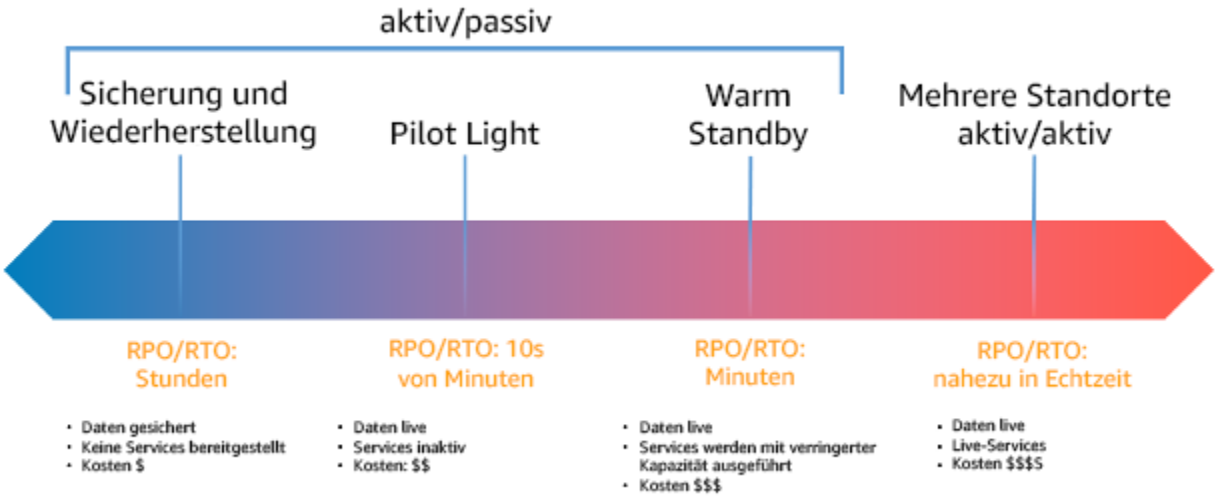


Abbildung 17: Notfallwiederherstellungsstrategien (DR)

- Backup und Wiederherstellung (RPO im Stundenbereich, RTO innerhalb von 24 Stunden oder weniger): Sichern Sie Ihre Daten und Anwendungen in der Wiederherstellungsregion. Die Verwendung automatisierter oder kontinuierlicher Backups ermöglicht eine zeitpunktgenaue Wiederherstellung, 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 die 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 im Minutenbereich, RTO innerhalb von zehn Minuten): Bereitstellung einer Kopie Ihrer Core-Workload-Infrastruktur in der Wiederherstellungsregion. 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 Compute werden nicht bereitgestellt, sondern können bei Bedarf mit der erforderlichen Konfiguration und dem Anwendungscode erstellt werden.
- Warm-Standby (RPO im Sekundenbereich, RTO im Minutenbereich): Aufrechterhaltung einer herunterskalierten, aber voll funktionsfähigen Version Ihres Workloads, die immer in der Wiederherstellungsregion ausgeführt wird. Geschäftskritische Systeme sind vollständig dupliziert und ständig aktiv, aber mit herunterskalierter Infrastruktur. 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 einer vollständigen Abdeckung spricht man von Hot-Standby.
- Multi-Region (Multi-Site) Aktiv/Aktiv (RPO nahe Null, RTO potenziell Null): Ihr Workload wird an mehreren AWS-Regionen-Standorten bereitgestellt und bedient aktiv den Datenverkehr von diesen. 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 zeitpunktgenaue 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 cloud-native 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 seine maximal zulässige 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.

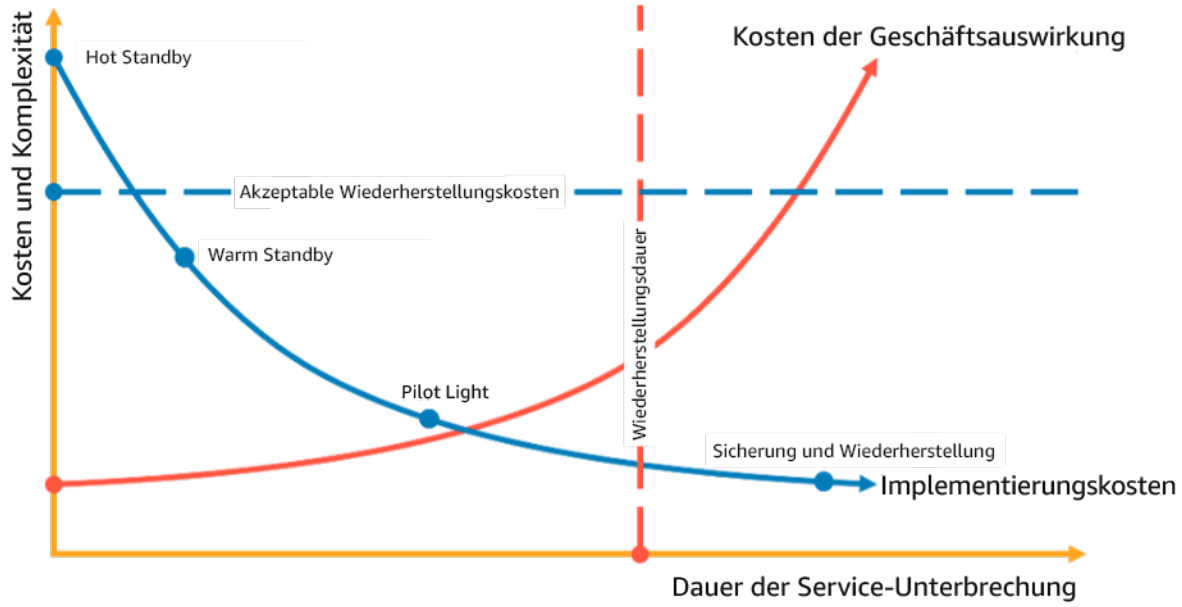


Abbildung 18: Auswahl einer DR-Strategie auf der Grundlage von RTO und Kosten

Weitere Informationen finden Sie unter Business Continuity Plan (BCP).

2. Überprüfen Sie die Muster, wie die ausgewählte 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 Verfügbarkeitszonen innerhalb einer einzigen Region als DR-Strategie verwenden, die Elemente mehrerer dieser Strategien nutzt.

In den folgenden Schritten können Sie die Strategie auf Ihren spezifischen Workload anwenden.

Sicherung und Wiederherstellung

Backup und Wiederherstellung ist die am einfachsten zu implementierende Strategie, erfordert jedoch mehr Zeit und Aufwand für die Wiederherstellung des Workloads, 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.

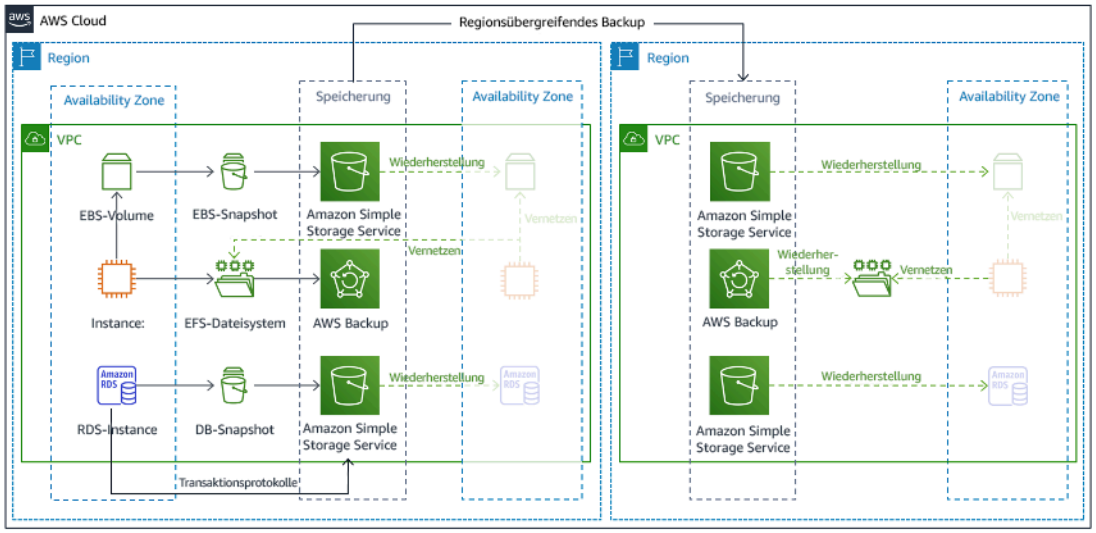


Abbildung 19: Sicherungs- und Wiederherstellungsarchitektur

Weitere Details zu dieser Strategie finden Sie unter Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery (Architektur zur Notfallwiederherstellung (DR) auf AWS, Teil II: Backup und Wiederherstellung mit schneller Wiederherstellung).

Pilot Light

Mit dem Pilot-Light-Ansatz replizieren Sie Ihre Daten von Ihrer primären Region auf Ihre Recovery Region. 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 Compute-Instances bereitgestellt.

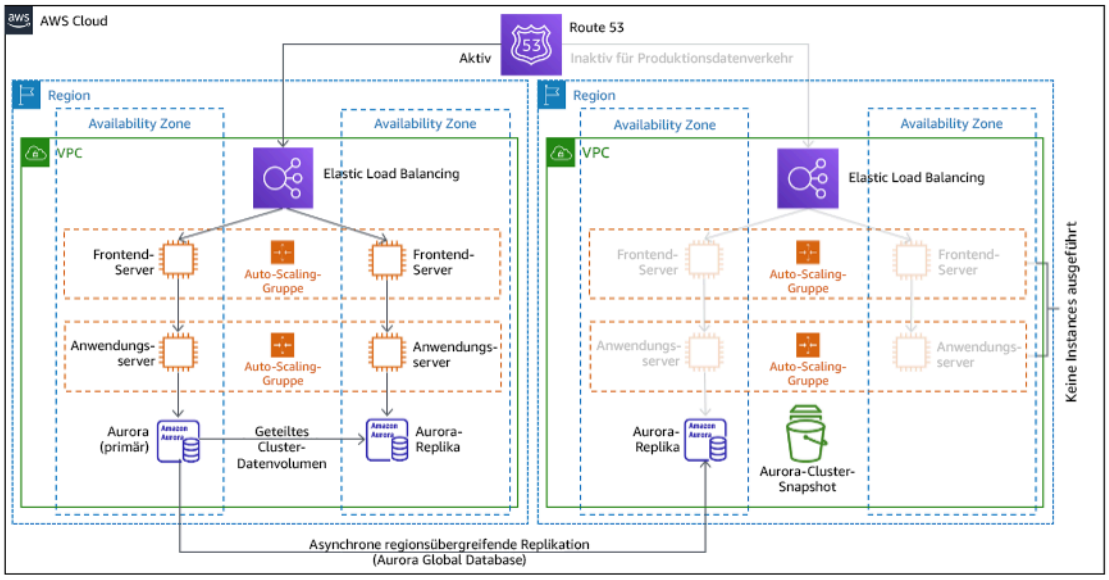


Abbildung 20: Pilot-Light-Architektur

Weitere Details zu dieser Strategie finden Sie unter Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby (Architektur zur Notfallwiederherstellung (DR) auf AWS, Teil III: Pilot-Light und Warm-Standby).

Warm Standby

Der Warm-Standby-Ansatz besteht darin, 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.

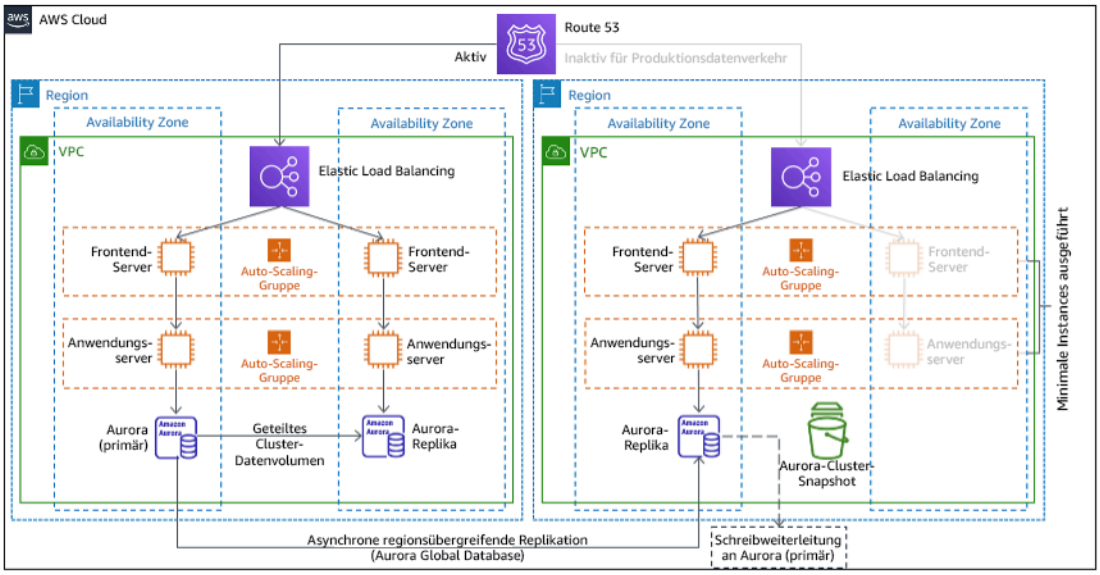


Abbildung 21: Warm-Standby-Architektur

Der Einsatz von Warm-Standby oder Pilot-Light erfordert ein Hochskalieren der Ressourcen in der Wiederherstellungsregion. Um sicherzustellen, dass Kapazität bei Bedarf verfügbar ist, sollten Sie die Verwendung von Kapazitätsreservierungen für EC2-Instances in Betracht ziehen. Wenn Sie AWS Lambda verwenden, können Sie mit provisioned concurrency Ausführungsumgebungen bereitstellen, damit diese sofort auf die Aufrufe Ihrer Funktion reagieren können.

Weitere Details zu dieser Strategie finden Sie unter Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby (Architektur zur Notfallwiederherstellung (DR) auf AWS, Teil III: Pilot-Light und Warm-Standby).

Mehrere Standorte aktiv/aktiv

Sie können Ihren Workload gleichzeitig in mehreren Regionen als Teil einer Multi-Site Aktiv/Aktiv-Strategie ausführen. Multi-Site Aktiv/Aktiv bedient den Datenverkehr aus allen Regionen, in denen es eingesetzt wird. Kunden können diese Strategie aus anderen Gründen als DR wählen. Sie 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 der Workload in einer der AWS-Regionen, in denen er 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.

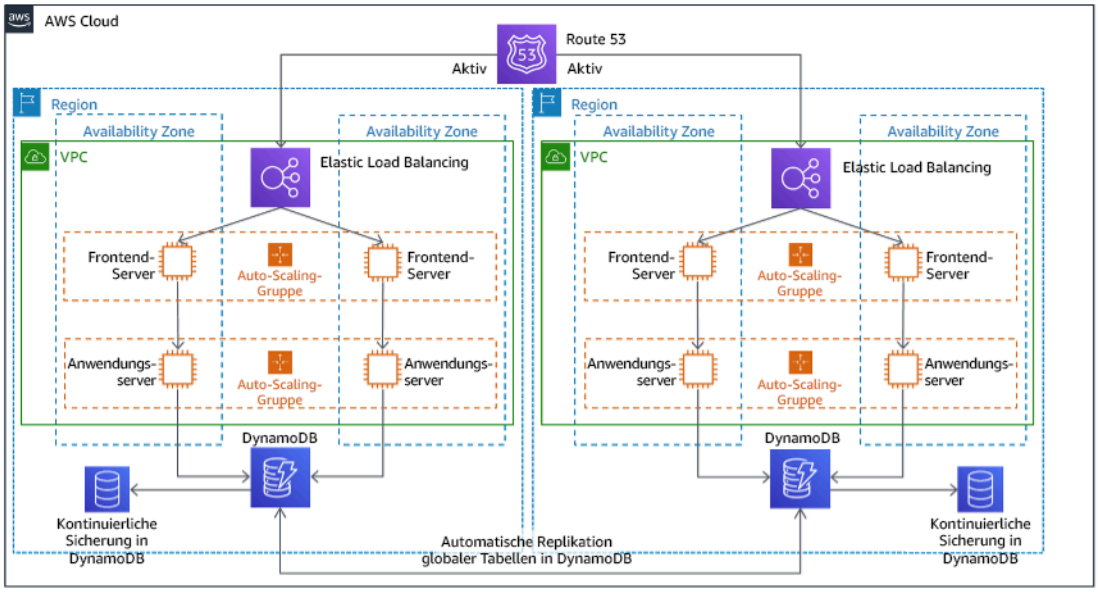


Abbildung 22: Multi-Site Aktiv/Aktiv Architektur

Weitere Details zu dieser Strategie finden Sie unter Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active (Architektur zur Notfallwiederherstellung (DR) auf AWS, Teil IV: Multi-Site Aktiv/Aktiv).

AWS Elastic Disaster Recovery

Wenn Sie für die Notfallwiederherstellung die Pilot-Light- oder die 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 Computing-Ressourcen, wenn diese als Teil eines Failover oder Drills initiiert wird.

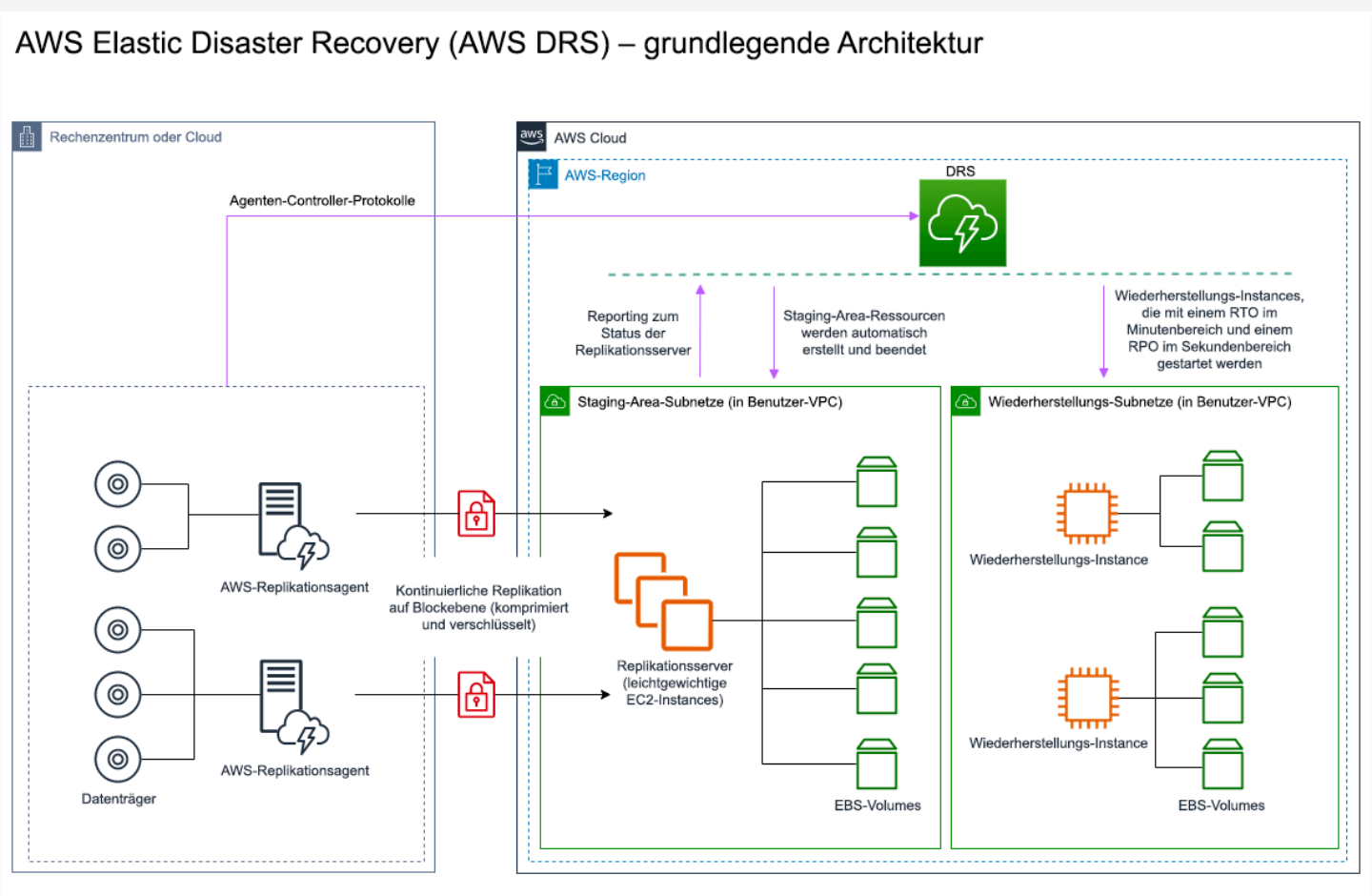


Abbildung 23: AWS Elastic Disaster Recovery-Architektur

Zusätzliche Praktiken 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 zeitpunktgenaue Wiederherstellung. Sie müssen auch die replizierten Daten in der Wiederherstellungssite sichern, um zusätzlich zu den Replikaten zeitpunktgenaue Sicherungen zu erstellen.

Verwendung von mehreren 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

nutzt einen Aktiv/Aktiv-Ansatz für mehrere Standorte, da die Amazon EC2-Instance und der Elastic-Load-Balancer über Ressourcen verfügen, die in mehreren AZs bereitgestellt werden und aktiv Anfragen weiterleiten. Die Architektur demonstriert auch Hot-Standby, d. h. wenn die primäre Amazon RDS-Instance ausfällt (oder die AZ selbst), wird die Standby-Instance zur primären Instance befördert.

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 bei Daten wichtig, die auf eine einzige Zone beschränkt sind – wie 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 eine Single-Region, Multi-AZ-Notfallwiederherstellung wird im Blogbeitrag Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack (Erstellen hoch belastbarer Anwendungen mit Amazon Route 53 Application Recovery Controller, Teil 1: Single-Region-Stack) 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 zur Aufbewahrung von Daten. 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. Beurteilen Sie die Ressourcen Ihrer Workloads und deren Konfiguration in der Wiederherstellungsregion vor dem Failover (während des normalen Betriebs).

Für die Infrastruktur und AWS-Ressourcen verwenden Sie Infrastructure-as-Code-Angebote wie AWS CloudFormation oder Drittanbieter-Tools wie Hashicorp Terraform. Um mehrere Konten und Regionen über einen einzelnen Vorgang bereitzustellen, können Sie AWS CloudFormation StackSets nutzen. 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. Mit CloudFormation-Parametern und bedingter Logik können Sie mit einer einzigen Vorlage steuern, ob ein bereitgestellter Stack aktiv oder standby ist. Wenn Sie Elastic Disaster Recovery verwenden, repliziert und orchestriert der Service die Wiederherstellung von Anwendungskonfigurationen und Computing-Ressourcen.

Alle Notfallwiederherstellungsstrategien setzen voraus, dass die Datenquellen innerhalb der AWS-Region gesichert werden und diese Backups dann in die Wiederherstellungsregion kopiert werden. AWS Backup bietet eine zentrale Anzeige, in der Sie Backups für diese Ressourcen konfigurieren, planen und überwachen können. Bei Pilot-Light, Warm-Standby und Multi-Site Aktiv/Aktiv sollten Sie außerdem Daten aus der primären Region auf Datenressourcen in der Wiederherstellungsregion replizieren (z. B. 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 über Regionen hinweg arbeiten, finden Sie in der Blogserie über die Erstellung einer multiregionalen Anwendung mit AWS-Services.

4. Legen Sie fest, wie Sie Ihre Wiederherstellungsregion bei Bedarf (während eines Notfallereignisses) für einen Failover bereit machen wollen, und setzen Sie diese um.

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. Unter REL09-BP01 Ermitteln und Sichern aller zu sichernden Daten oder Reproduzieren der Daten aus Quellen finden Sie weitere Informationen. Zum Wiederaufbau der Infrastruktur gehört auch die Erstellung von Ressourcen wie EC2-Instances, zusätzlich zu den Amazon Virtual Private Cloud (Amazon VPC), Subnetzen und Sicherheitsgruppen. Sie können einen Großteil des Wiederherstellungsprozesses automatisieren. Wie das geht, erfahren Sie in diesem Blogbeitrag.

5. Legen Sie fest und implementieren Sie, wie Sie den Datenverkehr bei Bedarf (im Notfall) zum Failover umleiten werden.

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 Möglichkeit 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-Domänennamen verknüpfen. Um einen manuell initiierten Failover zu implementieren, können Sie Amazon Route 53 Application Recovery Controller verwenden. Dieser Service bietet eine hochverfügbare API für die Datenebene, um den Datenverkehr in die Wiederherstellungsregion umzuleiten. Verwenden Sie bei der Implementierung von Failover Vorgänge auf der Datenebene und vermeiden Sie solche auf der Steuerebene, wie beschrieben in REL11-BP04 Nutzen der Datenebene und nicht der Steuerebene während der Wiederherstellung.

Weitere Informationen zu dieser und anderen Optionen finden Sie in diesem Abschnitt des Whitepapers zur Notfallwiederherstellung.

6. Entwerfen Sie einen Plan für den Failback Ihres Workloads.

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, führt DynamoDB die Weiterleitung aller ausstehenden Schreibvorgänge durch, sobald sie wieder online ist (selbst wenn die Tabelle in der primären Region nicht mehr verfügbar ist). Wenn Sie Amazon Aurora Global Database und einen verwalteten, geplanten Failover verwenden, dann wird Aurora die bestehende Replikationstopologie der globalen 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. Ein Beispiel für eine Anleitung, wie Sie dies mit Amazon Aurora Global Database unter der Annahme eines ungeplanten Failovers umsetzen, finden Sie in dieser Übung: Fail Back a Global Database (Failback einer globalen Datenbank).

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 Details finden Sie unter Performing a failback (Durchführen eines Failbacks).

Grad des Aufwands 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: Notfallwiederherstellungsserie
- Notfallwiederherstellung von Workloads auf AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)
- Optionen für die Notfallwiederherstellung in der Cloud
- Entwickeln Sie eine Multi-Region-Serverless-Backend-Lösung, die aktiv/aktiv ist.
- Multi-Region-Serverless-Backend – neu aufgelegt
- RDS: Regionsübergreifendes Replizieren von Lesereplikaten
- Route 53: Konfigurieren von DNS-Failover
- S3: Regionsübergreifende Replikation
- Was ist AWS Backup?
- Was ist Route 53 Application Recovery Controller?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Get Started – AWS (Erste Schritte)

- APN-Partner: Partner, die Sie bei der Notfallwiederherstellung unterstützen können
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte

Zugehörige Videos:

- Notfallwiederherstellung von Workloads auf AWS
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2) (Architekturmuster für Aktiv/Aktiv-Anwendungen in mehreren Regionen)
- Erste Schritte mit AWS Elastic Disaster Recovery | Amazon Web Services

Zugehörige Beispiele:

- Well-Architected Lab – Disaster Recovery (Well-Architected Lab – Notfallwiederherstellung) – Eine Reihe von Workshops zur Veranschaulichung von Notfallwiederherstellungsstrategien

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: Das regelmäßige Testen Ihres Plans zur Notfallwiederherstellung stellt sicher, dass er funktioniert, wenn er benötigt wird, und dass Ihr Team weiß, wie die Strategie auszuführen ist.

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 zweiten 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. Workloads für die Wiederherstellung auslegen. Regelmäßige Tests der Wiederherstellungspfade Das Recovery-orientierte Computing 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. Für Amazon EC2-basierte Workloads verwenden Sie AWS Elastic Disaster Recovery, um Drill-Instances für Ihre Notfallwiederherstellungsstrategie zu implementieren und zu starten. AWS Elastic Disaster Recovery bietet die Möglichkeit, Drills effizient auszuführen, was Ihnen bei der Vorbereitung auf ein Failover-Ereignis hilft. 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: Notfallwiederherstellungsserie
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte

- AWS Elastic Disaster Recovery
- Notfallwiederherstellung von Workloads auf AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)
- AWS Elastic Disaster Recovery – Vorbereitungen auf einen Failover
- The Berkeley/Stanford Recovery-Oriented Computing (ROC) Project
- Was ist AWS Fault Injection Simulator?

Zugehörige Videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (AWS re:Invent 2018: Architekturmuster für Multi-Region Aktiv/Aktive-Anwendungen)
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS (AWS re:Invent 2019: Backup-and-Wiederherstellung und Notfallwiederherstellungs-Lösungen mit AWS)

Zugehörige Beispiele:

- Well-Architected Lab – Testing for Resiliency (Well-Architected Lab – Testen auf Ausfallsicherheit)

REL13-BP04 Verwalten der Konfigurationsabweichungen am Standort oder in der Region der Notfallwiederherstellung:

Stellen Sie sicher, dass die Infrastruktur, die Daten und die Konfiguration am Standort oder in der Region der Notfallwiederherstellung den Anforderungen entsprechen. Sie sollten beispielsweise prüfen, ob AMIs und Service Quotas auf dem neuesten Stand sind.

AWS Config überwacht und zeichnet Ihre AWS-Ressourcenkonfigurationen kontinuierlich auf. Es kann Abweichungen erkennen und als Auslöser für AWS Systems Manager Automation dienen, um diese zu beheben und Warnmeldungen zu senden. AWS CloudFormation kann zusätzlich Abweichungen in bereitgestellten Stacks erkennen.

Gängige Antimuster:

- Versäumnis, Aktualisierungen an Ihren Wiederherstellungsstandorten vorzunehmen, wenn Sie Konfigurations- oder Infrastrukturänderungen an Ihren Hauptstandorten vornehmen.
- Mögliche Einschränkungen (z. B. Serviceunterschiede) an Ihren primären Standorten und den Standorten für die Notfallwiederherstellung werden nicht berücksichtigt.

Vorteile der Einführung dieser bewährten Methode: Wenn Ihre Umgebung für die Notfallwiederherstellung mit der vorhandenen Umgebung konsistent ist, gewährleisten dies eine vollständige Wiederherstellung.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Sicherstellen, dass die Bereitstellung an Haupt- und Sicherungsstandorte erfolgt Pipelines für die Bereitstellung von Anwendungen in der Produktion müssen die Anwendungen an alle Standorte verteilen, die in der Strategie für die Notfallwiederherstellung angegeben sind. Dazu gehören auch Entwicklungs- und Testumgebungen.
- Aktivieren von AWS Config zum Verfolgen von Standorten mit möglichen Abweichungen. Erstellen Sie mithilfe von AWS Config Regeln Systeme, die Ihre Strategien für die Notfallwiederherstellung durchsetzen und bei Erkennung von Abweichungen Warnungen generieren.
  - Korrigieren von nicht konformen AWS-Ressourcen mit AWS-Config-Regeln
  - AWS Systems Manager Automation
- Verwenden Sie AWS CloudFormation zur Bereitstellung Ihrer Infrastruktur. AWS CloudFormation kann Abweichungen zwischen den Angaben in den CloudFormation-Vorlagen und der tatsächlichen Bereitstellung erkennen.
  - AWS CloudFormation: Ermitteln von Abweichungen im gesamten CloudFormation-Stack

Ressourcen

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Notfallwiederherstellung unterstützen können
- AWS Architecture Blog: Notfallwiederherstellungsserie
- AWS CloudFormation: Ermitteln von Abweichungen im gesamten CloudFormation-Stack
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte
- AWS Systems Manager Automation
- Notfallwiederherstellung von Workloads auf 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

Relevante Videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2) (Architekturmuster für Aktiv-Aktiv-Anwendungen in mehreren Regionen)

REL13-BP05: Automatisieren der Wiederherstellung

Automatisieren Sie mit Tools von AWS oder Drittanbietern die Systemwiederherstellung und leiten Sie Datenverkehr zum Standort oder zur Region der Notfallwiederherstellung weiter.

Basierend auf konfigurierten Zustandsprüfungen können AWS-Services wie Elastic Load Balancing und AWS Auto Scaling die Last auf fehlerfreie Availability Zones verteilen während Services wie z. B, Amazon Route 53 und AWS Global Accelerator, die Last an fehlerfreie AWS-Regionen leiten können. Amazon Route 53 Application Recovery Controller hilft Ihnen, mithilfe von Bereitschaftsprüfungen und Routing-Steuerungsfunktionen Failover-Vorgänge zu verwalten und zu koordinieren. Diese Funktionen überwachen kontinuierlich die Fähigkeit Ihrer Anwendung, eine Wiederherstellung nach Fehlern durchzuführen, so dass Sie die Wiederherstellung der Anwendung über mehrere AWS-Regionen, Availability Zones und On-Premises kontrollieren können.

Für Workloads in bestehenden physischen oder virtuellen Rechenzentren oder privaten Clouds, AWS Elastic Disaster Recovery,verfügbar durch AWS Marketplace, ermöglicht es Unternehmen, eine automatisierte Notfallwiederherstellungsstrategie auf AWS einzurichten. CloudEndure unterstützt auch die regions- bzw. AZ-übergreifende Notfallwiederherstellung in AWS.

Gängige Antimuster:

- Die Implementierung von identischem automatisiertem Failover und Failback kann bei einem Fehler zu Flapping führen.

Vorteile der Einführung dieser bewährten Methode: Die automatisierte Wiederherstellung verkürzt die Wiederherstellungszeit, da manuelle Fehler nicht mehr möglich sind.

Risikostufe, wenn diese bewährte Methode nicht eingeführt wird: Mittel

Implementierungsleitfaden

- Automatisieren von Wiederherstellungspfaden. Wenn in Szenarien mit hoher Verfügbarkeit kurze Wiederherstellungszeiten erforderlich sind, sind menschliche Beurteilungen und Aktionen

zu langsam. Das System sollte in jeder Situation in der Lage sein, eine Wiederherstellung durchzuführen.

- Verwenden Sie CloudEndure Disaster Recovery für automatisiertes Failover und Failback. CloudEndure Disaster Recovery repliziert Ihre Computer (einschließlich Betriebssystem, Systemstatuskonfiguration, Datenbanken, Anwendungen und Dateien) kontinuierlich in einen kostengünstigen Staging-Bereich in Ihrem AWS-Konto-Zielkonto und in Ihrer bevorzugten Region. Bei einem Notfall können Sie CloudEndure Disaster Recovery anweisen, innerhalb weniger Minuten automatisch Tausende Ihrer virtuellen Maschinen vollständig bereitgestellt zu starten.
  - Ausführen von Failover und Failback bei Notfallwiederherstellungen
  - CloudEndure Disaster Recovery

Ressourcen

Zugehörige Dokumente:

- APN-Partner: Partner, die Sie bei der Notfallwiederherstellung unterstützen können
- AWS Architecture Blog: Notfallwiederherstellungsserie
- AWS Marketplace: Für die Notfallwiederherstellung geeignete Produkte
- AWS Systems Manager Automation
- CloudEndure Disaster Recovery auf AWS
- Notfallwiederherstellung von Workloads auf AWS: Wiederherstellung in der Cloud (AWS-Whitepaper)

Relevante Videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2) (Architekturmuster für Aktiv-Aktiv-Anwendungen in mehreren Regionen)

Beispielhafte Implementierungen für Verfügbarkeitsziele

In diesem Abschnitt werfen wir einen Blick auf Workload-Designs zur Bereitstellung einer typischen Web-Anwendung, die aus einem Reverse-Proxy, aus statischen Inhalten auf Amazon S3, einem Anwendungsserver und einer SQL-Datenbank für die dauerhafte Datenspeicherung besteht. Die einzelnen Verfügbarkeitsziele werden durch eine beispielhafte Implementierung dargestellt. Dieser Workload könnte stattdessen Container oder AWS Lambda für die Datenverarbeitung und NoSQL (z. B. Amazon DynamoDB) für die Datenbank verwenden, aber die Ansätze sind ähnlich. In jedem Szenario zeigen wir, wie Sie die Verfügbarkeitsziele durch Workload-Design für die folgenden Themen erreichen:

Thema	Weitere Informationen finden Sie in diesem Kapitel
Überwachung von Ressourcen	Überwachen von Workload-Ressourcen
Anpassungen aufgrund von Bedarfsänderungen durchführen	Entwerfen einer Workload, die sich an Bedarfsänderungen anpasst
Implementierung von Änderungen	Implementierung von Änderungen
Daten sichern	Daten sichern
Ausfallsichere Architekturen	Schützen von Workloads durch Fehlerisolierung Entwerfen von Workloads, die Komponentenausfälle verkraften
Ausfallsicherheit testen	Testen der Zuverlässigkeit
Planung der Notfallwiederherstellung	Planung der Notfallwiederherstellung

Abhängigkeitsauswahl

Wir haben entschieden, Amazon EC2 für unsere Anwendungen zu verwenden. Wir zeigen Ihnen, wie Sie mit der Verwendung von Amazon RDS und mehrerer Availability Zones die Verfügbarkeit

unserer Anwendungen verbessern können. Wir verwenden Amazon Route 53 für DNS. Wenn wir mehrere Availability Zones verwenden, verwenden wir Elastic Load Balancing. Amazon S3 wird für Sicherungen und statische Inhalte verwendet. Wenn wir ein Design für höhere Verfügbarkeit aufbauen, müssen wir Services verwenden, die ihrerseits eine höhere Verfügbarkeit bieten. Transparenz Anhang A: Konzipiert für Verfügbarkeit ausgewählter AWS-Services für die Designziele der jeweiligen AWS-Services.

Szenarien mit einer Region

Themen

Szenario mit zwei Neunen (99 %)

Diese Workloads sind für das Unternehmen hilfreich, aber es ist nicht kritisch, wenn sie nicht verfügbar sind. Bei dieser Art von Workload kann es sich um interne Tools, internes Wissensmanagement oder Projektverfolgung handeln. Oder es kann sich um tatsächliche kundenorientierte Workloads handeln, die aber von einem experimentellen Service gespeist werden, der bei Bedarf ausgeblendet werden kann.

Diese Workloads können in einer Region und einer Availability Zone bereitgestellt werden.

Überwachung von Ressourcen

Wir nutzen ein einfaches Überwachungsverfahren, aus dem hervorgeht, ob die Service-Startseite einen Status der Art "HTTP 200 OK" ausgibt. Bei Problemen zeigt unser Playbook an, dass eine Protokollierung der Instance angewendet wird, um der Ursache auf den Grund zu gehen.

Anpassungen aufgrund von Bedarfsänderungen durchführen

Wir stellen Playbooks für häufige Hardware-Fehler, dringende Software-Aktualisierungen und weitere störende Änderungen zur Verfügung.

Implementierung von Änderungen

Wir verwenden AWS CloudFormation, um unsere Infrastruktur als Code zu definieren und insbesondere, um den Neuaufbau im Falle eines Ausfalls zu beschleunigen.

Software-Aktualisierungen erfolgen manuell über ein Runbook, wobei Ausfallzeiten für die Installation und den Neustart des Services berücksichtigt werden sollten. Wenn während der Bereitstellung ein Problem auftritt, beschreibt das Runbook den Rollback auf eine vorherige Version.

Alle Korrekturen des Fehlers werden mithilfe der Analyse der Protokolle durch die Betriebs- und Entwicklungsteams durchgeführt. Die Korrektur wird durchgeführt, nachdem die Behebungsmaßnahme priorisiert und abgeschlossen wurde.

Daten sichern

Wir verwenden einen Anbieter oder eine spezielle Sicherungslösung, um verschlüsselte Sicherungsdaten über ein Runbook an Amazon S3 zu senden. Wir testen die Funktion dieser Sicherungen durch die Wiederherstellung der Daten und die Gewährleistung der Fähigkeit, dass sie regelmäßig mit einem Runbook verwendet werden können. Wir konfigurieren die Versionskontrolle für unsere Amazon S3-Objekte und entfernen Berechtigungen für das Löschen der Sicherungen. Außerdem verwenden wir eine Amazon S3-Bucket-Lebenszyklusrichtlinie, um Archivierungs- oder Löschvorgänge gemäß unseren Anforderungen durchzuführen.

Ausfallsichere Architekturen

Workloads werden in einer Region und einer Availability Zone bereitgestellt. Wir stellen die Anwendung, einschließlich der Datenbank, in einer einzelnen Instance bereit.

Ausfallsicherheit testen

Die Bereitstellungspipeline neuer Software wird terminiert und es wird Zeit für Gerätetests eingeplant, der Schwerpunkt liegt jedoch auf White-Box/Black-Box-Tests des erstellten Workloads.

Planung der Notfallwiederherstellung

Bei Ausfällen warten wir den Ausfall ab und leiten optional Anfragen über eine DNS-Änderung per Runbook an eine statische Website weiter. Die Wiederherstellungszeit für diesen Vorgang wird durch die Geschwindigkeit bestimmt, bei der die Infrastruktur bereitgestellt und die Datenbank auf die neueste Sicherung wiederhergestellt werden. Diese Bereitstellung kann bei einem Ausfall

einer Availability Zone über ein Runbook entweder in die gleiche Availability Zone oder eine andere Availability Zone vollzogen werden.

Verfügbarkeitsdesignziel

Es dauert etwa 30 Minuten, um den Sachverhalt zu verstehen und die Entscheidung zur Wiederherstellung zu treffen. Es dauert weitere 10 Minuten, um den gesamten Stack in AWS CloudFormation bereitzustellen, mit der Annahme, dass die Bereitstellung in einer neuen Availability Zone erfolgt und die Datenbank innerhalb von 30 Minuten wiederhergestellt werden kann. Dies bedeutet, dass es etwa 70 Minuten dauern wird, das System nach einem Ausfall wiederherzustellen. Ausgehend von einem Ausfall pro Quartal liegt die geschätzte zeitliche Auswirkung pro Jahr bei 280 Minuten oder vier Stunden und 40 Minuten.

Damit liegen wir am oberen Ende einer Verfügbarkeit von 99,9 %. Die tatsächliche Verfügbarkeit richtet sich auch nach der tatsächlichen Ausfallrate, der Dauer eines Ausfalls und wie schnell jeder Ausfall tatsächlich behoben werden kann. Bei dieser Architektur muss die Anwendung für Aktualisierungen offline genommen werden (mit einem Schätzwert von 24 Stunden pro Jahr, also vier Stunden pro Aktualisierung bei sechs Vorgängen pro Jahr); hinzu kommen tatsächliche Ereignisse. Bezogen auf die weiter oben in diesem Whitepaper genannte Tabelle zur Anwendungsverfügbarkeit sehen wir, dass unser Verfügbarkeitsdesignziel bei 99 % liegt.

Zusammenfassung

Thema	Implementierung
Überwachung von Ressourcen	Nur standortbasierte Zustandsüberprüfung, keine Alarme
Anpassungen aufgrund von Bedarfsänderungen durchführen	Vertikale Skalierung über Neu-Bereitstellungen
Implementierung von Änderungen	Runbook für Bereitstellungen und Rollbacks
Daten sichern	Runbook für Sicherung und Wiederherstellung.
Ausfallsichere Architekturen	Vollständiger Neuaufbau, Wiederherstellung aus Sicherung.

Thema	Implementierung
Ausfallsicherheit testen	Vollständiger Neuaufbau, Wiederherstellung aus Sicherung.
Planung der Notfallwiederherstellung	Verschlüsselte Sicherungen, ggf. Wiederherstellung in andere Availability Zone

Szenario mit drei Neunen (99,9 %)

Das nächste Verfügbarkeitsziel bezieht sich auf Anwendungen, für die Hochverfügbarkeit wichtig ist, die jedoch kürzere Zeiträume der Nichtverfügbarkeit tolerieren können. Dieser Workload-Typ wird in der Regel für interne Prozesse verwendet, deren Ausfall Mitarbeiter beeinträchtigen. Derartige Workloads können auch für Kunden verwendet werden, generieren aber keine großen Umsätze und tolerieren eine längere Wiederherstellungsdauer oder einen längeren Wiederherstellungspunkt. Solche Workloads umfassen administrative Anwendungen für die Konto- oder Informationsverwaltung.

Wir können die Verfügbarkeit für Workloads durch die Nutzung von zwei Availability Zones für unsere Bereitstellung und das Trennen der Anwendungen in separate Stufen verbessern.

Überwachung von Ressourcen

Die Überwachung wird erweitert, um eine Warnung in Bezug auf die Verfügbarkeit der gesamten Website auszugeben. Dazu wird die Startseite einer Prüfung auf den Status „HTTP 200 OK“ unterzogen. Außerdem gibt es einen Alarm bei jedem Webserver-Austausch und Datenbank-Failover. Wir überwachen außerdem die statischen Inhalte auf Amazon S3 in Bezug auf Verfügbarkeit und geben einen Alarm aus, falls die Inhalte nicht mehr verfügbar sind. Die Protokollierung wird zugunsten einer einfacheren Verwaltung und zur Unterstützung der Ursachenanalyse aggregiert.

Anpassungen aufgrund von Bedarfsänderungen durchführen

Auto Scaling ist so konfiguriert, dass die CPU-Auslastung auf EC2-Instances überwacht und Instances hinzugefügt oder entfernt werden, um die CPU-Auslastung bei 70 % zu halten, jedoch mit nicht weniger als einer EC2-Instance pro Availability Zone. Wenn Lastmuster auf unserer RDS-Instance darauf hinweisen, dass eine Skalierung erforderlich ist, ändern wir den Instance-Typ während eines Wartungszeitraums.

Implementierung von Änderungen

Die Technologien für die Infrastrukturbereitstellung bleiben im Vergleich zum vorherigen Szenario unverändert.

Die Auslieferung neuer Software erfolgt alle zwei bis vier Wochen auf Basis eines festen Zeitplans. Software-Aktualisierungen werden automatisiert, jedoch nicht über Canary- oder Blue-/Green-Bereitstellungsmodelle, sondern über das integrierte Ersetzen. Die Entscheidung für oder gegen ein Rollback erfolgt über das Runbook.

Wir werden Playbooks zur Ursachenermittlung aufstellen. Nachdem die Ursache identifiziert wurde, wird die Korrektur des Fehlers über eine Kombination aus Teams in Betrieb und Entwicklung identifiziert. Die korrigierte Version wird bereitgestellt, nachdem die Behebung entwickelt wurde.

Daten sichern

Sicherungen und Wiederherstellungen können über Amazon RDS erfolgen. Diese Schritte werden regelmäßig auf Basis eines Runbooks ausgeführt, um sicherzustellen, dass wir die Wiederherstellungsanforderungen erfüllen.

Ausfallsichere Architekturen

Wir können die Verfügbarkeit für Anwendungen durch die Nutzung von zwei Availability Zones für unsere Bereitstellung und das Trennen der Anwendungen in separate Stufen verbessern. Wir verwenden Services, die auf mehreren Availability Zones funktionieren, z. B. Elastic Load Balancing, Auto Scaling und Amazon RDS Multi-AZ mit verschlüsseltem Speicher über den AWS Key Management Service. Damit gewährleisten wir Toleranz bei Ausfällen auf der Ressourcenebene und der Ebene der Availability Zone.

Der Load Balancer leitet Datenverkehr nur an funktionsfähige Anwendungs-Instances weiter. Die Zustandsüberprüfung muss auf der Datentransport-/Anwendungsebene erfolgen und die Leistungsfähigkeit der Anwendung auf der Instance anzeigen. Diese Prüfung sollte sich nicht auf die Signalisierung beziehen. Eine Zustandsüberprüfungs-URL für die Web-Anwendung ist verfügbar und für Verwendung durch den Load Balancer und Auto Scaling konfiguriert, sodass Instances, die ausfallen, entfernt und ersetzt werden. Amazon RDS verwaltet die aktive Datenbank-Engine, damit sie in der sekundären Availability Zone verfügbar ist, falls die Instance in der primären Availability Zone ausfällt, und führt eine Reparatur aus, um sie mit derselben Ausfallsicherheit wiederherzustellen.

Nachdem wir die Stufen getrennt haben, können wir Ausfallsicherheitsmodelle für verteilte Systeme verwenden, um die Zuverlässigkeit der Anwendung zu erhöhen, sodass sie auch dann verfügbar ist, wenn die Datenbank während eines Availability Zone-Failovers vorübergehend nicht verfügbar ist.

Ausfallsicherheit testen

Wir führen Funktionstests wie im vorherigen Szenario durch. Wir testen die Selbstreparaturfunktionen von ELB, Auto Scaling oder RDS Failover nicht.

Wir verfügen über Playbooks für allgemeine Datenbankprobleme, sicherheitsbezogene Vorfälle und fehlgeschlagene Bereitstellungen.

Planung der Notfallwiederherstellung

Es sind Runbooks für die vollständige Workload-Wiederherstellung und allgemeine Berichte verfügbar. Die Wiederherstellung erfolgt über Sicherungen, die in derselben Region wie der Workload gespeichert sind.

Verfügbarkeitsdesignziel

Wir gehen davon aus, dass mindestens bei einigen Ausfällen eine manuelle Entscheidung zur Ausführung der Wiederherstellung erforderlich ist. Bei der größeren Automatisierung in diesem Szenario gehen wir allerdings davon aus, dass nur bei zwei Ereignissen pro Jahr eine solche Entscheidung erforderlich ist. Wir benötigen 30 Minuten, um über die Ausführung der Wiederherstellung zu entscheiden, und gehen davon aus, dass die Wiederherstellung innerhalb von 30 Minuten abgeschlossen ist. Außerdem entfallen 60 Minuten auf die Wiederherstellung nach einem Ausfall. Ausgehend von zwei Vorfällen pro Jahr liegt die geschätzte zeitliche Auswirkung für das Jahr bei 120 Minuten.

Damit liegen wir am oberen Ende einer Verfügbarkeit von 99,95 %. Die tatsächliche Verfügbarkeit richtet sich auch nach der tatsächlichen Ausfallrate, der Dauer eines Ausfalls und wie schnell nach jedem Ausfall tatsächlich wiederhergestellt werden kann. Bei dieser Architektur muss die Anwendung kurz für Aktualisierungen offline gesetzt werden, diese Aktualisierungen werden jedoch automatisiert. Wir gehen dabei von 150 Minuten pro Jahr aus: 15 Minuten pro Änderung bei zehn Änderungen pro Jahr. Dies summiert sich auf 270 Minuten pro Jahr, in denen der Service nicht verfügbar ist. Unser Verfügbarkeitsdesignziel liegt also bei 99,9 %.

Zusammenfassung

Thema	Implementierung
Überwachung von Ressourcen	Nur Zustandsüberprüfung am Standort, Alarme werden bei Ausfall versendet.
Anpassungen aufgrund von Bedarfsänderungen durchführen	ELB für die Web- und Auto Scaling-Anwendungsstufe, Änderung der Dimensionierung von Multi-AZ RDS.
Implementierung von Änderungen	Automatisierte Bereitstellung vorhanden und Runbook für Rollback.
Daten sichern	Automatisierte Sicherungen über RDS zur Erfüllung des RPO und Runbook für Wiederherstellung.
Ausfallsichere Architekturen	Auto Scaling für die selbstreparierende Web- und Anwendungsstufe, RDS ist Multi-AZ.
Ausfallsicherheit testen	ELB und Anwendung sind selbstheilend, RDS ist Multi-AZ, kein explizites Testen.
Planung der Notfallwiederherstellung	Verschlüsselte Sicherungen über RDS in dieselbe AWS-Region.

Szenario mit vier Neunen (99,99 %)

Bei diesem Verfügbarkeitsziel für Anwendungen muss die Anwendung hochgradig verfügbar sein und Komponentenausfälle tolerieren. Die Anwendung muss in der Lage sein, Ausfälle ohne den Abruf zusätzlicher Ressourcen zu verkraften. Dieses Verfügbarkeitsziel eignet sich für geschäftskritische Anwendungen, bei denen es sich um zentrale oder signifikante Umsatzquellen für ein Unternehmen handelt, z. B. eine E-Commerce-Website, einen B2B-Webservice oder eine Content-Medienhosting-Website mit hohem Datenverkehr.

Wir können die Verfügbarkeit durch die Nutzung einer Architektur weiter verbessern, die innerhalb einer Region statisch stabil ist. Für diese Verfügbarkeit ist keine Änderung bei der Signalisierung

in Bezug auf das Verhalten unseres Workloads erforderlich, um den Ausfall zu tolerieren. Es muss beispielsweise ausreichend Kapazität vorhanden sein, um den Verlust einer Availability Zone zu verkraften. Es sollten keine Aktualisierungen für Amazon Route 53 DNS erforderlich sein. Es sollte nicht erforderlich sein, eine neue Infrastruktur zu erstellen, ob für das Erstellen oder Ändern eines S3-Buckets, das Erstellen neuer IAM-Richtlinien (oder das Ändern von Richtlinien) oder das Ändern der Amazon ECS-Aufgabenkonfigurationen.

Überwachung von Ressourcen

Die Überwachung umfasst Erfolgskennzahlen sowie Alarme beim Auftreten von Problemen. Außerdem gibt es einen Alarm bei Austausch eines ausgefallenen Webservers, bei einem Datenbank-Failover und wenn eine AZ ausfällt.

Anpassungen aufgrund von Bedarfsänderungen durchführen

Wir verwenden Amazon Aurora als RDS, was die automatische Skalierung von Lesereplikaten ermöglicht. Bei diesen Anwendungen ist die Entwicklung für Lese-Verfügbarkeit im Vergleich zur Schreib-Verfügbarkeit für primäre Inhalte ebenfalls eine wichtige Architekturentscheidung. Außerdem kann Aurora bei Bedarf den Speicher automatisch in 10 GB-Schritten auf bis zu 64 TB vergrößern.

Implementierung von Änderungen

Wir stellen Aktualisierungen über Canary- oder Blue-/Green-Bereitstellungen separat in jeder Isolationszone bereit. Die Bereitstellungen sind vollständig automatisiert, einschließlich eines Rollbacks, falls KPIs auf ein Problem hinweisen.

Es sind Runbooks für das strikte Berichten von Anforderungen und der Leistungsverfolgung vorhanden. Wenn erfolgreiche Vorgänge einen Trend für das Verfehlen von Leistungs- oder Verfügbarkeitszielen erkennen lassen, wird ein Playbook verwendet, um zu ermitteln, was den Trend auslöst. Playbooks sind für nicht erkannte Fehlermodi und Sicherheitsvorfälle verfügbar. Playbooks sind außerdem für die Ermittlung der Fehlerursache verfügbar. Für das Infrastructure Event Management-Angebot nehmen wir außerdem Kontakt zum AWS Support auf.

Das für die Entwicklung und den Betrieb der Website zuständige Team identifiziert die Fehlerbehebung bei unerwarteten Ausfällen und priorisiert die bereitzustellende korrigierte Version nach der Implementierung der Fehlerbehebung.

Daten sichern

Sicherungen und Wiederherstellungen können über Amazon RDS erfolgen. Diese Schritte werden regelmäßig auf Basis eines Runbooks ausgeführt, um sicherzustellen, dass wir die Wiederherstellungsanforderungen erfüllen.

Ausfallsichere Architekturen

Wir empfehlen drei Availability Zones für diesen Ansatz. Bei einer Bereitstellung mit drei Availability Zones hat jede AZ in Spitzenzeiten eine statische Kapazität von 50 %. Es sollten zwei Availability Zones verwendet werden, die Kosten der statischen stabilen Kapazität würden jedoch steigen, da beide AZs über 100 % Kapazität in Spitzenzeiten verfügen müssten. Wir fügen Amazon CloudFront für geografisches Caching hinzu und fordern die Reduzierung des Datentransports für unsere Anwendung an.

Wir verwenden Amazon Aurora als RDS und stellen Lesereplikate in allen drei Zonen bereit.

Die Anwendung wird auf allen Ebenen unter Verwendung von Ausfallsicherheitsmodellen für Software/Anwendungen aufgebaut.

Ausfallsicherheit testen

Die Bereitstellungspipeline enthält eine umfassende Test-Suite, einschließlich Tests auf Leistung, Last und Fehlerinjektion.

Wir testen unsere Fehlerwiederherstellungsverfahren regelmäßig im Rahmen von Gamedays und nutzen Runbooks, um sicherzustellen, dass wir die erforderlichen Aufgaben erfüllen können und nicht von den Verfahren abweichen. Das Team, das die Website aufbaut, ist auch für den Betrieb der Website zuständig.

Planung der Notfallwiederherstellung

Es sind Runbooks für die vollständige Workload-Wiederherstellung und allgemeine Berichte verfügbar. Die Wiederherstellung erfolgt über Sicherungen, die in derselben Region wie der Workload gespeichert sind. Wiederherstellungsverfahren werden regelmäßig als Teil der Gamedays durchgeführt.

Verfügbarkeitsdesignziel

Wir gehen davon aus, dass zumindest bei einigen Fehlern eine manuelle Entscheidung zur Ausführung einer Wiederherstellung getroffen werden muss. Durch den größeren Anteil an

Automatisierung in diesem Szenario gehen wir jedoch davon aus, dass diese Entscheidung nur bei zwei Ereignissen pro Jahr getroffen werden muss und die Wiederherstellungsaktionen zügig erfolgen. Wir benötigen 10 Minuten, um über die Ausführung der Wiederherstellung zu entscheiden, und gehen davon aus, dass die Wiederherstellung innerhalb von 5 Minuten abgeschlossen ist. Außerdem entfallen 15 Minuten auf die Wiederherstellung nach einem Ausfall. Ausgehend von zwei Ausfällen pro Jahr liegt die geschätzte zeitliche Auswirkung für das Jahr bei 30 Minuten.

Damit liegen wir am oberen Ende einer Verfügbarkeit von 99,99 %. Die tatsächliche Verfügbarkeit richtet sich auch nach der tatsächlichen Ausfallrate, der Dauer eines Ausfalls und wie schnell jeder Faktor tatsächlich wiederhergestellt werden kann. Bei dieser Architektur gehen wir davon aus, dass die Anwendung auch während der Implementierung von Aktualisierungen online ist. Entsprechend liegt unser Verfügbarkeitsdesignziel bei 99,99 %.

Zusammenfassung

Thema	Implementierung
Überwachung von Ressourcen	Zustandsüberprüfungen auf allen Ebenen und für alle KPIs, Versand von Alarmen, nachdem konfigurierte Alarme ausgelöst wurden, Alarme bei allen Fehlern. Im Rahmen betrieblicher Meetings können Trends rigoros aufgedeck t und mit Designzielen in Einklang gebracht werden.
Anpassungen aufgrund von Bedarfsän derungen durchführen	ELB für Web- und Auto Scaling-Anwendungs ebene; Auto Scaling-Speicher und Leserepli kate in mehreren Zonen für Aurora RDS.
Implementierung von Änderungen	Die automatisierte Bereitstellung über Canary oder Blue/Green und das automatisierte Rollback, wenn KPIs oder Alarme auf nicht erkannte Probleme in einer Anwendung hinweisen. Bereitstellungen erfolgen nach Isolationszone.
Daten sichern	Automatisierte Sicherungen über RDS zur Erfüllung des RPO und automatisierte

Thema	Implementierung
	Wiederherstellung, die regelmäßig im Rahmen von Gamedays getestet wird.
Ausfallsichere Architekturen	Implementierte Fehlerisolationszonen für die Anwendung, Auto Scaling für die Bereitstellung von selbstheilenden Web- und Anwendung sstufen, RDS ist Multi-AZ.
Ausfallsicherheit testen	Das Testen von Fehlern in Komponenten und Isolationszonen ist in der Pipeline enthalten und wird im Rahmen von Gamedays regelmäßig durch Teams im Betrieb ausgeführt, Playbooks sind für die Diagnose unbekannter Probleme vorhanden, und es ist ein Prozess für die Ursachenanalyse vorhanden.
Planung der Notfallwiederherstellung	Verschlüsselte Sicherungen über RDS in dieselbe AWS-Region, in der im Rahmen von Gamedays getestet wird.

Szenarien mit mehreren Regionen

Die Implementierung unserer Anwendung in mehreren AWS-Regionen erhöht die Betriebskosten, und zwar teilweise deshalb, weil wir Regionen für die Aufrechterhaltung ihrer Autonomie isolieren. Es sollte genau überlegt werden, ob es sinnvoll ist, diesen Pfad zu verfolgen. Regionen bieten eine starke Isolationsbegrenzung, und wir unternehmen große Anstrengungen, korrelierte Fehler in anderen Regionen zu vermeiden. Durch die Verwendung mehrerer Regionen erhalten Sie mehr Kontrolle über Ihre Wiederherstellungszeit, wenn ein harter Abhängigkeitsfehler bei einem regionalen AWS-Service auftritt. In diesem Abschnitt besprechen wir verschiedene Implementierungsmodelle mit ihren typischen Verfügbarkeiten.

Themen

3,5 Neunen (99,95 %) mit einer Wiederherstellungszeit zwischen 5 und 30 Minuten

Dieses Verfügbarkeitsziel für Anwendungen setzt eine extrem kurze Ausfallzeit und einen sehr geringen Datenverlust zu bestimmten Zeiten voraus. Zu Anwendungen mit diesem Verfügbarkeitsziel gehören Anwendungen in den folgenden Bereichen: Bankwesen, Investments, Notfallservices und Datenerfassung. Diese Anwendungen weisen sehr kurze Wiederherstellungszeiten und Wiederherstellungspunkte auf.

Wir können diese Wiederherstellungszeit weiter verbessern, indem wir einen Warm Standby -Ansatz in zwei AWS-Regionen nutzen. Wir stellen den gesamten Workload in beiden Regionen bereit, wobei unsere passive Website herunterskaliert und alle Daten letztendlich konsistent gehalten werden. Beide Bereitstellungen sind in ihren jeweiligen Regionen statisch stabil. Die Anwendungen sollten unter Verwendung von Ausfallsicherheitsmodellen für verteilte Systeme entwickelt werden. Wir müssen eine einfache Routing -Komponente erstellen, die den Workload-Zustand überwacht und so konfiguriert werden kann, dass der Datenverkehr bei Bedarf an die passive Region weitergeleitet wird.

Überwachung von Ressourcen

Außerdem gibt es einen Alarm bei jedem Austausch eines Webservers sowie bei jedem Datenbank- und Regions-Failover. Wir überwachen außerdem die statischen Inhalte auf Amazon S3 in Bezug auf Verfügbarkeit und geben einen Alarm aus, falls die Inhalte nicht mehr verfügbar sind. Die Protokollierung wird zugunsten einer einfacheren Verwaltung und zur Unterstützung der Ursachenanalyse in jeder Region aggregiert.

Die Routing-Komponente überwacht sowohl den Anwendungsstatus als auch alle regionalen harten Abhängigkeiten, die wir haben.

Anpassungen aufgrund von Bedarfsänderungen durchführen

Identisch mit dem Szenario mit 4 Neunen.

Implementierung von Änderungen

Die Auslieferung neuer Software erfolgt alle zwei bis vier Wochen auf Basis eines festen Zeitplans. Software-Aktualisierungen werden über Canary- oder Blue-/Green-Bereitstellungsmodelle automatisiert.

Es sind Runbooks bei einem Failover in einer Region, bei allgemeinen Kundenproblemen, die im Rahmen dieser Ereignisse auftreten, und für das allgemeine Berichtswesen verfügbar.

Wir verfügen über Playbooks für allgemeine Datenbankprobleme, sicherheitsbezogene Vorfälle, fehlgeschlagene Bereitstellungen, unerwartete Kundenprobleme bei einem Failover in einer Region und der Ermittlung von Problemursachen. Nachdem die Ursache identifiziert wurde, wird die Fehlerbehebung über eine Kombination aus Teams in Betrieb und Entwicklung identifiziert und bereitgestellt, sobald die Fehlerbehebung implementiert wurde.

Für das Infrastructure Event Management nehmen wir außerdem Kontakt zum AWS Support auf.

Daten sichern

Wie im Szenario mit 4 Neunen werden automatische RDS-Sicherungen und die S3-Versionsverwaltung verwendet. Die Daten werden automatisch und asynchron aus dem Aurora RDS-Cluster in der aktiven Region in regionsübergreifende Lesereplikate in der passiven Region repliziert. Die regionsübergreifende S3-Replikation wird verwendet, um Daten automatisch und asynchron aus der aktiven in die passive Region zu verschieben.

Ausfallsichere Architekturen

Wie im Szenario mit 4 Neunen, regionales Failover möglich. Dies wird manuell verwaltet. Während des Failovers leiten wir Anfragen über einen DNS-Failover an eine statische Website weiter, bis die Wiederherstellung in der sekundären Region erfolgreich war.

Ausfallsicherheit testen

Wie im Szenario mit 4 Neunen validieren wir die Architektur im Rahmen von Ernstfallübungen und unter Verwendung von Runbooks. Der Korrektur auf Basis der Ursachenanalyse wird zudem im Vergleich zu Funktionsveröffentlichungen für die unmittelbare Implementierung und Bereitstellung Vorrang eingeräumt.

Planung der Notfallwiederherstellung

Regionales Failover wird manuell verwaltet. Alle Daten werden asynchron repliziert. Die Infrastruktur im Warm Standby wird skaliert. Dies kann mithilfe eines in AWS Step Functions ausgeführten Workflows automatisiert werden. AWS Systems Manager (SSM) kann auch bei dieser Automatisierung helfen, da Sie SSM-Dokumente erstellen können, die Auto Scaling-Gruppen aktualisieren und die Größe von Instances anpassen.

Verfügbarkeitsdesignziel

Wir gehen davon aus, dass zumindest bei einigen Fehlern eine manuelle Entscheidung zur Ausführung einer Wiederherstellung getroffen werden muss. Mit guter Automatisierung in diesem Szenario gehen wir jedoch davon aus, dass diese Entscheidung nur bei zwei Ereignissen pro Jahr getroffen werden muss. Wir benötigen 20 Minuten, um über die Ausführung der Wiederherstellung zu entscheiden, und gehen davon aus, dass die Wiederherstellung innerhalb von 10 Minuten abgeschlossen ist. Dies bedeutet, dass es etwa 30 Minuten dauern wird, das System nach einem Ausfall wiederherzustellen. Ausgehend von zwei Vorfällen pro Jahr liegt die geschätzte zeitliche Auswirkung für das Jahr bei 60 Minuten.

Damit liegen wir am oberen Ende einer Verfügbarkeit von 99,95 %. Die tatsächliche Verfügbarkeit richtet sich auch nach der tatsächlichen Ausfallrate, der Dauer eines Ausfalls und wie schnell jeder Faktor tatsächlich wiederhergestellt werden kann. Bei dieser Architektur gehen wir davon aus, dass die Anwendung auch während der Implementierung von Aktualisierungen online ist. Entsprechend liegt unser Verfügbarkeitsdesignziel bei 99,95 %.

Zusammenfassung

Thema	Implementierung
Ressourcen überwachen	Zustandsprüfungen auf allen Ebenen, einschließlich DNS-Zustand auf AWS-Regionsebene, und für alle KPIs, Ausgabe von Warnmeldungen bei Auslösung konfigurierter Alarme, Warnmeldungen bei allen Fehlern. Im Rahmen betrieblicher Meetings können Trends rigoros aufgedeckt und mit Designzielen in Einklang gebracht werden.
Anpassungen aufgrund von Bedarfsänderungen durchführen	ELB für Web- und Auto Scaling-Anwendungsebene; Auto Scaling-Speicher und Leserepli kate in mehreren Zonen in den aktiven und passiven Regionen für Aurora RDS. Daten und Infrastruktur, die zwischen AWS-Regionen synchronisiert werden, um statische Stabilität zu gewährleisten.

Thema	Implementierung
Implementierung von Änderungen	Die automatisierte Bereitstellung über Canary oder Blue/Green und das automatisierte Rollback, wenn KPIs oder Alarme auf nicht erkannte Probleme in einer Anwendung hinweisen. Die Bereitstellungen erfolgen nacheinander in einer Isolationszone einer AWS-Region.
Daten sichern	Automatisierte Sicherungen in jeder AWS-Region über RDS zur Erfüllung des RPO und automatisierte Wiederherstellung, die regelmäßig im Rahmen von Gamedays getestet wird. Aurora RDS- und S3-Daten werden automatisch und asynchron von der aktiven in die passive Region repliziert.
Ausfallsichere Architekturen	Auto Scaling für die Bereitstellung von selbstreparierenden Web- und Anwendung sstufen, RDS ist Multi-AZ, regionaler Failover wird manuell verwaltet, wobei der statische Standort während des Failovers präsentiert wird.

Thema	Implementierung
Ausfallsicherheit testen	Das Testen von Fehlern in Komponenten und Isolationszonen ist in der Pipeline enthalten und wird im Rahmen von Gamedays regelmäßig durch Teams im Betrieb ausgeführt, Playbooks sind für die Diagnose unbekannter Probleme vorhanden, und es ist ein Prozess für die Ursachenanalyse vorhanden, mit Kommunikationspfaden zur Problemursache und wie das Problem behoben oder vermieden wurde. Der Korrektur der Ursachenanalyse wird im Vergleich zu Funktionsveröffentlichungen für die unmittelbare Implementierung und Bereitstellung Vorrang eingeräumt.
Planung der Notfallwiederherstellung	Warm Standby wird in einer anderen Region bereitgestellt. Die Infrastruktur wird mithilfe von Workflows skaliert, die mit AWS Step Functions oder AWS Systems Manager-Dokumenten ausgeführt werden. Verschlüsselte Sicherungen über RDS. Regionsübergreifende Leserepli kate zwischen zwei AWS-Regionen. Regionsübergreifende Replikation statischer Komponenten in Amazon S3. Wiederherstellung auf die aktuell aktive AWS-Region, wird im Rahmen von Gamedays geprobt und mit AWS koordiniert.

Szenario mit 5 Neunen (99,999 %) oder mehr mit einer Wiederherstellungszeit unter 1 Minute

Dieses Verfügbarkeitsziel für Anwendungen setzt Ausfallzeiten oder Datenverlust für bestimmte Zeiten nahe null voraus. Anwendungen mit diesem Verfügbarkeitsziel könnten beispielsweise in den Bereichen Bankwesen, Investments, Finanzen, Behördenwesen und in geschäftskritischen

Anwendungen zu finden sein, die das Kerngeschäft eines extrem umsatzgenerierenden Geschäfts bilden. Hier liegt der Wunsch bei stark konsistenten Datenspeichern und einer vollständigen Redundanz auf allen Ebenen. Wir haben einen SQL-basierten Datenspeicher ausgewählt. In einigen Szenarios ist es u. U. schwierig, ein sehr kurzes RPO zu erreichen. Wenn Sie Ihre Daten partitionieren können, ist es möglich, einen Datenverlust von null zu erreichen. Dazu müssen Sie ggf. Anwendungslogik und Latenz hinzufügen, um sicherzustellen, dass Sie konsistente Daten zwischen geografischen Standorten nutzen können und in der Lage sind, Daten zwischen Partitionen zu verschieben oder zu kopieren. Eine solche Partitionierung ist u. U. einfacher, wenn Sie eine NoSQL-Datenbank verwenden.

Durch die Nutzung eines Aktiv-Aktiv -Ansatzes über mehrere AWS-Regionen hinweg können wir die Verfügbarkeit weiter verbessern. Der Workload wird in allen gewünschten Regionen bereitgestellt, die regionsübergreifend statisch stabil sind (sodass die restlichen Regionen die Last mit dem Verlust einer Region bewältigen können). A Routing -Ebene leitet den Datenverkehr an geografische Standorte weiter, die fehlerfrei sind, ändert automatisch das Ziel, wenn ein Standort fehlerhaft ist, und hält Datenreplikationsebenen vorübergehend an. Amazon Route 53 bietet Zustandsüberprüfungen mit einem 10-Sekunden-Intervall und außerdem TTL für Ihre Datensätze mit einem Wert von weniger als einer Sekunde.

Überwachung von Ressourcen

Entspricht dem Szenario mit 3,5 Neunen zzgl. Alarmen und Traffic-Umleitung, wenn eine Region als fehlerhaft erkannt wird.

Anpassungen aufgrund von Bedarfsänderungen durchführen

Entspricht dem Szenario mit 3,5 Neunen.

Implementierung von Änderungen

Die Bereitstellungs-Pipeline enthält eine umfassende Test-Suite, einschließlich Tests auf Leistung, Last und Fehlerinjektion. Wir stellen Aktualisierungen mit Canary- oder Blue-/Green-Bereitstellungen zunächst in einer Isolationszone bzw. einer Region bereit, bevor wir zu einer anderen Region übergehen. Im Zuge der Bereitstellung werden die alten Versionen für ein schnelleres Rollback weiterhin in Instances ausgeführt. Diese sind vollständig automatisiert, einschließlich eines Rollbacks, falls KPIs auf ein Problem hinweisen. Die Überwachung umfasst Erfolgskennzahlen sowie Alarme beim Auftreten von Problemen.

Es sind Runbooks für das strikte Berichten von Anforderungen und der Leistungsverfolgung vorhanden. Wenn erfolgreiche Vorgänge einen Trend für das Verfehlen von Leistungs- oder

Verfügbarkeitszielen erkennen lassen, wird ein Playbook verwendet, um zu ermitteln, was den Trend auslöst. Playbooks sind für nicht erkannte Fehlermodi und Sicherheitsvorfälle verfügbar. Playbooks sind außerdem für die Ermittlung der Fehlerursache verfügbar.

Das Team, das die Website aufbaut, ist auch für den Betrieb der Website zuständig. Dieses Team identifiziert die Fehlerbehebung bei unerwarteten Ausfällen und priorisiert die bereitzustellende korrigierte Version nach der Implementierung der Fehlerbehebung. Für das Infrastructure Event Management nehmen wir außerdem Kontakt zum AWS Support auf.

Daten sichern

Entspricht dem Szenario mit 3,5 Neunen.

Ausfallsichere Architekturen

Die Anwendungen sollten unter Verwendung von Ausfallsicherheitsmodellen für Software/Anwendungen aufgebaut werden. Es ist möglich, dass viele andere Routing-Ebenen erforderlich sind, um die erforderliche Verfügbarkeit zu implementieren. Die Komplexität dieser zusätzlichen Implementierung sollte nicht unterschätzt werden. Die Anwendung wird in Fehlerisolationszonen für die Bereitstellung implementiert und so partitioniert und bereitgestellt, dass sich selbst ein regionsweites Ereignis nicht auf alle Kunden auswirken wird.

Ausfallsicherheit testen

Wir validieren die Architektur im Rahmen von Gamedays und nutzen Runbooks, um sicherzustellen, dass wir die erforderlichen Aufgaben erfüllen können und nicht von den Verfahren abweichen.

Planung der Notfallwiederherstellung

Aktiv-Aktiv -Bereitstellung in mehreren Regionen mit vollständiger Workload-Infrastruktur und Daten in mehreren Regionen. Bei einer Strategie mit lokalen Lese- und globalen Schreibvorgängen ist eine Region die primäre Datenbank für alle Schreibvorgänge und Daten werden für Lesevorgänge in andere Regionen repliziert. Wenn die primäre DB-Region ausfällt, muss eine neue DB hochgestuft werden. Beim lokalen Lesen und globalem Schreiben werden Benutzer einer Stammregion zugewiesen, in der DB-Schreibvorgänge verarbeitet werden. Auf diese Weise können Benutzer aus jeder Region lesen oder schreiben, aber es ist eine komplexe Logik nötig, um potenzielle Datenkonflikte über Schreibvorgänge in verschiedenen Regionen hinweg zu verwalten.

Wenn eine Region als fehlerhaft erkannt wird, leitet die Routing-Ebene den Datenverkehr automatisch an die verbleibenden fehlerfreien Regionen weiter. Es ist kein manueller Eingriff erforderlich.

Datenspeicher müssen zwischen Regionen auf eine Art und Weise repliziert werden, die potenzielle Konflikte lösen kann. Es müssen Werkzeuge und automatisierte Prozesse erstellt werden, um Daten aus Gründen der Latenz zwischen den Partitionen zu verschieben und zu kopieren und Anfragen oder Datenmengen in jeder Partition ausgleichen zu können. Die Abhilfemaßnahme für die Datenkonfliktlösung erfordert auch zusätzliche betriebliche Runbooks.

Verfügbarkeitsdesignziel

Wir gehen davon aus, dass große Investitionen getätigt werden, um die gesamte Wiederherstellung zu automatisieren und dass die Wiederherstellung innerhalb einer Minute abgeschlossen werden kann. Wir gehen davon aus, dass keine manuell ausgelösten Wiederherstellungen ausgeführt werden, wir gehen jedoch von bis zu einer automatisierten Wiederherstellungsaktion pro Quartal aus. Dies bedeutet eine Wiederherstellungsdauer von vier Minuten pro Jahr. Wir gehen davon aus, dass die Anwendung auch während der Implementierung von Aktualisierungen online ist. Entsprechend liegt unser Verfügbarkeitsdesignziel bei 99,999 %.

Zusammenfassung

Thema	Implementierung
Ressourcen überwachen	Zustandsprüfungen auf allen Ebenen, einschließlich DNS-Zustand auf AWS-Regionsebene, und für alle KPIs, Ausgabe von Warnmeldungen bei Auslösung konfigurierter Alarme, Warnmeldungen bei allen Fehlern. Im Rahmen betrieblicher Meetings können Trends rigoros aufgedeckt und mit Designzielen in Einklang gebracht werden.
Anpassungen aufgrund von Bedarfsänderungen durchführen	ELB für Web- und Auto Scaling-Anwendungsebene; Auto Scaling-Speicher und Leserepli kate in mehreren Zonen in den aktiven und passiven Regionen für Aurora RDS. Daten und Infrastruktur, die zwischen AWS-Regionen

Thema	Implementierung
	synchronisiert werden, um statische Stabilität zu gewährleisten.
Implementierung von Änderungen	Die automatisierte Bereitstellung über Canary oder Blue/Green und das automatisierte Rollback, wenn KPIs oder Alarme auf nicht erkannte Probleme in einer Anwendung hinweisen. Die Bereitstellungen erfolgen nacheinander in einer Isolationszone einer AWS-Region.
Daten sichern	Automatisierte Sicherungen in jeder AWS-Region über RDS zur Erfüllung des RPO und automatisierte Wiederherstellung, die regelmäßig im Rahmen von Gamedays getestet wird. Aurora RDS- und S3-Daten werden automatisch und asynchron von der aktiven in die passive Region repliziert.
Ausfallsichere Architekturen	Implementierte Fehlerisolationszonen für die Anwendung, Auto Scaling für die Bereitstellung von selbstheilenden Web- und Anwendung sstufen, RDS ist Multi-AZ, regionales Failover automatisiert.

Thema	Implementierung
Ausfallsicherheit testen	Das Testen von Fehlern in Komponenten und Isolationszonen ist in der Pipeline enthalten und wird im Rahmen von Gamedays regelmäßig durch Teams im Betrieb ausgeführt, Playbooks sind für die Diagnose nicht erkannter Probleme vorhanden, und es ist ein Prozess für die Ursachenanalyse vorhanden, mit Kommunikationspfaden zur Problemursache und wie das Problem behoben oder vermieden wurde. Der Korrektur der Ursachenanalyse wird im Vergleich zu Funktionsveröffentlichungen für die unmittelbare Implementierung und Bereitstellung Vorrang eingeräumt.
Planung der Notfallwiederherstellung	Aktiv-Aktiv-Bereitstellung in mindestens zwei Regionen gleichzeitig. Die Infrastruktur ist vollständig skaliert und regionsübergreifend statisch stabil. Die Daten werden regionsübergreifend partitioniert und synchronisiert. Verschlüsselte Sicherungen über RDS. Der Ausfall von Regionen wird im Rahmen von Gamedays geprobt und mit AWS koordiniert. Während der Wiederherstellung muss möglicherweise eine neue Primärdatenbank hochgestuft werden.

Ressourcen

Dokumentation

- Die Amazon Builders’ Library – Wie Amazon Software erstellt und betreibt
- AWS-Architekturzentrum

Übungen

- AWS Well-Architected-Übungen für Zuverlässigkeit

Externe Links

- Adaptives Warteschlangenmuster: Fail at Scale
- Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS (Verfügbarkeit und mehr: Verdeutlichung und Verbesserung der Ausfallsicherheit bei verteilten Systemen in AWS)

Bücher

- Robert S. Hammer „Vorlagen für fehlertolerante Software“
- Andrew Tanenbaum und Marten van Steen „Distributed-Systeme: Prinzipien und Paradigmen“

Fazit

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 treffen können. Wir möchten Sie dazu ermuntern, die hier angebotenen Empfehlungen für Design, Betrieb und Wiederherstellung sowie das Wissen und die Erfahrung unserer AWS Solution Architects zu Ihrem Vorteil zu nutzen. Wir würden uns freuen, von Ihnen zu hören – vor allem bezüglich Ihrer Erfolgsgeschichten beim Erzielen höherer Verfügbarkeit in AWS. Wenden Sie sich an das für Sie zuständige Vertriebsteam oder setzen Sie sich über unsere Website mit uns in Verbindung.

Mitwirkende

An diesem Dokument haben folgende Personen mitgewirkt:

- Seth Eliot, Principal Developer Advocate, Amazon Web Services
- Mahanth Jayadeva, Solutions Architect – Well-Architected, Amazon Web Services
- Amulya Sharma, Principal Solutions Architect, Amazon Web Services
- Jason DiDomenico, Senior Solutions Architect – Cloud Foundations, Amazon Web Services
- Marcin Bednarz, Principal Solutions Architect, Amazon Web Services
- Tyler Applebaum, Senior Solutions Architect, Amazon Web Services
- Rodney Lester, Principal Solutions Architect – App Modernization, Amazon Web Services
- Joe Chapman, Senior Solutions Architect, Amazon Web Services
- Adrian Hornsby, Principal System Development Engineer, Amazon Web Services
- Kevin Miller, Vice President – S3, Amazon Web Services
- Shannon Richards, Principal Technical Program Manager, Amazon Web Services
- Laurent Domb, Chief Technologist - Fed Fin, Amazon Web Services
- Kevin Schwarz, Sr. Solutions Architect, Amazon Web Services
- Rob Martell, Principal Cloud Resilience Architect, Amazon Web Services
- Priyam Reddy, Senior Solutions Architect, Manager DR, Amazon Web Services
- Jeff Ferris, Principal Technologist, Amazon Web Services
- Matias Battaglia, Senior Solutions Architect, Amazon Web Services

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	Die bewährten 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 verkraften, Überwachen von Workload-Ressourcen, Entwerfen eines Workloads, der sich an Bedarfsänderungen anpasst, Implementieren von Änderungen und Testen der Zuverlässigkeit.	December 6, 2023
Leitfäden zu bewährten Methoden aktualisiert	Die bewährten Methoden wurden mit neuen Leitfäden in den folgenden Bereichen aktualisiert: Überwachen von Workload-Ressourcen und Entwerfen von Workloads, die Komponentenausfälle verkraften.	October 3, 2023
Leitfäden zu bewährten Methoden aktualisiert	Die bewährten Methoden wurden mit neuen Leitfäden in den folgenden Bereichen	July 13, 2023

	aktualisiert: Entwerfen Ihrer Download-Servicearchitektur , Entwerfen von Interaktionen in einem verteilten System, um Ausfälle abzumildern oder zu verkraften und Überwachen von Workload-Ressourcen.	
Kleineres Update	Entfernung von nicht-inklusiver Sprache	April 13, 2023
Updates für das neue Framework	Bewährte Methoden mit verbindlichen Anleitungen aktualisiert und neue bewährte Methoden hinzugefügt.	April 10, 2023
Whitepaper aktualisiert	Bewährte Methoden mit neuen Implementierungsanleitungen aktualisiert.	December 15, 2022
Kleinere Updates	Abbildungsnummern korrigier t und generell kleinere Änderungen vorgenommen.	November 17, 2022
Whitepaper aktualisiert	Weitere bewährte Methoden und Verbesserungspläne hinzugefügt.	October 20, 2022
Whitepaper aktualisiert	Zwei neue bewährte Methoden zur Säule „Zuverlässigkeit“ in den Abschnitten Fehleriso lierung zum Schutz von Workloads und Entwerfen von Workloads, die Komponent enausfälle verkraften hinzugefügt.	May 5, 2022

Kleineres Update	Säule „Nachhaltigkeit“ wurde zur Einführung hinzugefügt.	December 2, 2021
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.	October 26, 2021
Kleineres Update	Informationen zu AWS Fault Injection Service (AWS FIS) hinzugefügt.	March 15, 2021
Kleineres Update	Geringfügige Aktualisierung des Texts.	January 4, 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	December 7, 2020
Kleineres Update	Anhang A aktualisiert, um das Verfügbarkeitsdesignziel für AWS Lambda anzupassen	October 27, 2020

Kleineres Update	Anhang A aktualisiert, um das Verfügbarkeitsdesignziel für AWS Global Accelerator hinzuzufügen	July 24, 2020

Updates für das neue Framework	Erhebliche Aktualisierungen und neue/überarbeitete Inhalte, einschließlich: Hinzufügen eines Abschnitts mit bewährten Methoden zur „Workload-Architektur“, Unterteilung der bewährten Methoden in die Bereiche Änderungsmanagement und Fehlerverwaltung, Aktualisierung von Ressourcen, Aktualisierungen zur Einbeziehung der neuesten AWS-Ressourcen und -Services wie AWS Global Accelerator, AWS Service Quotas und AWS Transit Gateway, Hinzufügen/Aktualisierung von Definitionen für Zuverlässigkeit, Verfügbarkeit und Ausfallsicherheit, bessere Abstimmung des Whitepapers auf AWS Well-Architected Tool (Fragen und bewährte Methoden) für Well-Architected-Überprüfungen, Neuanordnung der Designprinzipien, Positionierung von Automatische Wiederherstellung nach einem Fehler vor Testen von Wiederherstellungsverfahren, Aktualisierung von Diagrammen und Formaten für Gleichungen, Entfernen von Abschnitten zu zentralen Services und stattdessen Einbeziehung von	July 8, 2020

	Verweisen auf zentrale AWS-Services in den bewährten Methoden.	
Kleineres Update	Fehlerhafter Link behoben	October 1, 2019
Whitepaper aktualisiert	Anhang A aktualisiert	April 1, 2019
Whitepaper aktualisiert	Spezifische AWS Direct Connect-Netzwerkem pfehlungen und zusätzliche Service-Designziele hinzugefü gt	September 1, 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.	June 1, 2018
Whitepaper aktualisiert	Regionsübergreifende DynamoDB-Lösung in DynamoDB Global Tables geändert. Service-Designziele hinzugefügt	March 1, 2018
Kleinere Updates	Geringfügige Korrektur bei der Verfügbarkeitsberechnung zur Berücksichtigung der Anwendungsverfügbarkeit	December 1, 2017

Whitepaper aktualisiert	Aktualisiert, um Hilfestellung bei Hochverfügbarkeitsdesigns zu geben, darunter Konzepte, bewährte Methoden und beispielhafte Implementierungen.	November 1, 2017
Erstveröffentlichung	Säule für Zuverlässigkeit des AWS-Well-Architected-Framework veröffentlicht.	November 1, 2016

Anhang A: Konzipiert für Verfügbarkeit ausgewählter AWS-Services

Im Folgenden wird die Verfügbarkeit angegeben, für die ausgewählte AWS-Services konzipiert wurden. Diese Werte stellen kein Service Level Agreement oder eine Garantie dar, sondern sollen Einblicke in die Designziele der einzelnen Services bieten. In bestimmten Fällen differenzieren wir Teilbereiche des Services, wenn ein aussagekräftiger Unterschied in Bezug auf das Verfügbarkeitsdesignziel besteht. Diese Liste umfasst nicht alle AWS-Services und wir werden die Liste regelmäßig mit Informationen zu weiteren Services aktualisieren. Amazon CloudFront, Amazon Route 53, AWS Global Accelerator und die AWS Identity and Access Management-Steuerebene bieten einen globalen Service und das Ziel der Komponentenverfügbarkeit wird entsprechend genannt. Weitere Services innerhalb einer AWS-Region und das Verfügbarkeitsziel werden entsprechend genannt. Viele Services werden innerhalb einer Availability Zone ausgeführt und sind getrennt von denen in anderen Availability Zones. In diesen Fällen geben wir das Verfügbarkeitsdesignziel für eine einzelne AZ an und wenn zwei (oder mehr) Availability Zones verwendet werden.

> **Note**
>
> Die Zahlen in der nachfolgenden Tabelle beziehen sich nicht auf die Datenhaltbarkeit (langfristige Aufbewahrung von Daten), sondern auf die Verfügbarkeitszahlen (Zugriff auf Daten oder Funktionen).

Service	Komponente	Verfügbarkeitsdesignziel
Amazon API Gateway	Signalisierung	99,950 %
	Datentransport	99,990 %
Amazon Aurora	Signalisierung	99,950 %
	Datentransport für einzelne AZs	99,950 %

Service	Komponente	Verfügbarkeitsdesignziel
	Datentransport für mehrere AZs	99,990 %
Amazon CloudFront	Signalisierung	99,900 %
	Datentransport (Inhaltsbereitstellung)	99,990 %
Amazon CloudSearch	Signalisierung	99,950 %
	Datentransport	99,950 %
Amazon CloudWatch	CW-Kennzahlen (Service)	99,990 %
	CW-Ereignisse (Service)	99,990 %
	CW-Protokolle (Service)	99,950 %
Amazon DynamoDB	Service (Standard)	99,990 %
	Service (globale Tabellen)	99,999 %
Amazon Elastic Block Store	Signalisierung	99,950 %
	Datentransport (Volume-Verfügbarkeit)	99,999 %
Virtuelle Server-Instances in der Amazon Elastic Compute Cloud (Amazon EC2)	Signalisierung	99,950 %
	Datentransport für einzelne AZs	99,950 %
	Datentransport für mehrere AZs	99,990 %
Amazon Elastic Container Service (Amazon ECS)	Signalisierung	99,900 %

Service	Komponente	Verfügbarkeitsdesignziel
	EC2 Container Registry	99,990 %
	EC2 Container Service	99,990 %
Amazon Elastic File System	Signalisierung	99,950 %
	Datentransport	99,990 %
Amazon ElastiCache	Service	99,990 %
Amazon EMR	Signalisierung	99,950 %
Amazon Data Firehose	Service	99,900 %
Amazon Kinesis Data Streams	Service	99,990 %
Amazon Kinesis Video Streams	Service	99,900 %
Amazon Managed Streaming for Apache Kafka (Amazon MSK)	Signalisierung	99,950 %
	Datenebene mit drei AZs	99,990 %
	Datenebene mit zwei AZs	99,950 %
Amazon MQ	Datentransport	99,950 %
	Signalisierung	99,950 %
Amazon Neptune	Service	99,900 %
Amazon OpenSearch Service	Signalisierung	99,950 %
	Datentransport	99,950 %
Amazon Redshift	Signalisierung	99,950 %
	Datentransport	99,950 %

Service	Komponente	Verfügbarkeitsdesignziel
Amazon Rekognition	Service	99,980 %
Amazon Relational Database Service (Amazon RDS)	Signalisierung	99,950 %
	Datentransport für einzelne AZs	99,950 %
	Datentransport für mehrere AZs	99,990 %
Amazon Route 53	Signalisierung	99,950 %
	Datentransport (Wartesch langenauflösung)	100,000 %
Amazon SageMaker	Datenebene (Modell-Hosting)	99,990 %
	Signalisierung	99,950 %
Amazon Simple Notification Service (Amazon SNS)	Datentransport	99,990 %
	Signalisierung	99,900 %
Amazon Simple Queue Service (Amazon SQS)	Datentransport	99,980 %
	Signalisierung	99,900 %
Amazon Simple Storage Service (Amazon S3)	Service (Standard)	99,990 %
AWS Auto Scaling	Signalisierung	99,900 %
	Datentransport	99,990 %
AWS Batch	Signalisierung	99,900 %

Service	Komponente	Verfügbarkeitsdesignziel
	Datentransport	99,950 %
AWS CloudFormation	Service	99,950 %
AWS CloudHSM	Signalisierung	99,900 %
	Datentransport für einzelne AZs	99,900 %
	Datentransport für mehrere AZs	99,990 %
AWS CloudTrail	Signalisierung (Konfiguration)	99,900 %
	Datentransport (Datenereignisse)	99,990 %
	Datentransport (Verwaltungsereignisse)	99,999 %
AWS Config	Service	99,950 %
AWS Data Pipeline	Service	99,990 %
AWS Database Migration Service (AWS DMS)	Signalisierung	99,900 %
	Datentransport	99,950 %
AWS Direct Connect	Signalisierung	99,900 %
	Datentransport für einzelnen Standort	99,900 %
	Datentransport für mehrere Standorte	99,990 %
AWS Global Accelerator	Signalisierung	99,900 %

Service	Komponente	Verfügbarkeitsdesignziel
	Datenebene mit einzelner Netzwerk-Zone	99,950 %
	Datenebene mit zwei Netzwerk-Zonen	99,995 %
AWS Glue	Service	99,990 %
AWS Identity and Access Management	Signalisierung	99,900 %
	Datentransport (Authentifizierung)	99,995 %
AWS IAM Identity Center	Signalisierung	99,900 %
	Datenebene (einschließlich Anmeldung)	99,950 %
AWS IoT Core	Service	99,900 %
AWS IoT Device Management	Service	99,900 %
AWS IoT Greengrass	Service	99,900 %
AWS Key Management Service (AWS KMS)	Signalisierung	99,990 %
	Datentransport	99,995 %
AWS Lambda	Funktionsaufrufe	99,990 %
AWS Secrets Manager	Service	99,990 %
AWS Shield	Signalisierung	99,500 %
	Datentransport (Erkennung)	99,000 %
	Datentransport (Minderung)	99,900 %

Service	Komponente	Verfügbarkeitsdesignziel
AWS Storage Gateway	Signalisierung	99,950 %
	Datentransport	99,950 %
AWS X-Ray	Signalisierung (Konsole)	99,900 %
	Datentransport	99,950 %
Elastic Load Balancing	Signalisierung	99,950 %
	Datentransport	99,990 %