Architektur für die Cloud

Bewährte AWS-Methoden

Februar 2016

Hinweise

Dieses Dokument wird nur zu Informationszwecken zur Verfügung gestellt. Es stellt das aktuelle Produktangebot und die Verfahren von AWS zum Ausstellungsdatum dieses Dokuments dar. Änderungen vorbehalten. Kunden sind für ihre eigene unabhängige Einschätzung der Informationen in diesem Dokument und jedwede Nutzung der AWS-Services verantwortlich. Jeder Service wird „wie besehen“ ohne Gewähr und ohne Garantie jeglicher Art, weder ausdrücklich noch impliziert, bereitgestellt. Dieses Dokument gibt keine Garantien, Gewährleistungen, vertraglichen Verpflichtungen, Bedingungen oder Zusicherungen von AWS, seinen Partnern, Zulieferern oder Lizenzgebern. Die Verantwortung und Haftung von AWS gegenüber seinen Kunden werden durch AWS-Vereinbarungen geregelt. Dieses Dokument ist weder ganz noch teilweise Teil der Vereinbarungen von AWS mit seinen Kunden und ändert diese Vereinbarungen auch nicht.

Inhalt

Kurzbeschreibung

Dieses Whitepaper richtet sich an Lösungsarchitekten und Entwickler, die Lösungen entwickeln, die auf Amazon Web Services (AWS) bereitgestellt werden. Es liefert Architekturkonzepte und Hinweise dazu, wie Sie Systeme entwerfen, die sicher, zuverlässig, leistungsfähig und kosteneffizient sind. Außerdem wird erläutert, wie Sie Attribute nutzen, die spezifisch für die dynamische Natur von Cloud Computing sind (Elastizität, Infrastrukturautomatisierung usw.). Darüber hinaus behandelt dieses Whitepaper auch allgemeine Muster, ihre Weiterentwicklung und ihre Anwendung im Kontext von Cloud Computing.

Einführung

Die Migration von Anwendungen zu AWS, auch ohne wesentliche Änderungen (ein Ansatz, der als „Lift and Shift“ bezeichnet wird), bietet Organisationen die Vorteile einer sicheren und kosteneffizienten Infrastruktur. Um jedoch die Elastizität und Flexibilität, die Cloud Computing ermöglicht, und die Vorteile von AWS optimal nutzen zu können, müssen Entwickler ihre Architekturen weiterentwickeln.

AWS-Kunden haben für neue Anwendungen cloudspezifische IT-Architekturmuster erkannt, die die Effizienz und Skalierbarkeit noch weiter steigern. Diese neuen Architekturen unterstützen verschiedenste Anwendungen – von der Echtzeitanalyse von Webscale-Daten bis zu Anwendungen mit unvorhersehbarem Datenverkehr von Tausenden verbundenen Internet of Things (IoT)-Anschlüssen oder Mobilgeräten.

Dieses Whitepaper erläutert die Prinzipien, die zu beachten sind, wenn Sie vorhandene Anwendungen zu AWS migrieren oder neue Anwendungen für die Cloud entwerfen.

Das Whitepaper setzt grundlegende Kenntnisse über die AWS-Services und -Lösungen voraus. Wenn Sie noch keine Erfahrung mit AWS haben, sehen Sie sich zuerst die Webseite „Informationen zu AWS“[1] an.

Der Cloud-Computing-Unterschied

Dieses Kapitel erklärt, wie Cloud Computing sich von einer herkömmlichen Umgebung unterscheidet und warum sich diese neuen bewährten Methoden herausgebildet haben.

IT-Komponenten werden zu programmierbaren Ressourcen

In einer Nicht-Cloud-Umgebung müssen Sie basierend auf einer Schätzung der theoretischen maximalen Spitzenlast Kapazität bereitstellen. Dies kann dazu führen, dass es Zeiträume gibt, in denen teure Ressourcen ungenutzt bleiben oder die Kapazität nicht ausreicht. Mit Cloud Computing können Sie so viel oder so wenig nutzen, wie Sie benötigen, und dynamisch skalieren, um den tatsächlichen Bedarf zu erfüllen. Sie zahlen nur für das, was Sie tatsächlich nutzen.

Server, Datenbanken, Speicher und übergeordnete Anwendungskomponenten können innerhalb von Sekunden auf AWS instanziiert werden. Sie können diese als temporäre und frei verfügbare Ressourcen behandeln, ohne die Unbeweglichkeit und die Einschränkungen einer festen und endlichen IT-Infrastruktur. Das ermöglicht Ihnen einen ganz neuen Ansatz für Änderungsmanagement, Tests, Zuverlässigkeit und Kapazitätsplanung.

Globale, verfügbare und unbeschränkte Kapazität

Mit der globalen Infrastruktur von AWS können Sie Ihre Anwendung in der AWS-Region[2] bereitstellen, die Ihren Anforderungen am besten entspricht (z. B. in Bezug auf Nähe zu Ihren Endbenutzern, Compliance oder Einschränkungen bezüglich des Lands, in dem Daten gespeichert werden dürfen, Kosten usw.). Bei globalen Anwendungen können Sie die Latenz für Endbenutzer weltweit verringern, indem Sie das Inhaltsbereitstellungsnetzwerk Amazon CloudFront verwenden. Es ist auch viel einfacher, Produktionsanwendungen und Datenbanken in mehreren Rechenzentren zu betreiben, um hohe Verfügbarkeit und Fehlertoleranz zu erreichen. Zusammen mit der praktisch unbegrenzten Kapazität auf Abruf, die AWS-Kunden zur Verfügung steht, erhalten Sie so völlig neue Möglichkeiten für die zukünftige Erweiterung über Ihre IT-Architektur.

Übergeordnete Managed Services

Neben den Rechenressourcen der Amazon Elastic Compute Cloud (Amazon EC2) haben AWS-Kunden auch Zugriff auf eine breite Palette von Speicher-, Datenbank-, Analyse-, Anwendungs- und Bereitstellungsservices. Da diese Services für Entwickler sofort verfügbar sind, verringern sie die Abhängigkeit von internem Fachwissen und ermöglichen Unternehmen, neue Lösungen schneller bereitzustellen. Diese Services werden von AWS verwaltet, was die betriebliche Komplexität und die Kosten senken kann. AWS Managed Services sind speziell auf Skalierbarkeit und hohe Verfügbarkeit ausgelegt. So können sie Risiken für Ihre Implementierungen mindern.

Integrierte Sicherheit

Bei herkömmlicher IT-Infrastruktur sind Sicherheits-Audits oft zyklische und manuelle Prozesse. Die AWS Cloud bietet dagegen Governance-Funktionen, die eine kontinuierliche Überwachung auf Konfigurationsänderungen bei IT-Ressourcen ermöglichen. Da AWS-Komponenten programmierbare Ressourcen sind, kann Ihre Sicherheitsrichtlinie formalisiert und in das Design Ihrer Infrastruktur eingebettet werden. Mit der Möglichkeit, temporäre Umgebungen einzurichten, können Sicherheitstests jetzt Bestandteil Ihrer kontinuierlichen Bereitstellungspipeline werden. Schließlich können Lösungsarchitekten auch eine Vielzahl nativer AWS-Sicherheits- und -Verschlüsselungsfunktionen nutzen, die zur Verbesserung von Datenschutz und Compliance führen können.

Designprinzipien

In diesem Kapitel erläutern wir Entwurfskonzepte und Architekturoptionen, die in einer Vielzahl von Anwendungsfällen eingesetzt werden können.

Skalierbarkeit

Systeme, die im Laufe der Zeit voraussichtlich erweitert werden sollen, müssen sich auf eine skalierbare Architektur stützen. Eine solche Architektur kann Zunahmen bei Benutzern, Datenverkehr oder Datengröße ohne Leistungsverluste meistern. Sie sollte diese Skalierung auf lineare Weise bereitstellen, was bedeutet, dass das Hinzufügen von Ressourcen mindestens zu einem proportionalen Anwachsen der Fähigkeit zur Verarbeitung zusätzlicher

Last führt. Wachstum sollte Skaleneffekte bewirken und die Kosten sollten der Dimension folgen, die geschäftlichen Nutzen aus dem System generiert. Cloud Computing bietet zwar praktisch unbegrenzte Kapazität auf Abruf; Ihr Design muss jedoch in der Lage sein, diese Ressourcen nahtlos zu nutzen. Es gibt generell zwei Möglichkeiten zum Skalieren von IT-Architekturen: vertikal und horizontal.

Vertikale Skalierung

Vertikale Skalierung erfolgt durch eine Aufwertung der Spezifikationen einer einzelnen Ressource (z. B. das Aktualisieren eines Servers mit einer größeren Festplatte oder einer schnelleren CPU). Bei Amazon EC2 lässt sich dies auf einfache Weise erreichen, indem eine Instance angehalten und ihre Größe auf einen Instance-Typ angepasst wird, der mehr RAM, CPU, E/A oder Netzwerkfunktionen bietet. Diese Art der Skalierung kann irgendwann an ihre Grenzen stoßen und ist nicht immer ein kosteneffizienter oder hoch verfügbarer Ansatz. Sie ist jedoch sehr einfach zu implementieren und kann für viele Anwendungsfälle ausreichend sein, besonders kurzfristig.

Horizontale Skalierung

Horizontale Skalierung erfolgt durch eine Erhöhung der Anzahl der Ressourcen (z. B. das Hinzufügen weiterer Festplatten zu einem Speicher-Array oder das Hinzufügen weiterer Server zur Unterstützung einer Anwendung). Sie ist sehr gut für die Erstellung von Webscale-Anwendungen geeignet, die die Elastizität von Cloud Computing nutzen. Nicht alle Architekturen sind so konzipiert, dass sie ihre Workload an mehrere Ressourcen verteilen. Sehen wir uns also einige mögliche Szenarien an.

Zustandslose Anwendungen

Wenn Benutzer oder Services mit einer Anwendung interagieren, führen sie häufig eine Reihe von Interaktionen durch, die eine Sitzung bilden. Eine zustandslose Anwendung ist eine Anwendung, die kein Wissen über vorherige Interaktionen benötigt und keine Sitzungsinformationen speichert. Ein Beispiel dafür wäre eine Anwendung, die bei derselben Eingabe allen Endbenutzern dieselbe Antwort liefert. Eine zustandslose Anwendung kann horizontal skaliert werden, da jede Anfrage von jeder beliebigen verfügbaren Rechenressource (z. B. EC2-Instances, AWS Lambda-Funktionen) erfüllt werden kann. Da keine Sitzungsdaten geteilt werden müssen, können Sie einfach weitere Rechenressourcen nach Bedarf hinzufügen. Wenn diese Kapazität nicht mehr

benötigt wird, können alle einzelnen Ressourcen problemlos beendet werden (nachdem die laufenden Aufgaben abgearbeitet wurden). Diese Ressourcen müssen nichts vom Vorhandensein der anderen Ressourcen wissen – alles, was benötigt wird, ist eine Möglichkeit zum Verteilen der Workload an sie.

So verteilen Sie Lasten auf mehrere Knoten

Push-Modell: Eine beliebte Möglichkeit zum Verteilen einer Workload ist die Verwendung einer Load-Balancing-Lösung wie z. B. des Elastic Load Balancing (ELB) Service. Elastic Load Balancing leitet eingehende Anwendungsanfragen über mehrere EC2-Instances. Alternativ könnte ein DNS-Roundrobin implementiert werden (z. B. mit Amazon Route 53). In diesem Fall liefern DNS-Antworten in Roundrobin-Manier eine IP-Adresse aus einer Liste gültiger Hosts. Dieser Ansatz bietet zwar eine einfache Implementierung, funktioniert aber nicht immer gut mit der Elastizität von Cloud Computing. Der Grund ist, dass sich – auch wenn Sie niedrige TTL-Werte (Time To Live, Gültigkeitsdauer) für Ihre DNS-Datensätze festlegen können – die Caching-DNS-Resolver außerhalb der Kontrolle von Amazon Route 53 befinden und möglicherweise nicht immer Ihren Einstellungen folgen.

Pull-Modell: Asynchrone ereignisgesteuerte Workloads erfordern keine Load-Balancing-Lösung, da Sie stattdessen ein Pull-Modell implementieren können. Bei einem Pull-Modell können auszuführende Aufgaben oder zu verarbeitende Daten als Nachrichten in einer Warteschlange gespeichert werden (mithilfe von Amazon Simple Queue Service (Amazon SQS)) oder es kann eine Streaming-Daten-Lösung wie Amazon Kinesis verwendet werden. Mehrere Rechenknoten können dann diese Nachrichten herausziehen und sie auf verteilte Weise verarbeiten.

Zustandslose Komponenten

In der Praxis müssen die meisten Anwendungen irgendeine Art von Statusinformationen pflegen. Beispielsweise müssen Webanwendungen nachverfolgen, ob ein Benutzer angemeldet ist, damit sie personalisierte Inhalte basierend auf vorherigen Aktionen anbieten können. Ein automatisierter, mehrstufiger Prozess muss auch vorherige Aktivitäten nachverfolgen, um zu entscheiden, was die nächste Aktion sein soll. Sie können weiterhin einen Teil dieser Architekturen zustandslos gestalten, indem Sie nichts im lokalen Dateisystem speichern, das für mehr als eine einzige Anfrage bestehen bleiben muss.

Beispielsweise können Webanwendungen HTTP-Cookies zum Speichern von Informationen über eine Sitzung auf dem Clientbrowser (z. B. Artikel im Warenkorb) verwenden. Der Browser leitet diese Informationen bei jeder nachfolgenden Anfrage wieder an den Server weiter, sodass die Anwendung sie nicht speichern muss. Dieser Ansatz hat jedoch zwei Nachteile: Erstens kann der Inhalt der HTTP-Cookies auf der Clientseite manipuliert werden. Deshalb sollten Sie sie immer als nicht vertrauenswürdige Daten behandeln, die validiert werden müssen. Zweitens werden HTTP-Cookies mit jeder Anfrage übermittelt. Das bedeutet, dass Sie ihre Größe auf ein Minimum beschränken sollten (zum Vermeiden unnötiger Latenz).

Sie könnten erwägen, nur eine eindeutige Sitzungs-ID in einem HTTP-Cookie zu speichern und ausführlichere Informationen zur Benutzersitzung serverseitig zu speichern. Die meisten Programmierplattformen bieten einen nativen Mechanismus zur Sitzungsverwaltung, der so funktioniert; dabei wird jedoch oft standardmäßig im lokalen Dateisystem gespeichert. Dies würde zu einer zustandsbehafteten Architektur führen. Eine gängige Lösung für dieses Problem besteht darin, Benutzersitzungsdaten in einer Datenbank zu speichern. Amazon DynamoDB ist aufgrund ihrer Skalierbarkeit, hohen Verfügbarkeit und Beständigkeit eine erstklassige Wahl. Für viele Plattformen gibt es Open-Source-Drop-In-Ersatzbibliotheken, mit denen Sie native Sitzungen in Amazon DynamoDB[3] speichern können.

Andere Szenarien erfordern die Speicherung größerer Dateien (z. B. Benutzer-Uploads, Zwischenergebnisse von Batch-Prozessen usw.). Indem Sie diese Dateien in einer gemeinsam genutzten Speicherebene wie Amazon S3 oder Amazon Elastic File System (Amazon EFS) speichern, können Sie die Einführung zustandsbehafteter Komponenten vermeiden. Ein anderes Beispiel ist ein komplexer, mehrstufiger Workflow, bei dem Sie den aktuellen Status jeder Ausführung verfolgen müssen. Mit Amazon Simple Workflow Service (Amazon SWF) können Sie den Ausführungsverlauf zentral speichern und diese Workloads zustandslos machen.

Zustandsbehaftete Komponenten

Unweigerlich wird es Ebenen in Ihrer Architektur geben, die Sie nicht in zustandslose Komponenten verwandeln möchten. Datenbanken sind per definitionem zustandsbehaftet. (Sie werden getrennt im Kapitel Datenbanken besprochen.) Darüber hinaus sind viele ältere Anwendungen dafür konzipiert, auf einem einzelnen Server ausgeführt zu werden, indem sie lokale

Rechenressourcen verwenden. Andere Anwendungsfälle erfordern möglicherweise Clientgeräte, um für längere Zeiträume eine Verbindung zu einem bestimmten Server aufrechtzuerhalten. Beispielsweise muss Echtzeit-Multiplayer-Gaming mehreren Spielern eine konsistente Ansicht der Spielwelt mit sehr niedriger Latenz bieten. Das ist in einer nicht-verteilten Implementierung, bei der die Teilnehmer mit demselben Server verbunden sind, sehr viel einfacher zu erreichen.

Möglicherweise können Sie diese Komponenten weiterhin horizontal skalieren, indem Sie die Last auf mehrere Knoten mit „Sitzungsaffinität“ verteilen. Bei diesem Modell binden Sie alle Transaktionen einer Sitzung an eine bestimmte Rechenressource. Sie sollten sich über die Einschränkungen dieses Modells im Klaren sein. Vorhandene Sitzungen profitieren nicht direkt von der Einführung neu gestarteter Rechenknoten. Noch wichtiger: Sitzungsaffinität kann nicht garantiert werden. Wenn beispielsweise ein Knoten beendet wird oder nicht mehr verfügbar ist, wird die Verbindung daran gebundener Benutzer getrennt und ihnen gehen sitzungsspezifische Daten verloren (z. B. alle Elemente, die nicht in einer gemeinsam genutzten Ressource wie S3, EFS, einer Datenbank usw. gespeichert sind).

So wird Sitzungsaffinität implementiert

Für HTTP/S-Datenverkehr kann Sitzungsaffinität über die „Sticky Sessions“-Funktion von ELB[4] erreicht werden. Elastic Load Balancing versucht, für diesen Benutzer für die Dauer der Sitzung denselben Server zu verwenden.

Wenn Sie den Code, der auf dem Client ausgeführt wird, steuern, ist eine andere Option die Verwendung von clientseitigem Load Balancing. Das sorgt für zusätzliche Komplexität, kann aber in Szenarien nützlich sein, in denen ein Load Balancer Ihre Anforderungen nicht erfüllt. Zum Beispiel verwenden Sie möglicherweise ein Protokoll, das von ELB nicht unterstützt wird, oder benötigen volle Kontrolle darüber, wie Benutzer Servern zugewiesen werden (in einem Gaming-Szenario müssen Sie z. B. möglicherweise sicherstellen, dass die Spieler abgeglichen und mit demselben Server verbunden werden). In diesem Modell benötigen die Clients eine Möglichkeit, gültige Serverendpunkte zu erkennen, mit denen sie sich direkt verbinden können. Sie können hierfür DNS verwenden oder eine einfache Erkennungs-API erstellen, um der Software, die auf dem Client ausgeführt wird, die Informationen bereitzustellen. In Ermangelung eines Load Balancers muss der Zustandsprüfungsmechanismus auch auf der Clientseite implementiert werden. Sie sollten die Clientlogik so gestalten, dass Geräte sich bei Nichtverfügbarkeit des Servers mit einem anderen Server verbinden, bei minimalen Ausfallzeiten für die Anwendung.

Verteilte Verarbeitung

Anwendungsfälle, die die Verarbeitung sehr großer Datenmengen umfassen (z. B. alles, was nicht von einer einzelnen Rechenressource in angemessener Zeit verarbeitet werden kann), erfordern eine verteilte Verarbeitung. Durch das Aufteilen einer Aufgabe und ihrer Daten in viele kleine Arbeitsfragmente können Sie jede von ihnen in einer beliebigen verfügbaren Rechenressource ausführen.

> ***So implementieren Sie die verteilte Verarbeitung***
>
> *Offline-Batch-Aufträge können horizontal skaliert werden, indem eine verteilte Datenverarbeitungs-Engine wie Apache Hadoop verwendet wird. In AWS können Sie mithilfe des Services Amazon Elastic MapReduce (Amazon EMR) Hadoop-Workloads auf einer Flotte von EC2-Instances ohne die übliche Betriebskomplexität ausführen. Für die Echtzeitverarbeitung von Streaming-Daten partitioniert Amazon Kinesis Daten in mehrere Shards, die dann von mehreren Amazon EC2- oder AWS Lambda-Ressourcen verwendet werden können, um Skalierbarkeit zu erreichen.*

Weitere Informationen zu diesen Workload-Typen finden Sie im Whitepaper „Big Data-Analyseoptionen auf AWS“[5].

Frei verfügbare Ressourcen statt fester Server

In einer herkömmlichen Infrastrukturumgebung muss man aufgrund der Investitionskosten und der Vorlaufzeit für die Einführung neuer Hardware mit fixen Ressourcen arbeiten. Das bringt Praktiken wie manuelles Anmelden an Servern, um Software zu konfigurieren oder Probleme zu beheben, Hartcodieren von IP-Adressen, sequenzielles Testen oder Verarbeiten von Aufträgen usw. mit sich.

Beim Entwickeln für AWS haben Sie die Möglichkeit, einen ganz anderen Denkansatz zu verfolgen, um die Vorteile der dynamischen Bereitstellung beim Cloud Computing auszuschöpfen. Sie können Server und andere Komponenten als temporäre Ressourcen ansehen. Sie können so viele davon starten, wie Sie benötigen, und nur so lange verwenden, wie nötig.

Ein weiteres Problem bei fixen, lange laufenden Servern ist die sogenannte *Konfigurationsabweichung*. Änderungen und Software-Patches, die im Laufe der Zeit angewendet werden, können zu ungetesteten und heterogenen Konfigurationen in verschiedenen Umgebungen führen. Dieses Problem lässt sich durch das Konzept *unveränderliche Infrastruktur* lösen. Bei diesem

Ansatz wird ein Server, wenn er einmal gestartet wurde, während seiner gesamten Lebensdauer niemals aktualisiert. Stattdessen wird er, wenn ein Problem auftritt oder eine Aktualisierung erforderlich wird, durch einen neuen Server ersetzt, der über die neueste Konfiguration verfügt. Auf diese Weise haben Ressourcen immer einen konsistenten (und getesteten) Status und Rollbacks können einfacher ausgeführt werden.

Instanziieren von Rechenressourcen

Ob Sie eine neue Umgebung zum Testen bereitstellen oder die Kapazität eines vorhandenen Systems erhöhen möchten, um zusätzliche Last zu bewältigen – Sie werden neue Ressourcen nicht manuell mit ihrer Konfiguration und ihrem Code einrichten wollen. Es ist wichtig, dies zu einem automatisierten und wiederholbaren Prozess zu machen, der keine langen Vorlaufzeiten hat und nicht fehlerträchtig ist.

Es gibt einige Ansätze, wie sich ein automatisierter und wiederholbarer Prozess erreichen lässt.

Bootstrapping

Beim Starten einer AWS-Ressource wie einer Amazon EC2-Instance oder einer DB-Instance von Amazon Relational Database (Amazon RDS) beginnen Sie mit einer Standardkonfiguration. Anschließend können Sie automatisierte Bootstrapping-Aktionen ausführen – d. h. Skripte, die Software installieren oder Daten kopieren, um diese Ressource in einen bestimmten Zustand zu bringen. Sie können Konfigurationsdetails, die zwischen verschiedenen Umgebungen variieren (z. B. Produktion, Test usw.), parametrisieren, damit dieselben Skripte ohne Änderungen wiederverwendet werden können.

> ***Bootstrapping in der Praxis***
>
> *Sie können Benutzerdatenskripte und cloud-init-Anweisungen[6] oder AWS OpsWorks-Lebenszyklusereignisse[7] verwenden, um neue EC2-Instances automatisch einzurichten. Sie können einfache Skripte und Konfigurationsverwaltungstools wie Chef oder Puppet verwenden. AWS OpsWorks bietet native Unterstützung für Chef-Rezepte oder Bash-/PowerShell-Skripte. Darüber hinaus kann durch benutzerdefinierte Skripte und die AWS-APIs oder durch die Verwendung der AWS CloudFormation-Unterstützung für AWS Lambda-gestützte benutzerdefinierte Ressourcen[8] eine Bereitstellungslogik geschrieben werden, die für fast alle AWS-Ressourcen wirksam ist.*

Golden Images

Bestimmte AWS-Ressourcentypen wie Amazon EC2-Instances, DB-Instances von Amazon RDS, Volumes von Amazon Elastic Block Store (Amazon EBS) usw. können über ein Golden Image gestartet werden – einen Snapshot eines bestimmten Status dieser Ressource. Im Vergleich zum Bootstrapping-Ansatz resultiert ein Golden Image in kürzeren Startzeiten und eliminiert Abhängigkeiten von Konfigurationsservices oder Drittanbieter-Repositorys. Das ist in automatisch skalierten Umgebungen wichtig, in denen Sie schnell und zuverlässig zusätzliche Ressourcen als Reaktion auf Nachfrageänderungen starten können möchten.

Sie können eine Amazon EC2-Instance anpassen und dann ihre Konfiguration speichern, indem Sie ein Amazon Machine Image (AMI)[9] erstellen. Sie können über das AMI so viele Instances bereitstellen, wie Sie benötigen, und sie alle enthalten die von Ihnen vorgenommenen Anpassungen. Jedes Mal, wenn Sie Ihre Konfiguration ändern möchten, müssen Sie ein neues Golden Image erstellen. Deshalb müssen Sie eine Versioning-Konvention für die Verwaltung Ihrer Golden Images implementieren. Wir empfehlen, zum Bootstrappen der EC2-Instances, die Sie zum Erstellen Ihrer AMIs verwenden, ein Skript zu nutzen. Auf diese Weise erhalten Sie eine flexible Möglichkeit zum Testen und Modifizieren der Images im Laufe der Zeit.

Wenn Sie über eine virtualisierte Umgebung vor Ort verfügen, können Sie alternativ VM Import/Export von AWS verwenden, um verschiedene Virtualisierungsformate zu einem AMI zu konvertieren. Sie können auch vorgefertigte gemeinsame AMIs suchen und nutzen, die von AWS oder von Drittanbietern im AMI-Katalog der AWS-Community oder im AWS Marketplace bereitgestellt werden.

Golden Images werden zwar am häufigsten beim Starten neuer EC2-Instances verwendet, sie können jedoch auch auf Ressourcen wie Amazon RDS-Datenbanken oder Amazon EBS-Volumes angewendet werden. Wenn Sie beispielsweise eine neue Testumgebung starten, könnten Sie ihre Datenbank vorausfüllen, indem Sie sie aus einem bestimmten Amazon RDS-Snapshot instanziieren, anstatt die Daten aus einem langen SQL-Skript zu importieren.

Container

Eine weitere beliebte Möglichkeit bei Entwicklern ist Docker – eine Open-Source-Technologie, mit der Sie verteilte Anwendungen innerhalb von Softwarecontainern erstellen und bereitstellen können. Docker ermöglicht das Verpacken einer Software in einem Docker-Image – einer standardisierten Einheit zur Softwareentwicklung, die alles enthält, was die Software zur Ausführung benötigt: Code, Laufzeit, Systemtools, Systembibliotheken usw. AWS Elastic Beanstalk und Amazon EC2 Container Service (Amazon ECS) unterstützen Docker und ermöglichen Ihnen die Bereitstellung und Verwaltung mehrerer Docker-Container in einem Cluster von Amazon EC2-Instances.

Hybrid

Sie können eine Kombination beider Ansätze verwenden, bei der einige Teile der Konfiguration in einem Golden Image erfasst werden, während andere dynamisch über eine Bootstrapping-Aktion konfiguriert werden.

Abgrenzung zwischen Bootstrapping und Golden Image

Elemente, die sich nicht häufig ändern oder die externe Abhängigkeiten einführen, sind üblicherweise Teil des Golden Image. Ihre Webserversoftware, die ansonsten bei jedem Start einer Instance von einem Drittanbieter-Repository heruntergeladen werden müsste, ist z. B. ein guter Kandidat.

Elemente, die sich häufig ändern oder sich in Ihren verschiedenen Umgebungen unterscheiden, können dynamisch über Bootstrapping-Aktionen eingerichtet werden. Wenn Sie beispielsweise häufig neue Versionen Ihrer Anwendung bereitstellen, kann das Erstellen eines neuen AMI für jede Anwendungsversion in der Praxis nicht durchführbar sein. Es wäre auch nicht empfehlenswert, die Datenbank-Hostnamen-Konfiguration für das AMI hartzucodieren, da sie sich zwischen Test- und Produktionsumgebung unterscheidet. Sie können Benutzerdaten oder Tags verwenden, um allgemeinere AMIs verwenden zu können, die beim Starten geändert werden können. Wenn Sie z. B. Webserver für verschiedene kleine Unternehmen ausführen, können diese alle dasselbe AMI verwenden und ihre Inhalte aus einem Amazon S3-Bucket-Speicherort abrufen, den Sie in den Benutzerdaten beim Starten angeben.

AWS Elastic Beanstalk folgt dem Hybridmodell. Es stellt vorkonfigurierte Laufzeitumgebungen bereit (die jeweils von ihrem eigenen AMI initiiert werden[10]), erlaubt jedoch, Bootstrap-Aktionen auszuführen (über Konfigurationsdateien namens .ebextensions[11]) und Umgebungsvariablen zu konfigurieren, um die Umgebungsunterschiede zu parametrisieren.

Eine detailliertere Erläuterung der verschiedenen Möglichkeiten zum Verwalten von Bereitstellungen neuer Ressourcen finden Sie in den Whitepapers Übersicht über die Bereitstellungsoptionen in AWS und Verwaltung Ihrer AWS-Infrastruktur nach Größe.

Infrastruktur als Code

Die Anwendung der von uns diskutierten Prinzipien muss nicht auf die Ebene einzelner Ressourcen begrenzt sein. Da AWS-Komponenten programmierbar sind, können Sie Techniken, Praktiken und Tools aus der Softwareentwicklung anwenden, um Ihre gesamte Infrastruktur wiederverwendbar, wartbar, erweiterbar und testbar zu machen.

> *Vorlagen von **AWS CloudFormation** bieten Entwicklern und Systemadministratoren eine einfache Möglichkeit, eine Sammlung verwandter AWS-Ressourcen zu erstellen und zu verwalten und diese in geordneter und vorhersehbarer Form bereitzustellen und zu aktualisieren. Sie können die AWS-Ressourcen und etwaige zugehörige Abhängigkeiten oder Laufzeitparameter beschreiben, die zur Ausführung Ihrer Anwendung erforderlich sind. Ihre CloudFormation-Vorlagen können mit Ihrer Anwendung in Ihrem Versionskontroll-Repository gespeichert werden, sodass Architekturen wiederverwendet und Produktionsumgebungen zum Testen zuverlässig geklont werden können.*

Automatisierung

In einer herkömmlichen IT-Infrastruktur muss man häufig manuell auf verschiedene Ereignisse reagieren. Beim Bereitstellen auf AWS gibt es viele Möglichkeiten zur Automatisierung, sodass Sie sowohl die Stabilität Ihres Systems als auch die Effizienz der Organisation steigern können:

- **AWS Elastic Beanstalk**[12] ist die schnellste und einfachste Möglichkeit, eine Anwendung auf AWS auszuführen. Entwickler können einfach ihren Anwendungscode hochladen und der Service kümmert sich automatisch um alle Details wie Ressourcenbereitstellung, Load Balancing, Auto Scaling und Überwachung.

- **Automatische Wiederherstellung in Amazon EC2**[13]: Sie können einen Amazon CloudWatch-Alarm erstellen, der eine Amazon EC2-Instance überwacht und sie automatisch wiederherstellt, wenn sie beeinträchtigt wird. Eine wiederhergestellte Instance ist mit der ursprünglichen Instance identisch. Das schließt auch die Instance-ID, private IP-Adressen, elastische IP-Adressen und alle Instance-Metadaten ein. Allerdings ist diese Funktion nur für bestimmte Instance-Konfigurationen verfügbar. Eine aktuelle

Beschreibung dieser Voraussetzungen finden Sie in der Amazon EC2-Dokumentation. Darüber hinaus wird die Instance während der Instance-Wiederherstellung durch einen Instance-Neustart migriert und alle In-Memory-Daten gehen verloren.

- **Auto Scaling**[14]: Wenn Auto Scaling angewendet wird, können Sie die Anwendungsverfügbarkeit aufrechterhalten und Ihre Amazon EC2-Kapazität gemäß Bedingungen, die Sie definieren, automatisch nach oben oder unten anpassen. Sie können Auto Scaling nutzen, um sicherzustellen, dass Sie die gewünschte Anzahl intakter Amazon EC2-Instances in mehreren Availability Zones ausführen. Auto Scaling kann zudem die Anzahl der Amazon EC2-Instances bei Nachfragespitzen automatisch erhöhen, um die Leistung aufrechtzuerhalten, und bei niedrigerer Nachfrage die Kapazität senken, um die Kosten zu optimieren.

- **Amazon CloudWatch-Alarme**[15]: Sie können einen CloudWatch-Alarm erstellen, der eine Nachricht über Amazon Simple Notification Service (Amazon SNS) sendet, wenn eine bestimmte Metrik für eine bestimmte Anzahl von Zeiträumen einen angegebenen Schwellenwert überschreitet. Diese Amazon SNS-Nachrichten können automatisch die Ausführung einer abonnierten AWS Lambda-Funktion auslösen, eine Benachrichtigung in eine Amazon SQS-Warteschlange einreihen oder eine POST-Anfrage an einen HTTP/S-Endpunkt stellen.

- **Amazon CloudWatch Events**[16]: Der CloudWatch-Service stellt einen Stream von Systemereignissen nahezu in Echtzeit bereit, die Änderungen an AWS-Ressourcen beschreiben. Mit einfachen Regeln, die Sie in wenigen Minuten aufsetzen können, können Sie ganz einfach jeden Ereignistyp an ein oder mehrere Ziele leiten: AWS Lambda-Funktionen, Amazon Kinesis Streams, Amazon SNS-Themen usw.

- **AWS OpsWorks-Lebenszyklusereignisse**[17]: AWS OpsWorks unterstützt eine kontinuierliche Konfiguration über Lebenszyklusereignisse, die die Konfiguration Ihrer Instances zur Anpassung an Umgebungsveränderungen automatisch aktualisieren. Diese Ereignisse können zum Auslösen von Chef-Rezepten für jede Instance verwendet werden, um bestimmte Konfigurationsaufgaben auszuführen. Wenn beispielsweise eine neue Instance erfolgreich zu einer Datenbankserverebene hinzugefügt wird, könnte das Konfigurationsereignis ein Chef-Rezept auslösen, das die Konfiguration der Anwendungsserverebene so aktualisiert, dass sie auf die neue Datenbank-Instance verweist.

- **AWS Lambda und geplante Ereignisse**[18]: Diese Ereignisse ermöglichen Ihnen, eine Lambda-Funktion zu erstellen und AWS Lambda anzuweisen, sie regelmäßig auszuführen.

Lose Verkoppelung

Mit zunehmender Anwendungskomplexität ist es ein wünschenswertes Attribut eines IT-Systems, in kleinere, lose gekoppelte Komponenten aufgebrochen werden zu können. Das bedeutet, dass IT-Systeme so ausgelegt sein sollten, dass gegenseitige Abhängigkeiten reduziert werden – eine Änderung oder ein Fehler in einer Komponente sollte sich nicht auf andere Komponenten auswirken.

Gut definierte Schnittstellen

Eine Möglichkeit zur Reduzierung der Abhängigkeiten in einem System ist, den verschiedenen Komponenten das Interagieren nur über bestimmte, technologieunabhängige Schnittstellen zu erlauben (z. B. RESTful-APIs). Technische Implementierungsdetails sind so nicht sichtbar und Teams können die zugrunde liegende Implementierung anpassen, ohne dass dies Auswirkungen auf andere Komponenten hat. Solange diese Schnittstellen Rückwärtskompatibilität wahren, sind Bereitstellungen unterschiedlicher Komponenten entkoppelt.

> ***Amazon API Gateway*** *ist ein vollständig verwalteter Service, der das Erstellen, Veröffentlichen, Warten, Überwachen und Sichern von APIs für Entwickler in jeder beliebigen Größenordnung vereinfacht. Er führt alle Aufgaben für das Akzeptieren und Verarbeiten von bis zu Hunderttausenden gleichzeitigen API-Aufrufen aus, einschließlich Verwaltung des Datenverkehrs, Autorisierung und Zugriffskontrolle, Überwachung und Verwaltung der API-Version.*

Service-Erkennung

Bei Anwendungen, die als eine Reihe kleinerer Services bereitgestellt werden, kommt es darauf an, ob diese Services in der Lage sind, miteinander zu interagieren. Da jeder dieser Services über mehrere Rechenressourcen ausgeführt werden könnte, muss es eine Möglichkeit geben, jeden Service anzusprechen. Ein Beispiel: Wenn sich in einer herkömmlichen Infrastruktur Ihr Front-End-Webservice mit dem Back-End-Webservice verbinden müsste, könnten Sie die IP-Adresse der Rechenressource, in der dieser Service ausgeführt wurde, hartcodieren. Dieser Ansatz kann zwar auch beim Cloud Computing noch funktionieren, aber wenn diese Services lose gekoppelt sein sollen, sollten sie verarbeitet werden können, ohne dass vorherige Kenntnisse ihrer Netzwerktopologiedetails erforderlich sind. Abgesehen vom Verbergen der Komplexität können so auch Infrastrukturdetails jederzeit geändert werden. Lose Kopplung ist ein wesentliches Element, wenn Sie die Vorteile der Elastizität von

Cloud Computing nutzen möchten, bei der neue Ressourcen jederzeit gestartet oder beendet werden können. Um dies zu erreichen, benötigen Sie eine Möglichkeit zur Implementierung von Service-Erkennung.

So implementieren Sie Service-Erkennung

Eine einfache Möglichkeit, für einen in Amazon EC2 gehosteten Service Service-Erkennung zu erreichen, ist über den Service Elastic Load Balancing. Da jeder Load Balancer seinen eigenen Hostnamen erhält, haben Sie jetzt die Möglichkeit, einen Service über einen stabilen Endpunkt zu nutzen. Das kann mit DNS und privaten Amazon Route 53-Zonen kombiniert werden, sodass sogar der Endpunkt des bestimmten Load Balancers jederzeit abstrahiert und geändert werden kann.

Eine weitere Möglichkeit wäre die Verwendung einer Serviceregistrierung und Erkennungsmethode, um das Abrufen der Endpunkt-IP-Adressen und der Portnummer jedes beliebigen Services zu ermöglichen. Da die Service-Erkennung zu dem Element wird, das die Komponenten zusammenhält, ist es wichtig, dass es hoch verfügbar und zuverlässig ist. Wenn keine Load Balancer verwendet werden, sollte die Service-Erkennung auch Dinge wie die Zustandsprüfung abdecken. Beispielimplementierungen sind angepasste Lösungen mit einer Kombination aus Tags, einer hoch verfügbaren Datenbank und angepassten Skripten, die die AWS-APIs aufrufen, sowie Open-Source-Tools wie Netflix Eureka, Airbnb Synapse oder HashiCorp Consul.

Asynchrone Integration

Asynchrone Integration ist eine weitere Form loser Verkoppelung zwischen Services. Dieses Modell eignet sich für jede Interaktion, die keine sofortige Antwort erfordert und bei der eine Bestätigung, dass eine Anfrage registriert wurde, ausreicht. Es umfasst eine Komponente, die Ereignisse generiert, und eine andere, die diese verarbeitet. Die beiden Komponenten integrieren sich nicht über direkte Punkt-zu-Punkt-Interaktion, sondern üblicherweise über eine zwischengeschaltete, beständige Speicherebene (z. B. eine Amazon SQS-Warteschlange oder eine Streaming-Daten-Plattform wie Amazon Kinesis).

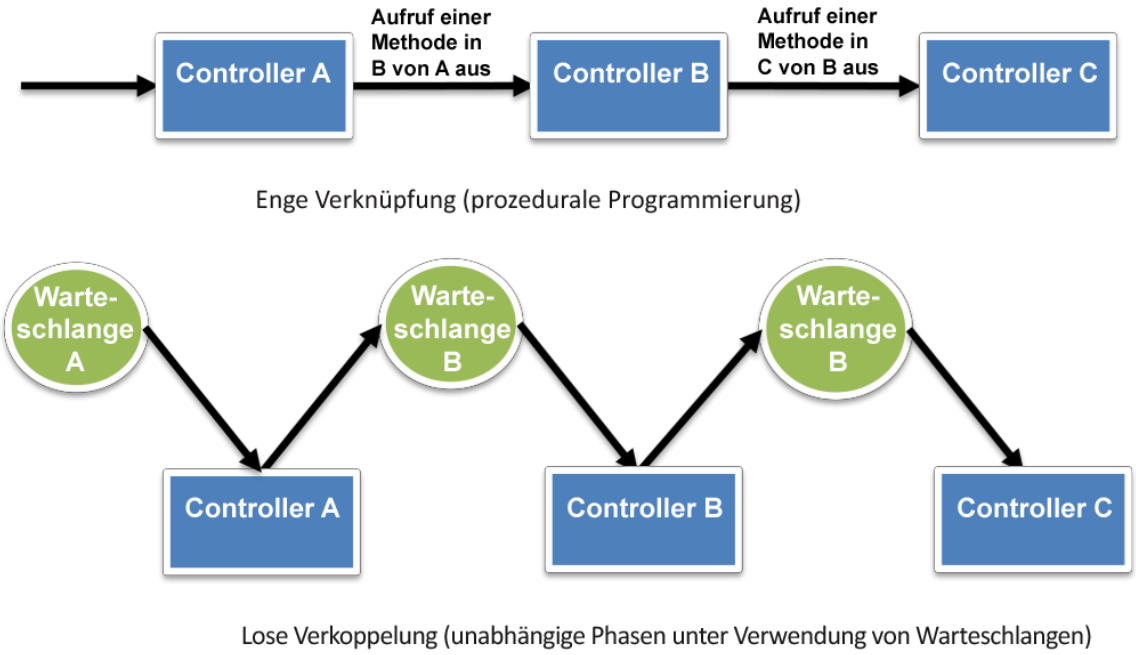

Abbildung 1: Enge Verknüpfung und lose Verkoppelung

Dieser Ansatz entkoppelt die beiden Komponenten und schafft zusätzliche Resilienz. Wenn beispielsweise ein Prozess, der Nachrichten aus der Warteschlange liest, fehlschlägt, können trotzdem weiterhin Nachrichten zur Warteschlange hinzugefügt und dann verarbeitet werden, wenn das System wiederhergestellt ist. Außerdem können Sie so einen weniger skalierbaren Back-End-Service vor Front-End-Bedarfsspitzen schützen und den richtigen Kompromiss zwischen Kosten und Verarbeitungsverzögerung finden. Beispielsweise können Sie entscheiden, dass Sie Ihre Datenbank nicht skalieren müssen, um gelegentliche Spitzen von Schreibabfragen bewältigen zu können, solange diese Abfragen irgendwann asynchron mit einiger Verzögerung verarbeitet werden. Schließlich können Sie durch das Verlagern langsamer Vorgänge aus interaktiven Abfragepfaden auch die Endbenutzererfahrung verbessern.

Beispiele für asynchrone Integration

- *Eine Front-End-Anwendung fügt Aufträge zu einem Warteschlangensystem wie Amazon SQS hinzu. Ein Back-End-System ruft diese Aufträge ab und verarbeitet sie in ihrem eigenen Tempo.*

- *Eine API generiert Ereignisse und überträgt sie in Amazon Kinesis Streams. Eine Back-End-Anwendung verarbeitet diese Ereignisse in Batches, um aggregierte Zeitreihendaten zu erstellen, die in einer Datenbank gespeichert werden.*

- *Mehrere heterogene Systeme nutzen Amazon SWF, um den Arbeitsfluss zwischen ihnen zu kommunizieren, ohne direkt miteinander zu interagieren.*

- *AWS Lambda-Funktionen können Ereignisse aus einer Vielzahl von AWS-Quellen verarbeiten (z. B. Amazon DynamoDB-Updatestreams, Amazon S3-Ereignisbenachrichtigungen usw.). In diesem Fall brauchen Sie sich nicht einmal Gedanken über die Implementierung einer Warteschlangen- oder anderen asynchronen Integrationsmethode zu machen, da der Service dies für Sie übernimmt.*

Graceful Failure

Eine weitere Möglichkeit, lose Verkoppelung zu stärken, ist, Anwendungen so zu erstellen, dass sie Komponentenausfälle ohne negative Folgen meistern (Graceful Failure). Sie können Wege finden, wie sich die Auswirkungen auf die Endbenutzer verringern lassen, und Ihre Fähigkeit, Fortschritte bei den Offline-Verfahren zu machen, erweitern – auch beim Ausfall von Komponenten.

Graceful Failure in der Praxis

Eine fehlgeschlagene Anfrage kann mit einem exponentiellen Backoff und einer Jitter-Strategie[19] erneut versucht werden oder zur späteren Verarbeitung in einer Warteschlange gespeichert werden. Bei Front-End-Schnittstellen können Sie möglicherweise alternative oder zwischengespeicherte Inhalte anbieten, anstatt einen kompletten Ausfall zu verzeichnen, z. B., wenn Ihr Datenbankserver nicht verfügbar ist. Die DNS Failover-Funktion von Amazon Route 53 bietet Ihnen auch die Möglichkeit, Ihre Webseite zu überwachen und Besucher automatisch auf eine Backup-Webseite zu leiten, wenn Ihre primäre Webseite nicht verfügbar ist. Sie können Ihre Backup-Webseite als statische Webseite in Amazon S3 oder als separate dynamische Umgebung hosten.

Services, nicht Server

Das Entwickeln, Verwalten und Betreiben von Anwendungen – vor allem in großem Maßstab – erfordert eine Vielzahl zugrunde liegender Technologiekomponenten. Bei herkömmlichen IT-Infrastrukturen müssten Unternehmen alle diese Komponenten erstellen und betreiben.

AWS bietet eine breite Palette an Datenverarbeitungs-, Speicher-, Datenbank-, Analyse-, Anwendungs- und Bereitstellungsservices, die Unternehmen agiler machen und die IT-Kosten senken. Architekturen, die diese Bandbreite nicht nutzen (z. B., wenn sie nur Amazon EC2 verwenden), schöpfen möglicherweise die Leistungsfähigkeit von Cloud Computing nicht voll aus und versäumen die Gelegenheit, die Produktivität von Entwicklern und die betriebliche Effizienz zu steigern.

Managed Services

In AWS gibt es eine Reihe von Services, die Bausteine bereitstellen, die Entwickler für ihre Anwendungen nutzen können. Diese Managed Services umfassen Datenbanken, maschinelles Lernen, Analyse, Warteschlangen, Suche, E-Mail-Benachrichtigungen und vieles mehr. Beispielsweise können Sie mit dem Amazon Simple Queue Service (Amazon SQS) den administrativen Aufwand von Betrieb und Skalierung eines hoch verfügbaren Messaging-Clusters auslagern. Dabei zahlen Sie nur einen niedrigen Preis für die tatsächlich in Anspruch genommenen Ressourcen. Zudem ist Amazon SQS inhärent skalierbar. Das Gleiche gilt für Amazon S3. Dort können Sie so viele Daten speichern, wie Sie möchten, und bei Bedarf darauf zugreifen, ohne sich Gedanken über Kapazität, Festplattenkonfigurationen, Replikation usw. machen zu müssen. Darüber hinaus kann Amazon S3 auch statische Komponenten einer Webanwendung oder Mobil-App verarbeiten und liefert so eine hoch verfügbare Hosting-Lösung, die sich automatisch gemäß den Datenverkehrsanforderungen skalieren lässt.

Es gibt viele weitere Beispiele, darunter Amazon CloudFront zur Inhaltsbereitstellung, ELB für Load Balancing, Amazon DynamoDB für NoSQL-Datenbanken, Amazon CloudSearch für Such-Workloads, Amazon Elastic Transcoder zur Videocodierung und Amazon Simple Email Service (Amazon SES) zum Senden und Empfangen von E-Mails und mehr.[20].

Serverlose Architekturen

Ein anderer Ansatz, der die betriebliche Komplexität beim Ausführen von Anwendungen verringern kann, sind serverlose Architekturen. Es ist möglich, sowohl ereignisgesteuerte als auch synchrone Services für Mobil-, Web- und Analyseanwendungen und das Internet of Things (IoT) zu erstellen, ohne jegliche Serverinfrastruktur zu verwalten. Diese Architekturen können die Kosten reduzieren, da Sie weder für evtl. unausgelastete Server zahlen noch redundante Infrastruktur bereitstellen, um hohe Verfügbarkeit zu implementieren.

> *Sie können Ihren Code an den Datenverarbeitungsservice von **AWS Lambda** hochladen. Der Service führt dann den Code mithilfe von AWS-Infrastruktur in Ihrem Namen aus. Bei AWS Lambda werden Ihnen alle 100 ms, die Ihr Code ausgeführt wird, und die Anzahl der Auslösungen Ihres Codes in Rechnung gestellt. Durch die Verwendung von Amazon API Gateway können Sie praktisch unbegrenzt skalierbare synchrone APIs unter Verwendung von AWS Lambda entwickeln. In Kombination mit Amazon S3 zur Bereitstellung statischer Inhaltskomponenten kann dieses Konzept eine komplette Webanwendung bereitstellen. Weitere Informationen über diese Art von Architektur finden Sie im Whitepaper „Serverlose mehrstufige AWS-Architekturen“[21].*

Bei Mobil-Apps gibt es eine weitere Möglichkeit, den Umfang einer serverbasierten Infrastruktur zu reduzieren: Sie können Amazon Cognito nutzen, sodass Sie keine Back-End-Lösung für Benutzerauthentifizierung, Netzwerkstatus, Speicher und Synchronisierung verwalten müssen. Amazon Cognito generiert eindeutige Kennungen für Ihre Benutzer. Diese können in Ihren Zugriffsrichtlinien referenziert werden, um auf Benutzerbasis den Zugriff auf andere AWS-Ressourcen zu ermöglichen oder einzuschränken. Amazon Cognito bietet temporäre AWS-Zugriffsberechtigungen für Ihre Benutzer, sodass die auf dem Gerät ausgeführte Mobil-App direkt mit über AWS Identity and Access Management (IAM) geschützten AWS-Services interagieren kann. Durch Verwendung von IAM können Sie beispielsweise den Zugriff auf einen Ordner innerhalb eines Amazon S3-Buckets auf einen bestimmten Endbenutzer eingrenzen.

Bei IoT-Anwendungen mussten Unternehmen üblicherweise ihre eigenen Server als Geräte-Gateways bereitstellen, betreiben, skalieren und warten, um die Kommunikation zwischen den verbundenen Geräten und ihren Services zu handhaben. AWS IoT bietet ein vollständig verwaltetes Geräte-Gateway, das

automatisch anhand Ihrer Nutzung skaliert wird, ohne dass betrieblicher Overhead für Sie anfällt.

Datenbanken

Bei herkömmlicher IT-Infrastruktur waren Unternehmen oft auf die Datenbank- und Speichertechnologien beschränkt, die sie nutzen konnten.
Es gab häufig Einschränkungen in Bezug auf Lizenzierungskosten und die Fähigkeit, unterschiedliche Datenbank-Engines zu unterstützen. Bei AWS werden diese Einschränkungen durch verwaltete Datenbankservices eliminiert, die Leistung der Enterprise-Klasse bei Open-Source-Kosten bieten. Aus diesem Grund ist es nicht ungewöhnlich, wenn Anwendungen auf einer mehrsprachigen Datenebene ausgeführt werden, bei der für jede Workload die richtige Technologie gewählt wird.

Bestimmen der richtigen Datenbanktechnologie für jede Workload

Die folgenden Fragen können Ihnen bei der Entscheidung helfen, welche Lösungen in Ihrer Architektur enthalten sein sollten:

- *Handelt es sich um eine leseintensive, schreibintensive oder ausgewogene Workload? Wie viele Lese- und Schreibvorgänge pro Sekunde benötigen Sie? Wie ändern sich diese Werte, wenn sich die Anzahl der Benutzer erhöht?*

- *Wie viele Daten müssen Sie speichern, und wie lange? Wie schnell werden diese voraussichtlich anwachsen? Gibt es in absehbarer Zeit eine Höchstgrenze? Was ist die Größe jedes Objekts (durchschnittlich, minimal, maximal)? Wie wird auf diese Objekte zugegriffen?*

- *Was sind die Anforderungen in Bezug auf die Beständigkeit von Daten? Wird dieser Datenspeicher Ihre „Source of Truth“ sein?*

- *Was sind Ihre Latenzanforderungen? Wie viele Benutzer sollen gleichzeitig unterstützt werden?*

- *Was ist Ihr Datenmodell und wie fragen Sie die Daten ab? Sind Ihre Abfragen relational (z. B. JOINs zwischen mehreren Tabellen)? Könnten Sie Ihr Schema denormalisieren, um flachere Datenstrukturen zu erstellen, die einfacher zu skalieren sind?*

- *Welche Funktionen benötigen Sie? Benötigen Sie starke Integritätskontrollen oder mehr Flexibilität (z. B. schemalose Datenspeicher)? Benötigen Sie anspruchsvolle Berichts- oder Suchfunktionen? Sind Ihre Entwickler mit relationalen Datenbanken vertrauter als mit NoSQL?*

In diesem Kapitel werden die verschiedenen Kategorien von Datenbanktechnologien erörtert.

Relationale Datenbanken

Relationale Datenbanken (oft als RDBS oder SQL-Datenbanken bezeichnet) normalisieren Daten in gut definierte tabellarische Strukturen, die als Tabellen bezeichnet werden und aus Zeilen und Spalten bestehen. Sie bieten eine leistungsstarke Abfragesprache, flexible Indizierungsfunktionen, starke Integritätskontrollen und die Möglichkeit, Daten aus mehreren Tabellen schnell und effizient zu kombinieren. Amazon Relational Database Service (Amazon RDS) erleichtert das Einrichten, Betreiben und Skalieren einer relationalen Datenbank in der Cloud.

Skalierbarkeit

Relationale Datenbanken können vertikal skaliert werden (z. B. durch Aktualisieren auf eine größere Amazon RDS-DB-Instance oder Hinzufügen von mehr und schnellerem Speicher). Erwägen Sie außerdem die Verwendung von Amazon RDS for Aurora, einer Datenbank-Engine, die darauf ausgelegt ist, beim Ausführen auf der gleichen Hardware einen wesentlich höheren Durchsatz als Standard-MySQL zu bieten. Für leseintensive Anwendungen können Sie auch über die Kapazitätsbeschränkungen einer einzelnen DB-Instance hinaus horizontal skalieren, indem Sie eine oder mehrere Read Replicas erstellen.

> ***So nutzen Sie die Vorteile von Read Replicas***
>
> *Read Replicas sind separate Datenbank-Instances, die asynchron repliziert werden. Das hat zur Folge, dass sie Replikationsverzögerung unterliegen und ihnen möglicherweise einige der neuesten Transaktionen fehlen. Anwendungsdesigner müssen abwägen, welche Abfragen Toleranz für leicht veraltete Daten haben. Diese Abfragen können auf einer Read Replica ausgeführt werden, während die übrigen auf dem primären Knoten ausgeführt werden sollten. Read Replicas können auch keine Schreibabfragen verarbeiten.*

Workloads relationaler Datenbanken, die ihre Schreibkapazität über die Beschränkungen einer einzelnen DB-Instance hinaus skalieren müssen, erfordern einen anderen Ansatz, der als *Datenpartitionierung* oder *Sharding* bezeichnet wird. Bei diesem Modell werden Daten über mehrere Datenbankschemas verteilt, die jeweils in einer eigenen autonomen primären DB-Instance ausgeführt werden. Amazon RDS eliminiert zwar den betrieblichen Overhead für die Ausführung dieser Instances, Sharding verursacht jedoch

einige Komplexität für die Anwendung. Die Datenzugriffsebene der Anwendung muss geändert werden, damit Sie wissen, wie Daten aufgeteilt werden, damit Abfragen an die richtige Instance weitergeleitet werden können. Darüber hinaus müssen Schemaänderungen in mehreren Datenbankschemas durchgeführt werden. Daher lohnt es sich, Anstrengungen zu unternehmen, diesen Prozess zu automatisieren.

Hohe Verfügbarkeit

Für jede relationale Datenbank für die Produktion empfehlen wir die Verwendung der Multi-AZ-Bereitstellungsfunktion von Amazon RDS. Diese erstellt eine synchron replizierte Standby-Instance in einer anderen Availability Zone (AZ). Bei einem Ausfall des primären Knotens führt Amazon RDS ein automatisches Failover auf die Standby-Instance aus, ohne dass ein manueller administrativer Eingriff erforderlich ist. Wenn ein Failover durchgeführt wird, gibt es einen kurzen Zeitraum, in dem der primäre Knoten nicht verfügbar ist. Resiliente Anwendungen können durch das Anbieten reduzierter Funktionalität (z. B. schreibgeschützter Modus mithilfe von Read Replicas) auf Graceful Failure ausgelegt werden.

Anti-Patterns

Wenn Ihre Anwendung primär Daten indiziert und abfragt und keine Join-Vorgänge oder komplexen Transaktionen benötigt (insbesondere, wenn Sie von einem Schreibdurchsatz über die Beschränkungen einer einzelnen Instance hinaus ausgehen), könnte eine NoSQL-Datenbank die richtige Lösung sein. Wenn Sie über große Binärdateien (Audio-, Video- und Bilddateien) verfügen, ist es effizienter, die tatsächlichen Dateien im Amazon Simple Storage Service (Amazon S3) zu speichern und nur die Metadaten für die Dateien in Ihrer Datenbank zu speichern.

Weitere Informationen über bewährte Methoden für relationale Datenbanken finden Sie in der Dokumentation zu Amazon RDS.[22]

NoSQL-Datenbanken

Mit dem Begriff NoSQL werden Datenbanken beschrieben, die einen Teil der Abfrage- und Transaktionsfunktionen relationaler Datenbanken in ein flexibleres Datenmodell eintauschen, das sich nahtlos horizontal skalieren lässt. NoSQL-Datenbanken nutzen verschiedene Datenmodelle, darunter Diagramme, Schlüssel-Wert-Paare und JSON-Dokumente. NoSQL-Datenbanken sind

bekannt für ihre einfache Bereitstellung, skalierbare Leistung, hohe Verfügbarkeit und Resilienz. Amazon DynamoDB ist ein schneller und flexibler NoSQL-Datenbank-Service[23] für Anwendungen, die eine konsistente Latenz im einstelligen Millisekundenbereich für beliebig große Datenmengen benötigen. Es handelt sich um eine vollständig verwaltete Clouddatenbank, die sowohl Dokument- als auch Schlüssel-Wert-Datenmodelle unterstützt.

Skalierbarkeit

NoSQL-Datenbank-Engines führen normalerweise eine Datenpartitionierung und -replikation durch, um sowohl die Lese- als auch die Schreibvorgänge horizontal zu skalieren. Sie tun dies auf transparente Weise, ohne dass die Datenpartitionierungslogik in der Datenzugriffsebene Ihrer Anwendung implementiert werden muss. Insbesondere Amazon DynamoDB verwaltet die Tabellenpartitionierung automatisch für Sie und fügt neue Partitionen hinzu, wenn sich Ihre Tabelle vergrößert oder die für Lese- und Schreibvorgänge bereitgestellte Kapazität sich ändert.

Um die Skalierbarkeit von Amazon DynamoDB bei der Entwicklung Ihrer Anwendung optimal zu nutzen, lesen Sie das Kapitel mit bewährten Methoden für Amazon DynamoDB[24] in der Dokumentation.

Hohe Verfügbarkeit

Der Amazon DynamoDB-Service repliziert synchron Daten in drei Einrichtungen in einer AWS-Region, um bei einem Serverausfall oder einer Störung einer Availability Zone Fehlertoleranz zu bieten.

Anti-Patterns

Wenn Ihr Schema nicht denormalisiert werden kann und Ihre Anwendung Join-Vorgänge oder komplexe Transaktionen erfordert, könnte eine relationale Datenbank geeigneter sein. Wenn Sie über große Binärdateien (Audio-, Video- und Bilddateien) verfügen, empfiehlt es sich vielleicht, die Dateien in Amazon S3 zu speichern und die Metadaten für die Dateien in Ihrer Datenbank zu speichern.

Weitere Informationen zum Migrieren von einer relationalen Datenbank zu DynamoDB oder zur Frage, welche Workloads migriert werden sollten, finden Sie im Whitepaper „Bewährte Methoden zum Migrieren von RDBMS zu Amazon DynamoDB“[25].

Data Warehouse

Ein Data Warehouse ist ein spezieller Typ von relationaler Datenbank, der für die Analyse und Berichterstellung bei großen Datenmengen optimiert ist. Er kann verwendet werden, um transaktionale Daten aus unterschiedlichen Quellen zu kombinieren (z. B. Benutzerverhalten bei einer Webanwendung, Daten aus Ihrem Finanz- und Abrechnungssystem, CRM usw.) und sie zur Analyse und Entscheidungsfindung verfügbar zu machen.

Üblicherweise ist die Einrichtung, Ausführung und Skalierung eines Data Warehouse kompliziert und kostspielig. Bei AWS können Sie Amazon Redshift nutzen, einen verwalteten Data-Warehouse-Service, der auf weniger als ein Zehntel der Betriebskosten herkömmlicher Lösungen ausgelegt ist.

Skalierbarkeit

Amazon Redshift erreicht effiziente Speicherung und optimale Abfrageleistung durch eine Kombination aus Massively Parallel Processing (MPP), spaltenbasierter Datenspeicherung und gezielten Codierungsschemas für die Datenkomprimierung. Es eignet sich besonders zur Analyse und Berichterstellung bei sehr großen Datensätzen. Mit der MPP-Architektur von Amazon Redshift können Sie die Leistung erhöhen, indem Sie die Anzahl der Knoten in Ihrem Data-Warehouse-Cluster erhöhen.

Hohe Verfügbarkeit

Amazon Redshift bietet mehrere Funktionen zum Steigern der Zuverlässigkeit Ihres Data-Warehouse-Clusters. Wir empfehlen, Produktions-Workloads in Clustern mit mehreren Knoten bereitzustellen, in denen Daten, die in einen Knoten geschrieben werden, automatisch auf andere Knoten im Cluster repliziert werden. Die Daten werden auch kontinuierlich auf Amazon S3 gesichert. Amazon Redshift überwacht den Status des Clusters kontinuierlich, Daten von fehlerhaften Laufwerken werden automatisch neu repliziert und Knoten werden nach Bedarf ersetzt. Weitere Informationen finden Sie in den häufig gestellten Fragen zu Amazon Redshift[26].

Anti-Patterns

Da Amazon Redshift ein SQL-basiertes Managementsystem für relationale Datenbanken (RDBMS) ist, ist es mit anderen RDBMS-Anwendungen und Business-Intelligence-Tools kompatibel. Amazon Redshift bietet zwar die Funktionalität eines typischen RDBMS, einschließlich Funktionen zur

Onlinetransaktionsverarbeitung (Online Transaction Processing, OLTP), ist jedoch nicht auf diese Workloads ausgelegt. Wenn Sie von einer Workload mit hoher Parallelität ausgehen, die in der Regel das Lesen und Schreiben aller Spalten für jeweils eine kleine Anzahl von Datensätzen umfasst, ist Amazon RDS oder Amazon DynamoDB vermutlich geeigneter.

Suche

Anwendungen, die ausgefeilte Suchfunktionen erfordern, entwachsen im Allgemeinen früher oder später den Fähigkeiten von relationalen und NoSQL-Datenbanken. Ein Suchservice kann zum Indizieren und Durchsuchen sowohl strukturierter als auch freier Textformate verwendet werden und Funktionen unterstützen, die in anderen Datenbanken nicht verfügbar sind, z. B. anpassbare Ergebnissortierung, Facettierung für Filterung, Synonyme, Stammverknüpfung usw.

Bei AWS haben Sie die Wahl zwischen Amazon CloudSearch und Amazon Elasticsearch Service (Amazon ES). Amazon CloudSearch ist ein Managed Service, der nur wenig Konfiguration erfordert und automatisch skaliert wird. Amazon ES bietet dagegen eine Open-Source-API und gibt Ihnen mehr Kontrolle über die Konfigurationsdetails. Amazon ES wurde zudem weiterentwickelt und ist nun sehr viel mehr als nur eine Suchlösung. Der Service wird häufig als Analyse-Engine für Anwendungsfälle wie Protokollanalysen, Echtzeit-Anwendungsüberwachung und Click-Stream-Analysen verwendet.

Skalierbarkeit

Sowohl Amazon CloudSearch als auch Amazon ES verwenden Datenpartitionierung und Replikation für horizontale Skalierung. Der Unterschied besteht darin, dass Sie sich mit Amazon CloudSearch keine Gedanken über die Anzahl der Partitionen und Replikate machen müssen. Der Service übernimmt dies automatisch für Sie.

Hohe Verfügbarkeit

Beide Services bieten Funktionen, mit denen Daten redundant über mehrere Availability Zones hinweg gespeichert werden. Weitere Informationen finden Sie in der Dokumentation des jeweiligen Services.

Entfernen von Single Points of Failure

Produktionssysteme bieten meist definierte oder implizite Ziele in Bezug auf Verfügbarkeit. Ein System ist hoch verfügbar, wenn es Ausfällen von einzelnen oder mehreren Komponenten (z. B. Festplatten, Servern, Netzwerklinks usw.) standhält. Überlegen Sie sich, wie Sie Unterbrechungen auf allen Architekturebenen verringern können und wie die Wiederherstellung automatisiert werden kann. Dieses Kapitel befasst sich mit Entwurfskonzepten für hohe Verfügbarkeit. Weitere Informationen zu diesem Thema finden Sie im Whitepaper „Erstellen fehlertoleranter Anwendungen“ [27].

Einführung von Redundanz

Single Points of Failure können durch die Einführung von Redundanz entfernt werden. Dabei handelt es sich um mehrere Ressourcen für die gleiche Aufgabe. Redundanz kann entweder im Standby- oder Aktiv-Modus implementiert werden.

Wenn bei *Standby-Redundanz* eine Ressource ausfällt, wird die Funktionalität in einer zweiten Ressource wiederhergestellt. Diesen Prozess nennt man *Failover*. Der Abschluss eines Failovers nimmt in der Regel einige Zeit in Anspruch, während der die Ressource nicht verfügbar ist. Die sekundären Ressourcen können entweder nur bei Bedarf automatisch gestartet werden (zur Kostensenkung) oder bereits ausgeführt werden (um den Failover zu beschleunigen und Unterbrechungen zu verringern). Standby-Redundanz wird häufig für zustandsbehaftete Komponenten wie relationale Datenbanken verwendet.

Bei *aktiver Redundanz*. werden Anfragen an mehrere redundante Datenverarbeitungsressourcen verteilt. Wenn eine Ressource ausfällt, können die übrigen einfach einen größeren Teil der Workload übernehmen. Aktive Redundanz sorgt, im Gegensatz zur Standby-Redundanz, für bessere Auslastung. Außerdem sind die Auswirkungen durch Ausfälle geringer.

Erkennen von Ausfällen

Bei der Erkennung von und dem Reagieren auf Ausfälle sollten Sie für so viel Automatisierung sorgen wie möglich. Sie können mit Services wie ELB und Amazon Route 53 Zustandsprüfungen konfigurieren und Fehler maskieren, indem Sie Datenverkehr an intakte Endpunkte leiten. Darüber hinaus kann

Auto Scaling so konfiguriert werden, dass fehlerhafte Knoten automatisch ersetzen werden. Sie können auch fehlerhafte Knoten ersetzen, indem Sie die automatische Wiederherstellung in Amazon EC2[28] oder Services wie AWS OpsWorks und AWS Elastic Beanstalk nutzen. Es ist nicht möglich, von Anfang an alle möglichen Ausfallszenarien vorherzusagen. Sammeln Sie ausreichend Protokolle und Metriken, um den Normalzustand des Systems besser zu verstehen. Anhand dieser Erkenntnisse können Sie Alarme einrichten, die manuelle oder automatische Reaktionen hervorrufen.

Entwerfen von guten Zustandsprüfungen

Wenn die Zustandsprüfungen für Ihre Anwendungen richtig konfiguriert sind, können Sie schnell und richtig auf eine Vielzahl von Ausfallszenarien reagieren. Falsche Zustandsprüfungen hingegen können die Verfügbarkeit Ihrer Anwendung gefährden.

Mit dem Elastic Load Balancing-Service können Sie Zustandsprüfungen für typische dreistufige Anwendungen konfigurieren. Entwerfen Sie Ihre Zustandsprüfungen so, dass der Zustand der Back-End-Knoten zuverlässig beurteilt werden kann. Eine einfache TCP-Zustandsprüfung ist nicht in der Lage, Szenarien zu erkennen, in denen die Instance selbst fehlerfrei ist, aber der Webserver-Prozess abgestürzt ist. Stattdessen sollte geprüft werden, ob der Webserver bei einer einfachen Anfrage eine HTTP-200-Antwort zurückgeben kann.

Auf dieser Ebene ist es nicht sinnvoll, eine umfassende Zustandsprüfung (einen Test, der von anderen Anwendungsebenen abhängig ist) durchzuführen. Dies könnte Falschmeldungen zur Folge haben. Beispiel: Wenn Ihre Zustandsprüfung ebenfalls prüft, ob sich die Instance mit einer Back-End-Datenbank verbinden kann, kann es sein, dass alle Ihre Webserver als fehlerhaft gekennzeichnet werden, wenn der Datenbankknoten für kurze Zeit nicht verfügbar ist. Ein Ansatz auf mehreren Ebenen ist häufig am besten. Gegebenenfalls sollte eine umfassende Zustandsprüfung auf Amazon Route 53-Ebene implementiert werden. Indem Sie eine ganzheitlichere Prüfung durchführen, die feststellt, ob diese Umgebung tatsächlich die erforderliche Funktionalität bereitstellen kann, können Sie Amazon Route53 so konfigurieren, dass ein Failover auf eine statische Version Ihrer Webseite durchgeführt wird, bis Ihre Datenbank wieder einsatzbereit ist.

Beständige Datenspeicher

Ihre Anwendung und Ihre Benutzer erstellen und verwalten eine Vielzahl von Daten. Es ist wichtig, dass Ihre Architektur sowohl die Verfügbarkeit als auch die Integrität der Daten schützt. Mit Datenreplikation werden redundante Kopien Ihrer Daten erstellt. Die horizontale Skalierung der Lesekapazität wird so vereinfacht und die Beständigkeit und Verfügbarkeit von Daten wird

erhöht. Die Replikation kann in verschiedenen Modi erfolgen.
Bei *synchroner Replikation* wird eine Transaktion nur dann anerkannt, wenn sie sowohl am primären Speicherort als auch in den Replikaten dauerhaft gespeichert wurde. Diese Form der Replikation eignet sich ideal zum Schutz der Datenintegrität im Fall eines Ausfalls des primären Knotens. Die synchrone Replikation dient auch der Skalierung der Lesekapazität für Abfragen, die hochaktuelle Daten erfordern (starke Konsistenz). Der Nachteil der synchronen Replikation ist, dass der primäre Knoten an die Replikate gekoppelt ist. Eine Transaktion kann erst anerkannt werden, wenn der Schreibvorgang aller Replikate beendet wurde. Dies kann die Leistung und Verfügbarkeit beeinträchtigen (insbesondere bei Topologien, die über unzuverlässige oder latenzreiche Netzwerkverbindungen ausgeführt werden). Aus demselben Grund ist es nicht empfehlenswert, viele synchrone Replikate zu nutzen.

> ***Beständigkeit: Kein Ersatz für Sicherungen***
>
> *Egal, wie beständig Ihre Lösung ist – sie ist kein Ersatz für regelmäßige Sicherungen. Bei der synchronen Replikation werden alle Ihre Datenupdates redundant gespeichert, auch solche, die durch Software- oder menschliche Fehler entstanden sind. Sie können jedoch insbesondere für Objekte, die in Amazon S3 gespeichert sind, Versioning[29] nutzen, um Objektversionen aufzubewahren, abzurufen und wiederzuherstellen. Daten lassen sich dank Versioning nach unbeabsichtigten Benutzeraktionen und Anwendungsfehlern wiederherstellen.*

Asynchrone Replikation trennt den primären Knoten von seinen Replikaten. Dabei kommt es zur Replikationsverzögerung. Das bedeutet, dass am primären Knoten durchgeführte Änderungen nicht sofort auf die Replikate übertragen werden. Asynchrone Replikate werden zur horizontalen Skalierung der Systemlesekapazität genutzt, wenn Abfragen die Replikationsverzögerung tolerieren können. Asynchrone Replikation eignet sich auch, um die Datenbeständigkeit zu erhöhen, wenn der Verlust von einigen aktuellen Transaktionen bei einem Failover toleriert werden kann. Sie können beispielsweise ein asynchrones Replikat einer Datenbank in einer separaten AWS-Region speichern, um es bei einer Notfallwiederherstellung zu nutzen.

Quorum-basierte Replikation kombiniert synchrone und asynchrone Replikation, um die Herausforderungen großer verteilter Datenbanksysteme zu bewältigen. Die Replikation auf mehrere Knoten verwalten Sie, indem Sie eine Mindestanzahl von Knoten definieren, die an einen erfolgreichen Schreibvorgang beteiligt sein müssen. Eine detaillierte Beschreibung verteilter Datenspeicher

kann im Rahmen dieses Dokuments nicht behandelt werden. Im Whitepaper zu Amazon Dynamo[30] erfahren Sie mehr über die Grundprinzipien für ein hoch skalierbares und hochzuverlässiges Datenbanksystem.

> ***Datenbeständigkeit in der Praxis***
>
> *Es ist wichtig zu verstehen, wie sich jede von Ihnen genutzte Technologie in diese Datenspeichermodelle einfügt. Ihr Verhalten während verschiedener Failover- oder Sicherungs-/Wiederherstellungs-Szenarien sollte zu Ihrem Recovery Point Objective (RPO) und Recovery Time Objective (RTO) passen. Sie müssen herausfinden, wie viele verlorene Daten Sie erwarten und wie schnell Sie den Betrieb wieder aufnehmen zu können. Beispiel: Die Redis-Engine für Amazon ElastiCache unterstützt Replikation mit automatischem Failover. Jedoch ist die Replikation mit Redis Engine asynchron. Während eines Failovers ist es sehr wahrscheinlich, dass einige aktuelle Transaktionen verloren gehen. Amazon RDS mit Multi-AZ-Funktion ist jedoch so konzipiert, dass synchrone Replikate erstellt werden, um die Daten am Standby-Knoten auf dem Stand des Primärknotens zu halten.*

Automatisierte Resilienz für Multi-Rechenzentren

Auch geschäftskritische Anwendungen müssen vor Unterbrechungsszenarien geschützt werden, die weit mehr als nur eine einzelne Festplatte, einen Server oder ein Rack betreffen. Herkömmliche Infrastrukturen verfügen normalerweise über einen Plan zur Notfallwiederherstellung, um einen Failover zu einem entfernten zweiten Rechenzentrum zu erlauben, wenn es zu großen Unterbrechungen im Hauptrechenzentrum kommt. Aufgrund der großen Entfernung zwischen den beiden Rechenzentren ist es aufgrund der Latenz unpraktisch, synchrone rechenzentrumsübergreifende Kopien der Daten aufrechtzuerhalten. Dies hat zur Folge, dass ein Failover mit großer Sicherheit zu Datenverlusten oder einem sehr teuren Datenwiederherstellungsprozess führt. Dies führt zu einem risikoreichen und ggf. zu einem nicht ausreichend getesteten Verfahren. Dieses Modell eignet sich jedoch hervorragend zum Schutz gegen Risiken, die zwar wenig wahrscheinlich, aber folgenreich sind – z. B. Naturkatastrophen, die zu einem längeren Ausfall der Infrastruktur führen. Informationen dazu, wie Sie diesen Ansatz in AWS implementieren, erhalten Sie im Whitepaper „Verwendung von AWS zur Notfallwiederherstellung“[31].

Der wahrscheinlichere Fall ist, dass es zu kürzeren Unterbrechungen im Rechenzentrum kommt. Bei kürzeren Unterbrechungen, bei denen kein längerer Ausfall erwartet wird, ist die Wahl eines Failovers schwierig und wird daher meist vermieden. AWS bietet einen einfacheren und effizienteren Schutz

vor Ausfällen dieser Art. Jede AWS-Region verfügt über mehrere Standorte, die als Availability Zones bezeichnet werden. Jede Availability Zone ist so konzipiert, dass sie nicht von Ausfällen in anderen Availability Zones beeinflusst wird. Eine Availability Zone ist ein Rechenzentrum. In manchen Fällen besteht eine Availability Zone auch aus mehreren Rechenzentren. Die Availability Zones einer Region bieten kostengünstige, latenzarme Netzwerkverbindungen zu anderen Zones in der gleichen Region. So können Sie Daten in Rechenzentren synchron replizieren. Auf diese Weise kann der Failover automatisiert werden und ist transparent für Ihre Benutzer.

Es ist auch möglich, aktive Redundanz zu implementieren, damit Sie nicht für ungenutzte Ressourcen zahlen. Eine Flotte von Anwendungsservern kann beispielsweise über mehrere Availability Zones hinweg verteilt und mit dem Elastic Load Balancing-Service (ELB) verbunden werden. Wenn die Amazon EC2-Instances einer bestimmten Availability Zone die Zustandsprüfungen nicht bestehen, sendet ELB keinen Datenverkehr mehr an diese Knoten. Sofern vorhanden, sorgt Auto Scaling dafür, dass die Anzahl fehlerfreier Knoten automatisch mit den anderen Availability Zones ausgeglichen wird, ganz ohne manuelles Zutun.

Viele der höheren AWS-Services wurden gemäß dem Muli-AZ-Prinzip entwickelt. Amazon RDS bietet beispielsweise hohe Verfügbarkeit und automatische Failover-Unterstützung für DB-Instances durch Verwendung von Multi-AZ-Bereitstellungen. Bei Amazon S3 und Amazon DynamoDB werden Ihre Daten redundant an mehreren Standorten gespeichert.

Fehlerisolation und herkömmliche horizontale Skalierung

Das aktive Redundanzmuster eignet sich hervorragend zum Ausgleich von Datenverkehr und für Unterbrechungen bei Instances oder Availability Zones. Jedoch ist es nicht ausreichend, wenn Anfragen als solche schädlich sind. Es sind Szenarien denkbar, bei denen alle Instances betroffen sind. Wenn eine bestimmte Anfrage einen Fehler auslöst, der zum Ausfall des Systems führt, könnte der Aufrufer weitere Fehler auslösen, indem dieselbe Anfrage wiederholt auf allen Instances durchgeführt wird.

Shuffle Sharding

Eine Methode der Fehlerisolation für die horizontale Skalierung ist das *Sharding*. Die Methode ähnelt derjenigen für herkömmliche Datenspeicher. Anstatt dass Sie den Datenverkehr von allen Kunden auf jeden Knoten verteilen,

können Sie die Instances in Shards gruppieren. Wenn Sie zum Beispiel über acht Instances für Ihren Service verfügen, können Sie vier Shards von jeweils zwei Instances erstellen (zwei Instances für etwas Redundanz innerhalb jedes Shards) und Kunden zu einem bestimmten Shard zuweisen. Auf diese Weise können Sie die Auswirkungen auf Kunden proportional zur Anzahl der Shards verringern. Dennoch werden weiterhin Kunden betroffen sein. Daher ist es wichtig, dass der Client fehlertolerant ist. Wenn der Client jeden Endpunkt in einer Reihe von Shard-Ressourcen ausprobieren kann, bis einer erfolgreich ist, erhalten Sie eine dramatische Verbesserung. Diese Methode wird als *Shuffle Sharding* bezeichnet. Mehr erfahren Sie im entsprechenden Blogbeitrag[32].

Kostenoptimierung

Allein durch die Verlagerung bestehender Architekturen in die Cloud können Unternehmen ihre Kapitalkosten senken und Einsparungen durch die Skaleneffekte von AWS erzielen. Durch die Iteration und Verwendung von mehreren AWS-Funktionen können weitere kostengünstige Cloud-Architekturen erstellt werden. In diesem Abschnitt werden die wichtigsten Grundsätze der Kostenoptimierung mit AWS Cloud Computing erklärt.

Richtige Dimensionierung

AWS bietet eine breite Palette an Ressourcentypen und Konfigurationen für eine Vielzahl von Anwendungsfällen. Beispiel: Services wie Amazon EC2, Amazon RDS, Amazon Redshift und Amazon Elasticsearch Service (Amazon ES) bieten Ihnen viele unterschiedliche Instance-Typen. In einigen Fällen sollten Sie den günstigsten Workload-Typ wählen, der Ihren individuellen Anforderungen entspricht. In anderen Fällen könnten weniger Instances eines größeren Instance-Typs zu geringeren Gesamtkosten oder einer besseren Leistung führen. Sie sollten den richtigen Instance-Typ benchmarken und auswählen, je nachdem, wie Ihre Workload CPU, RAM, Netzwerk, Speicher und E/A-Größe nutzt.

Ebenso können Sie Kosten sparen, indem Sie die richtige Speicherlösung für Ihre Anforderungen auswählen. Beispiel: Amazon S3 bietet eine Vielzahl von Speicherklassen, einschließlich Standard, Reduced Redundancy und Standard-Infrequent Access. Andere Services, z. B. Amazon EC2, Amazon RDS und Amazon ES unterstützen verschiedene Amazon Elastic Block Store-Volume-Typen (Amazon EBS), wie (Magnetfestplatten, Allzweck-SSD, bereitgestellte IOPS-SSD), sollten Sie bewerten.

Kontinuierliche Überwachung und Tagging

Kostenoptimierung ist ein iterativer Prozess. Ihre Anwendung und deren Nutzung entwickeln sich mit der Zeit weiter. Darüber hinaus iteriert AWS häufig und veröffentlicht regelmäßig neue Optionen.

AWS stellt Tools[33] bereit, mit denen Sie Kosten sparen und Ihre Ressourcen passend dimensionieren können. Um die Ergebnisse dieser Tools richtig zu interpretieren, sollten Sie Tagging-Richtlinien für Ihre AWS-Ressourcen definieren und implementieren. Sie können das Tagging zu einem Teil Ihres Erstellungsprozesses machen und mit AWS-Verwaltungstools wie AWS Elastic Beanstalk und AWS OpsWorks automatisieren. Sie können auch die verwalteten Regeln von AWS Config nutzen, um zu prüfen, ob Ihre Ressourcen mit bestimmten Tags versehen werden oder nicht.

Elastizität

Sie können mit AWS weitere Kosten zu sparen, indem Sie sich die Elastizität der Plattform zunutze machen. Planen Sie die Implementierung von Auto Scaling für so viele Amazon EC2-Workloads wie möglich. So können Sie bei Bedarf horizontal hochskalieren und automatisch zurückskalieren und Ausgaben verringern, wenn Sie nicht mehr so viel Kapazität benötigen. Darüber hinaus können Sie automatisch Nicht-Produktions-Workloads deaktivieren, wenn diese nicht in Gebrauch sind[34]. Letztlich sollten Sie sich überlegen, welche Datenverarbeitungs-Workloads Sie in AWS Lambda implementieren könnten, sodass Sie nie für ungenutzte oder redundante Ressourcen zahlen müssen.

Ersetzen Sie, falls möglich, Amazon EC2-Workloads mit AWS Managed Services, bei denen Sie keine Kapazitätsentscheidungen treffen müssen (z. B. ELB, Amazon CloudFront, Amazon SQS, Amazon Kinesis Firehose, AWS Lambda, Amazon SES, Amazon CloudSearch) oder bei denen Sie Kapazität einfach und nach Bedarf anpassen können (z. B. Amazon DynamoDB, Amazon RDS, Amazon Elasticsearch Service).

Nutzen Sie die verschiedenen Kaufoptionen

Amazon EC2 On-Demand-Instance-Preise bieten Ihnen maximale Flexibilität ohne langfristige Verpflichtungen. Es gibt zwei weitere Möglichkeiten zur Zahlung für Amazon EC2-Instances, mit denen Sie Kosten sparen können: Reserved Instances und Spot-Instances.

Reservierte Kapazität

Reserved Instances ermöglichen die Reservierung von Amazon EC2-

Rechenkapazität und bieten einen wesentlich günstigeren Stundenpreis im Vergleich zu On-Demand-Instance-Preisen. Diese Lösung eignet sich optimal für Anwendungen mit vorhersehbaren Mindestkapazitätsanforderungen. Sie können Tools wie AWS Trusted Advisor oder Amazon EC2-Nutzungsberichte nutzen, um herauszufinden, welche Rechenressourcen Sie besonders häufig nutzen und reservieren sollten. Die Preisnachlässe in Ihrer monatlichen Rechnung richten sich danach, welche Reserved Instances Sie erworben haben. Technisch gesehen gibt es keinen Unterschied zwischen einer On-Demand-EC2-Instance und einer Reserved Instance. Der Unterschied liegt in der Art und Weise, wie Sie für Instances zahlen, die Sie reservieren.

Optionen für reservierte Kapazität gibt es auch bei anderen Services (z. B. Amazon Redshift, Amazon RDS, Amazon DynamoDB und Amazon CloudFront).

> ***Tipp:*** *Sie sollten sich nicht zum Kauf einer Reserved Instance verpflichten, bevor Sie nicht Ihre Anwendung in der Produktion ausreichend gebenchmarkt haben. Nachdem Sie reservierte Kapazität erworben haben, können Sie die Nutzungsberichte für Reserved Instances verwenden, um sicherzustellen, dass Sie die reservierte Kapazität weiterhin sinnvoll nutzen.*

Spot-Instances

Für weniger beständige Workloads eignen sich Spot-Instances. Mit Amazon EC2 Spot-Instances können Sie auf nicht genutzte Amazon-EC2-Rechenkapazität bieten. Da Spot-Instances häufig zu einem niedrigeren Preis verfügbar sind (verglichen mit On-Demand-Instances), können Sie Ihre Anwendungen deutlich kostengünstiger betreiben.

Spot-Instances sind ideal für Workloads mit flexiblen Start- und Endzeiten. Ihre Spot-Instance wird gestartet, wenn Ihr Gebot den aktuellen Spot-Marktpreis übersteigt, und wird so lange ausgeführt, bis Sie sie beenden oder bis der Spot-Marktpreis Ihr Gebot übersteigt. Wenn der Spot-Marktpreis über Ihrem Gebotspreis liegt, wird die Instance automatisch beendet. Für die angebrochene Stunde, in der die Instance ausgeführt wurde, werden Ihnen keine Gebühren berechnet.

Spot-Instances eignen sich also ideal für Workloads, bei denen Unterbrechungen toleriert werden können. Sie können jedoch auch Spot-Instances verwenden, wenn eine vorhersehbarere Verfügbarkeit erforderlich ist:

Bietstrategie: Solange die Spot-Instance ausgeführt wird, wird Ihnen der Spot-Marktpreis (nicht der Gebotspreis) berechnet. Ihre Bietstrategie könnte

sein, deutlich höher zu bieten. Selbst wenn der Marktpreis hin und wieder sehr hoch ist, erzielen Sie auf lange Sicht betrachtet deutliche Einsparungen.

Kombination mit On-Demand-Instances: Ziehen Sie es in Betracht, Reserved, On-Demand- und Spot-Instances zu kombinieren. Auf diese Weise erhalten Sie eine vorhersehbare Mindestkapazität und bedarfsgerechten Zugriff auf zusätzliche Rechenressourcen abhängig vom Spot-Marktpreis. Dies ist eine ausgezeichnete Methode zur Verbesserung von Durchsatz oder Anwendungsleistung.

Spot-Blöcke für Workloads mit definierter Dauer: Sie können auch auf Spot-Instances mit fest definierter Dauer bieten. Diese haben unterschiedliche Stundenpreise. Sie können jedoch eine Dauer festlegen. Wenn Ihr Gebot akzeptiert wird, wird die Instance so lange ausgeführt, bis Sie sie beenden oder bis die angegebene Dauer abgelaufen ist. Die Instance wird dann nicht aufgrund von Änderungen im Spot-Preis beendet. (Dennoch sollten Sie bei Ihrer Planung Fehlertoleranz berücksichtigen, denn auch eine Spot-Instance kann, wie jede andere EC2-Instance, ausfallen.)

Bewährte Methoden bei Spot-Preisen

Spot-Instances erlauben es Ihnen, auf mehrere Instance-Typen gleichzeitig bieten. Da die Preise für jeden Instance-Typ einer Availability Zone unabhängig voneinander schwanken, können Sie oft mehr Rechenleistung für den gleichen Preis erhalten, wenn Ihre Anwendung so konzipiert wurde, dass sie flexibel in Bezug auf Instance-Typen ist. Testen Sie Ihre Anwendung, falls möglich, auf verschiedenen Instance-Typen. Bieten Sie auf alle Instance-Typen, die Ihre Anforderungen erfüllen, um die Kosten weiter zu senken.

Cache-Speicherung

Beim Caching werden zuvor berechnete Daten gespeichert, um sie später zu nutzen. Caching wird verwendet, um die Anwendungsleistung zu verbessern und die Kosteneffizienz einer Implementierung zu steigern. Caching kann auf mehreren Ebenen einer IT-Architektur angewendet werden.

Anwendungsdaten-Caching

Anwendungen können so gestaltet werden, dass sie Informationen aus schnellen, verwalteten In-Memory-Caches speichern und abrufen können. Zwischengespeicherte Informationen umfassen auch die Ergebnisse ein- und

ausgabeintensiver Datenbankabfragen oder berechnungsintensiver Verarbeitung. Ist ein Ergebnissatz nicht im Cache auffindbar, kann die Anwendung ihn berechnen oder aus einer Datenbank abrufen. Er kann dann für folgende Anfragen im Cache gespeichert werden. Wenn jedoch ein Ergebnissatz im Cache gefunden wird, kann die Anwendung ihn direkt verwenden. Dies verbessert die Latenz für Endbenutzer und reduziert die Last auf Back-End-Systemen. Ihre Anwendung kann steuern, wie lange die zwischengespeicherten Elemente gültig sein sollen. In manchen Fällen kann es bereits sinnvoll sein, beliebte Objekte nur wenige Sekunden zwischenzuspeichern. Dies kann zu einer deutlichen Entlastung Ihrer Datenbank führen.

Amazon ElastiCache ist ein Web-Service, mit dem ein In-Memory-Cache auf einfache Weise in der Cloud bereitgestellt, betrieben und skaliert werden kann. Der Service unterstützt zwei Open-Source In-Memory-Caching-Engines: Memcached und Redis. Weitere Informationen zur Auswahl der richtigen Engine für Ihre Workload und eine Beschreibung gängiger ElastiCache-Entwurfskonzepte finden Sie im Whitepaper „Optimale Leistung mit Amazon ElastiCache“.[35]

Edge-Caching

Kopien statischer Inhalte (z. B. Bilder, CSS-Dateien, Streams von vorab aufgezeichneten Videos) und dynamischer Inhalte (z. B. HTML-Antwort, Live-Video) können in Amazon CloudFront zwischengespeichert werden. Amazon CloudFront ist ein Netzwerk zur Bereitstellung von Inhalten (Content Delivery Network, CDN) mit mehreren Edge-Standorten auf der ganzen Welt. Edge-Caching ermöglicht die Bereitstellung von Inhalten durch eine Infrastruktur, die näher an den Besuchern Ihrer Webseite liegt. Dadurch verringert sich die Latenzzeit und Sie erhalten hohe, anhaltende Datenübertragungsraten, die für die Bereitstellung großer, beliebter Objekte an Endbenutzer in großem Maßstab erforderlich sind.

Anfragen für Ihre Inhalte werden an Amazon S3 oder Ihre Ursprungs-Server weitergeleitet. Wenn der Ursprung in AWS ausgeführt wird, werden Anfragen über optimierte Netzwerkpfade übertragen, was für verbesserte Zuverlässigkeit und Konsistenz sorgt. Amazon CloudFront kann verwendet werden, um Ihre gesamte Webseite bereitzustellen, einschließlich Inhalten, die nicht zwischengespeichert werden können. Der Vorteil in diesem Fall ist, dass Amazon CloudFront bestehende Verbindungen zwischen der Amazon

CloudFront-Edge und dem Ursprungsserver wiederverwendet. So verkürzt sich die Latenzzeit für die Verbindungseinrichtung jeder Ursprungsanfrage. Andere Verbindungsoptimierungen helfen ebenfalls, Internet-Engpässe zu vermeiden und die verfügbare Bandbreite zwischen dem Edge-Standort und dem Webseitenbesucher voll auszunutzen. Amazon CloudFront kann so die Bereitstellung Ihrer dynamischen Inhalte beschleunigen und den Besuchern Ihrer Webseite eine einheitliche und zuverlässige, aber dennoch persönlich angepasste Erfahrung bieten. Amazon CloudFront bietet dieselben Leistungsvorteile für Upload-Anfragen wie für Anfragen zum Herunterladen von dynamischen Inhalten.

Sicherheit

Die meisten der Sicherheitstools und -methoden, die Sie vielleicht bereits aus einer herkömmlichen IT-Infrastruktur kennen, können in der Cloud verwendet werden. Zusätzlich kann AWS zur Verbesserung der Sicherheit beitragen. AWS ist eine Plattform, die es Ihnen ermöglicht, das Design von Sicherheitskontrollen direkt auf der Plattform zu formalisieren. AWS vereinfacht die Systemnutzung für Administratoren und IT-Verantwortliche und sorgt für eine einfachere regelmäßige Überprüfung Ihrer Umgebung. In diesem Abschnitt erhalten Sie eine allgemeine Übersicht über die bewährten Methoden für die Sicherheit in AWS. Umfassende Informationen darüber, wie Sie ein hohes Maß an Sicherheits-Governance erreichen, finden Sie in den Whitepapers „Sicherheit nach Maß: Governance in AWS“[36] und „Bewährte AWS-Sicherheitsmethoden“[37].

Nutzen Sie AWS-Funktionen für einen „Defense-in-Depth“-Schutz

AWS bietet eine Vielzahl von Funktionen, die IT-Architekten dabei helfen, einen „Defense-in-Depth“-Schutz umzusetzen. Auf Netzwerkebene können Sie eine VPC-Topologie erstellen, die Teile der Infrastruktur durch die Verwendung von Subnetzen, Sicherheitsgruppen und Routing-Kontrollen isoliert. Services wie AWS WAF (eine Firewall für Webanwendungen) schützen Ihre Webanwendungen vor SQL Injections und anderen Sicherheitslücken in Ihrem Anwendungscode. Bei der Zugriffskontrolle können Sie mithilfe von IAM granulare Richtlinien definieren und diese Benutzern, Gruppen und AWS-Ressourcen zuordnen. Die AWS-Plattform bietet zudem zahlreiche Optionen zum Schutz von Daten bei der Übertragung oder im Ruhezustand mit Verschlüsselung[38]. Eine umfassende Liste aller Sicherheitsfunktionen kann im Rahmen dieses Dokuments nicht behandelt

werden. Sie können jedoch mehr erfahren, indem Sie die AWS-Seite zur Sicherheit besuchen[39].

Auslagern von Sicherheitsverantwortung nach AWS

AWS arbeitet mit einem Modell der geteilten Verantwortung für Sicherheit. Dabei ist AWS für die Sicherheit der zugrunde liegenden Cloudinfrastruktur verantwortlich. Sie hingegen sind für die Sicherung der Workloads, die Sie in AWS bereitstellen, verantwortlich. So müssen Sie weniger Verantwortung übernehmen und können sich auf Ihre Kernkompetenzen konzentrieren, indem Sie AWS Managed Services nutzen. Wenn Sie beispielsweise Services wie Amazon RDS, Amazon ElastiCache, Amazon CloudSearch usw. nutzen, fallen Sicherheits-Patches in den Verantwortungsbereich von AWS. Dies senkt nicht nur den betrieblichen Overhead für Ihr Team, sondern kann Sie auch vor Sicherheitslücken schützen.

Einschränken von Zugriffsberechtigungen

Wenn Sie Server als programmierbare Ressourcen behandeln, können Sie davon auch im Sicherheitsbereich profitieren. Wenn Sie Ihre Server jederzeit wechseln können, wird der Zugriff von Gastbetriebssystemen auf die Produktionsumgebungen überflüssig. Wenn auf einer Instance ein Problem auftritt, können Sie sie automatisch oder manuell beenden und ersetzen. Bevor Sie jedoch Instances ersetzen, sollten Sie Protokolle für Ihre Instances erfassen und zentral speichern. Sie können Ihnen dabei helfen, Probleme in Ihrer Entwicklungsumgebung nachzubilden und sie als Fehlerbehebungen in Ihrem kontinuierlichen Bereitstellungsprozess bereitzustellen. Dies ist besonders in elastischen Rechenumgebungen mit temporären Servern wichtig. Sie können Amazon CloudWatch Logs zur Datenerfassung verwenden. Wenn Sie keinen Zugriff haben, aber benötigen, können Sie einen Just-in-Time-Zugriff implementieren, indem Sie eine API-Aktion nutzen, mit der das Netzwerk bei Bedarf zu Verwaltungszwecken geöffnet werden kann. Sie können diese Zugriffsanfragen in Ihr Ticketsystem integrieren, sodass Zugriffsanfragen erst nach der Genehmigung verfolgt und dynamisch bearbeitet werden.

Eine weitere häufige Ursache für Sicherheitsrisiken ist die Verwendung von Servicekonten. In einer herkömmlichen Umgebung werden Servicekonten häufig langfristige Anmeldeinformationen zugewiesen, die in einer Konfigurationsdatei gespeichert werden. In AWS können Sie stattdessen IAM-Rollen nutzen, um Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt

werden, zu erteilen. Dabei werden kurzzeitige Anmeldeinformationen genutzt. Diese Anmeldeinformationen werden automatisch verteilt und rotiert. Bei mobilen Anwendungen ermöglicht Amazon Cognito Client-Geräten kontrollierten Zugriff auf AWS-Ressourcen über temporäre Token. Für Benutzer der AWS-Managementkonsole können Sie Federated Access gewähren, indem Sie temporäre Token nutzen, anstatt IAM-Benutzer in Ihrem AWS-Konto zu erstellen. Mitarbeiter, die das Unternehmen verlassen, werden so nicht nur aus dem Identitätsverzeichnis des Unternehmens gelöscht, sondern können auch nicht mehr auf Ihr AWS-Konto zugreifen.

Sicherheit als Code

Bei herkömmlichen Sicherheits-Frameworks, Vorschriften und organisatorischen Richtlinien werden Sicherheitsanforderungen definiert, indem Methoden wie z. B. Firewall-Regeln, Netzwerk-Zugriffskontrollen, interne/externe Subnetze und Betriebssystemhärtung genutzt werden. Diese Methoden können in einer AWS-Umgebung implementiert werden, aber Sie haben nun die Möglichkeit, sie alle in einem Skript zu vereinen, das eine „Golden Environment“ definiert. Das bedeutet, dass Sie ein AWS CloudFormation-Skript zur Erfassung und zuverlässigen Bereitstellung von Sicherheitsrichtlinien erstellen können. Die bewährten Sicherheitsmethoden können jetzt für mehrere Projekte wiederverwendet werden und in die Pipeline für kontinuierliche Integration aufgenommen werden. Sie können Sicherheitstests als Teil Ihres Veröffentlichungszyklus durchführen und automatisch Anwendungslücken und Abweichungen von Ihrer Sicherheitsrichtlinie feststellen.

Für noch mehr Kontrolle und Sicherheit können AWS CloudFormation-Vorlagen als „Produkte“ in den AWS Service Catalog importiert werden.[40] Dies ermöglicht die zentrale Verwaltung von Ressourcen zur Unterstützung konsistenter Governance-, Sicherheits- und Compliance-Anforderungen. Zudem können Benutzer schneller nur die genehmigten IT-Services bereitstellen, die sie benötigen. Mit IAM-Berechtigungen können Sie festlegen, wer Ihre Produkte anzeigen und ändern darf. Zudem definieren Sie Einschränkungen, um festzulegen, wie bestimmte AWS-Ressourcen für ein Produkt bereitgestellt werden können.

Echtzeit-Audits

Tests und Audits Ihrer Umgebung sind der Schlüssel zu schnellem, aber sicherem Handeln. Herkömmliche Ansätze, wie zyklische Überprüfungen (die

oft manuell oder stichprobenartig durchgeführt werden), reichen nicht aus, insbesondere nicht in agilen Umgebungen, die sich stetig wandeln. In AWS können kontinuierliche Überwachung und automatisierte Kontrollen implementiert werden. Dies verstärkt den Schutz vor Sicherheitsrisiken. Services wie AWS Config, Amazon Inspector und AWS Trusted Advisor prüfen kontinuierlich auf Compliance und Schwachstellen. Sie bieten eine klare Übersicht darüber, welche IT-Ressourcen die Compliance-Anforderungen einhalten und welche nicht. Mit AWS Config-Regeln können Sie prüfen, ob eine Komponente kurzzeitig die Compliance-Anforderungen nicht eingehalten hat. Dadurch steigern Sie die Effektivität von Point-in-Time- und Period-in-Time-Audits. Umfangreiche Protokolle für Ihre Anwendungen können Sie mit Amazon CloudWatch Logs implementieren. Die Protokollierung für AWS API-Aufrufe erfolgt durch die Aktivierung von AWS CloudTrail.[41] AWS CloudTrail ist ein Webservice, der API-Aufrufe an unterstützte AWS-Services in Ihrem AWS-Konto aufzeichnet und eine Protokolldatei an Ihren Amazon S3-Bucket überträgt. Protokolle können dann unveränderlich gespeichert und automatisch verarbeitet werden, um Sie zu benachrichtigen oder sogar Maßnahmen in Ihrem Namen zu ergreifen. So schützen Sie Ihr Unternehmen vor fehlender Compliance. Sie können AWS Lambda, Amazon EMR, Amazon Elasticsearch Service oder Drittanbieter-Tools aus dem AWS Marketplace nutzen, um Protokolle zu durchsuchen. So erkennen Sie ungenutzte Berechtigungen, übermäßige Verwendung berechtigter Konten, die Nutzung von Schlüsseln, irreguläre Anmeldungen, Richtlinienverletzungen und Systemmissbrauch.

Fazit

Dieses Whitepaper bietet Leitlinien für das Entwerfen von Architekturen, die die Leistungsfähigkeit der AWS-Plattform optimal ausschöpfen, und behandelt wichtige Prinzipien und Gestaltungsmuster: von der Auswahl der richtigen Datenbank für Ihre Anwendung bis zur Architektur von Anwendungen, die horizontal skaliert werden können und hohe Verfügbarkeit bieten. Jeder Anwendungsfall ist einzigartig, deshalb müssen Sie abschätzen, inwieweit die Fälle auf Ihre Implementierung angewendet werden können. Das Thema Cloud-Computing-Architekturen ist umfangreich und entwickelt sich stetig weiter. Sie können sich auf dem neuesten Stand halten, indem Sie die Fülle von Materialien nutzen, die Sie auf der AWS-Webseite finden, und die Schulungs- und Zertifizierungsangebote von AWS in Anspruch nehmen.

Autoren

Dieses Dokument ist unter der Mitarbeit folgender Person entstanden:

- Andreas Chatzakis, Manager, AWS Solutions Architecture

Weitere Informationen

Weitere Architekturbeispiele finden Sie im AWS-Architekturzentrum[42]. Für Anwendungen, die bereits in AWS ausgeführt werden, empfehlen wir, auch das Whitepaper „AWS Well-Architected Framework“[43] zu lesen, das dieses Dokument ergänzt und ein strukturiertes Evaluierungsmodell bietet. Um Ihre betriebliche Bereitschaft zu überprüfen, können Sie auch die umfangreiche AWS-Checkliste für den Betrieb[44] lesen.

Hinweise

[1] Informationen zu AWS: https://aws.amazon.com/about-aws/

[2] Die globale AWS-Infrastruktur: https://aws.amazon.com/about-aws/global-infrastructure/

[3] Es gibt beispielsweise den PHP Amazon DynamoDB Session Handler (http://docs.aws.amazon.com/aws-sdk-php/v3/guide/service/dynamodb-session-handler.html) und den Tomcat Amazon DynamoDB Session Handler (http://docs.aws.amazon.com/AWSSdkDocsJava/latest//DeveloperGuide/java-dg-tomcat-session-manager.html)

[4] ELB Sticky Sessions http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-sticky-sessions.html

[5] Whitepaper „Big Data-Analyseoptionen in AWS“ https://do.awsstatic.com/whitepapers/Big_Data_Analytics_Options_on_AWS.pdf

[6] Bootstrapping mit Benutzerdatenskripten und cloud-init: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html

[7] AWS Opsworks-Lebenszyklusereignisse http://docs.aws.amazon.com/opsworks/latest/userguide/workingcookbook-events.html

[8] AWS Lambda-gestützte benutzerdefinierte CloudFormation-Ressourcen: http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-custom-resources-lambda.html

[9] Amazon Machine Images http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

[10] AMIs für die AWS Elastic Beanstalk-Laufzeiten: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.platforms.html

[11] AWS Elastic Beanstalk-Anpassungen mit Konfigurationsdateien: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html

[12] AWS Elastic Beanstalk: https://aws.amazon.com/elasticbeanstalk/

[13] Automatische Wiederherstellung in Amazon EC2: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-recover.html

[14] Auto Scaling: https://aws.amazon.com/autoscaling/

[15] Amazon CloudWatch-Alarme: http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/AlarmThatSendsEmail.html

[16] Amazon CloudWatch Events: http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/WhatIsCloudWatchEvents.html

[17] AWS OpsWorks-Lebenszyklus http://docs.aws.amazon.com/opsworks/latest/userguide/workingcookbook-events.html

[18] AWS Lambda und geplante Ereignisse: http://docs.aws.amazon.com/lambda/latest/dg/with-scheduled-events.html

[19] Exponentielles Backoff und Jitter http://www.awsarchitectureblog.com/2015/03/backoff.html

[20] Die vollständige Liste aller AWS-Produkte finden Sie hier: http://aws.amazon.com/products/

[21] Whitepaper „Serverlose mehrstufe AWS-Architekturen“ https://do.awsstatic.com/whitepapers/AWS_Serverless_Multi-Tier_Archiectures.pdf

[22] Bewährte Methoden für Amazon RDS: http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html

[23] NoSQL-Datenbanken in AWS https://aws.amazon.com/nosql/

[24] Bewährte Methoden für DynamoDB: http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/BestPractices.html

[25] Bewährte Methoden zum Migrieren von RDBMS zu Amazon DynamoDB: https://do.awsstatic.com/whitepapers/migration-best-practices-rdbms-to-dynamodb.pdf

[26] Amazon RedShift – Häufig gestellte Fragen: https://aws.amazon.com/redshift/faqs/

[27] Whitepaper „Erstellung fehlertoleranter Anwendungen“: https://d0.awsstatic.com/whitepapers/aws-building-fault-tolerant-applications.pdf

[28] Wiederherstellung einer Instance: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-recover.html

[29] Amazon S3-Versioning: http://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html

[30] „Dynamo: Amazon’s Highly Available Key-value Store“ http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html

[31] „Verwendung von AWS zur Notfallwiederherstellung“ https://media.amazonwebservices.com/AWS_Disaster_Recovery.pdf

[32] Shuffle Sharding http://www.awsarchitectureblog.com/2014/04/shuffle-sharding.html

[33] Überwachung Ihrer Nutzung und Kosten http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/monitoring-costs.html

[34] Erstellen von Alarmen, die eine Instance stoppen oder beenden http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/UsingAlarmActions.html

[35] „Optimale Leistung mit Amazon ElastiCache“: https://d0.awsstatic.com/whitepapers/performance-at-scale-with-amazon-elasticache.pdf

[36] „Sicherheit nach Maß: Governance in AWS“ https://d0.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Governance_in_AWS_Whitepaper.pdf

[37] „Bewährte AWS-Sicherheitsmethoden“: https://d0.awsstatic.com/whitepapers/aws-security-best-practices.pdf

[38] Schützen ruhender Daten mit Verschlüsselung: https://d0.awsstatic.com/whitepapers/aws-securing-data-at-rest-with-encryption.pdf

[39] AWS-Sicherheit: http://aws.amazon.com/security

[40] AWS Service Catalog: https://aws.amazon.com/servicecatalog/

[41] „Sicherheit nach Maß: Protokollierung in AWS“ https://do.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf

[42] AWS-Architekturzentrum https://aws.amazon.com/architecture

[43] „AWS Well-Architected Framework“: http://do.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

[44] AWS-Checkliste für den Betrieb http://media.amazonwebservices.com/AWS_Operational_Checklists.pdf