EXPERT INSIGHT

AWS for Solutions Architects

The definitive guide to AWS Solutions Architecture for migrating to, building, scaling, and succeeding in the cloud

Second Edition

Foreword by:
Dr. Siddhartha Choubey , Ph.D

Saurabh Shrivastava **Neelanjali Srivastav**
Alberto Artasanchez **Imtiaz Sayed**

packt

4

Networking in AWS

Enterprises today have become exponentially more agile by leveraging the power of the cloud. In this chapter, we will highlight the scale of AWS Global Infrastructure and teach you about AWS networking foundations.

Networking is the first step for any organization to set up its landing zone and the entire IT workload built on top of it. You could say that networking is the backbone of the IT application and infrastructure workload. AWS provides various networking services for building your IT landscape in the cloud and in this chapter, you will dive deep into AWS networking services.

Every business is now running at a global scale and organizations need to target global populations with their product. With a traditional on-premise IT workload, it becomes challenging to scale globally and provide the same user experience across the globe. AWS helps solve these problems through edge networking, and you will learn more about deploying your application for global users without compromising their experience. Furthermore, you will learn about network security and building a hybrid cloud.

In this chapter, you will learn about the following topics:

- Learning about the AWS Global Infrastructure
- AWS networking foundations
- Edge networking
- Building hybrid cloud connectivity in AWS
- AWS cloud network security

Without further ado, let's get down to business.

Learning about the AWS Global Infrastructure

The infrastructure offered by AWS is highly secure and reliable. It offers over 200 services. Most are available in all AWS Regions worldwide, spread across 245 countries. Regardless of the type of technology application you are planning to build and deploy, AWS is sure to provide a service that will facilitate its deployment.

AWS has millions of customers and thousands of consulting and technology partners worldwide. Businesses large and small across all industries rely on AWS to handle their workloads. Here are some statistics to give you an idea of the breadth of AWS's scale. AWS provides the following as its global infrastructure:

- 26 launched Regions and 8 announced Regions
- 84 Availability Zones
- Over 110 Direct Connect locations
- Over 310 Points of Presence
- 17 Local Zones and 32 announced LZs
- 24 Wavelength Zones

IMPORTANT NOTE

These numbers are accurate at the time of writing this book. By the time you read this, it would not be surprising for the numbers to have changed.

Now that we have covered how the AWS infrastructure is organized at a high level, let's learn about the elements of the AWS Global Infrastructure in detail.

Regions, Availability Zones, and Local zones

How can Amazon provide such a reliable service across the globe? How can they offer reliability and durability guarantees for some of their services? The answer reveals why they are the cloud leaders and why it's difficult to replicate what they offer. AWS has billions of dollars worth of infrastructure deployed across the world. These locations are organized into different Regions and Zones. More formally, AWS calls them the following:

- AWS Regions
- **Availability Zones (AZs)**
- **Local Zones (LZs)**

As shown in the following diagram, an AZ is comprised of multiple distinct data centers, each equipped with redundant power, networking, and connectivity, and located in separate facilities.

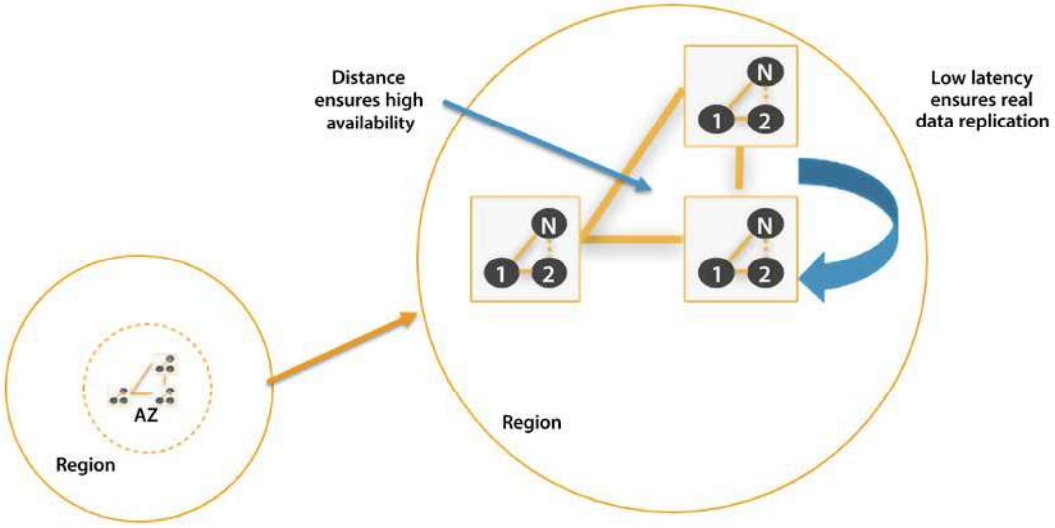

Figure 4.1: AWS Regions and AZs

AWS Regions exist in separate geographic areas. Each AWS Region comprises several independent and isolated data centers (AZs) that provide a full array of AWS services.

AWS is continuously enhancing its data centers to provide the latest technology. AWS's data centers have a high degree of redundancy. AWS uses highly reliable hardware, but the hardware is not foolproof. Occasionally, a failure can happen that interferes with the availability of resources in each data center. Suppose all instances were hosted in only one data center. If a failure occurred with the whole data center, then none of your resources would be available. AWS mitigates this issue by having multiple data centers in each Region.

By default, users will always be assigned a default Region. You can obtain more information about how to change and maintain AWS environment variables using the AWS CLI by visiting this link: `https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html`.

In the following subsection, we will look at AWS Regions in greater detail and see why they are essential.

AWS Regions

AWS Regions are groups of data centers in one geographic location that are specially designed to be independent and isolated from each other. A single Region consists of a collection of data centers spread within that Region's geographic boundary. This independence promotes availability and enhances fault tolerance and stability. While working on the console, you will see AWS services available in that Region. There is a possibility that a particular service is not available in your Region. Eventually, all services become **generally available** (**GA**) after their launch; however, the timing of availability may differ between different Regions. Some services are global – for example, **Direct Connect Gateway** (**DXGW**), **Identity and Access Management** (**IAM**), CloudFront, and Route53.

Other services are not global but allow you to create inter-Region fault tolerance and availability. For example, Amazon **Relational Database Service** (**RDS**) allows you to create read replicas in multiple Regions. To find out more about this, visit `https://aws.amazon.com/blogs/aws/cross-region-read-replicas-for-amazon-rds-for-mysql/`.

One of the advantages of using such an architecture is that resources will be closer to users, increasing access speed and reducing latency.

Another obvious advantage is that you can serve your clients without disruption even if a whole Region becomes unavailable by planning your disaster recovery workload to be in another Region. You will be able to recover faster if something goes wrong, as these read replicas can be automatically converted to the primary database if the need arises.

As of May 2022, there are 26 AWS Regions and 8 announced Regions. The naming convention that is usually followed is to list the country code, followed by the geographic region, and the number. For example, the US East Region in Ohio is named as follows:

- Location: US East (Ohio)
- Name: us-east-2

AWS has a global cloud infrastructure, so you can likely find a Region near your user base, with a few exceptions, such as Russia. If you live in Greenland, it may be a little further away from you – however, you will still be able to connect as long as you have an internet connection.

In addition, AWS has dedicated Regions specifically and exclusively for the US government called AWS GovCloud. This allows US government agencies and customers to run highly sensitive applications in this environment. AWS GovCloud offers the same services as other Regions, but it complies explicitly with requirements and regulations specific to the needs of the US government.

The full list of available Regions can be found here: `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-available-regions`.

As AWS continues to grow, do not be surprised if it offers similar Regions to other governments worldwide, depending on their importance and the demand they can generate.

AWS AZs

As we discussed earlier, AZs are components of AWS Regions. The clusters of data centers within a Region are called AZs. A single AZ consists of multiple data centers. These data centers are connected to each other using AWS-owned dedicated fiber optic cables and located within a 60-mile radius, which is far enough to avoid localized failures, yet achieves a faster data transfer between the data centers. AZs have multiple power sources, redundant connectivity, and redundant resources. All this translates into unparalleled customer service, allowing them to deliver highly available, fault-tolerant, and scalable applications.

The AZs within an AWS Region are interconnected. These connections have the following properties:

- Fully redundant
- High-bandwidth
- Low-latency
- Scalable
- Encrypted
- Dedicated

Depending on the service you are using, if you decide to perform a multi-AZ deployment, an AZ will automatically be assigned to the service, but for some services, you may be able to designate which AZ is to be used.

Every AZ forms a completely segregated section of the AWS Global Infrastructure, physically detached from other AZs by a substantial distance, often spanning several miles. Each AZ operates on a dedicated power infrastructure, providing customers with the ability to run production applications and databases that are more resilient, fault-tolerant, and scalable compared to relying on a single data center.

High-bandwidth, low-latency networking interconnects all the AZs – but what about fulfilling the need for low-latency bandwidth within a highly populated city? For that, AWS launched LZs. Let's learn more about them.

AWS LZs

While AZs focus on covering larger areas throughout regions, such as US-West and US-East, AWS fulfills the needs of highly populated cities through LZs. AWS LZs are newer components in the AWS infrastructure family. LZs place select services close to end users, allowing them to create AWS applications that deliver single-digit, millisecond responses. An LZ is the compute and storage infrastructure located close to high-population areas and industrial centers, and offers high-bandwidth, low-latency connectivity to the broader AWS infrastructure. Due to their proximity to the customer, LZs facilitate the delivery of applications that necessitate latency in single-digit milliseconds to end-users. As of March 2023, AWS had has 32 LZs.

AWS LZs can run various AWS services, such as Amazon Elastic Compute Cloud, Amazon Virtual Private Cloud, Amazon Elastic Block Store, Amazon Elastic Load Balancing, Amazon FSx, Amazon EMR, Amazon ElastiCache, and Amazon RDS in geographic proximity to your end users.

The naming convention for LZs is to use the AWS Region followed by a location identifier, for example, **us-west-2-lax-2a**. Please refer to this link for the latest supported services in LZs – `https://aws.amazon.com/about-aws/global-infrastructure/localzones/features/?nc=sn&loc=2`.

Now you have learned about the different components of the AWS Global Infrastructure, let's look at the benefits of using AWS infrastructure.

Benefits of the AWS Global Infrastructure

The following are the key benefits of using AWS's cloud infrastructure:

- **Security** – One of the most complex and risky tasks is to maintain security, especially when it comes to the data center's physical security. With AWS's shared security responsibility model, you offload infrastructure security to AWS and focus on the application security that matters for your business.
- **Availability** – One of the most important factors for the user's experience is to make sure your application is highly available, which means you need to have your workload deployed in a physically separated geographic location to reduce the impact of natural disasters. AWS Regions are fully isolated, and within each Region, the AZs are further isolated partitions of AWS infrastructure. You can use AWS infrastructure with an on-demand model to deploy your applications across multiple AZs in the same Region or any Region globally.

- **Performance** – Performance is another critical factor in retaining and increasing the user base. AWS provides low-latency network infrastructure by using redundant 100 GbE fiber, which leads to terabits of capacity between regions. Also, you can use AWS Edge AZs for applications that require low millisecond latency, such as 5G, gaming, AR/VR, and IoT.
- **Scalability** – When user demands increase, you must have the required capacity to scale your application. With AWS, you can quickly spin up resources, deploying thousands of servers in minutes to handle any user demand. You can also scale down when demand goes down and don't need to pay for any overprovisioned resources.
- **Flexibility** – With AWS, you can choose how and where to run your workloads; for example, you can run applications globally by deploying into any of the AWS Regions and AZs worldwide. You can run your applications with single-digit millisecond latencies by choosing AWS LZs or AWS Wavelength. You can choose AWS Outposts to run applications on-premises.

Now that you have learned about the AWS Global Infrastructure and its benefits, the next question that comes to mind is how am I going to use this infrastructure? Don't worry – AWS has you covered by providing network services that allow you to create your own secure logical data center in the cloud and completely control your IT workload and applications. Furthermore, these network services help you to establish connectivity to your users, employees, on-premises data centers, and content distributions. Let's learn more about AWS's networking services and how they can help you to build your cloud data center.

AWS networking foundations

When you set up your IT infrastructure, what comes to mind first? I have the servers now – how can I connect them to the internet and each other so that they can communicate? This connectivity is achieved by networking, without which you cannot do anything.

Networking concepts are the same when it comes to the cloud. In this chapter, you will not learn what networking is, but instead how to set up your private network in the AWS cloud and establish connectivity between the different servers in the cloud and from on-premises to an AWS cloud. First, let's start with the foundation; the first step to building your networking backbone in AWS is using Amazon VPC.

Amazon Virtual Private Cloud (VPC)

VPC is one of the core services AWS provides. Simply speaking, a VPC is your version of the AWS cloud, and as the name suggests, it is "private," which means that by default, your VPC is a logically isolated and private network inside AWS. You can imagine a VPC as being the same as your own logical data center in a virtual setting inside the AWS cloud, where you have complete control over the resources inside your VPC. AWS resources like AWS servers, and Amazon EC2 and Amazon RDS instances are placed inside the VPC, including all the required networking components to control the data traffic as per your needs.

Creating a VPC could be a very complex task, but AWS has made it easy by providing **Launch VPC Wizard**. You can visualize your network configuration when creating the VPC. The following screenshot shows the VPC network configuration across two AZs, **us-east-1a** and **us-east-1b**:

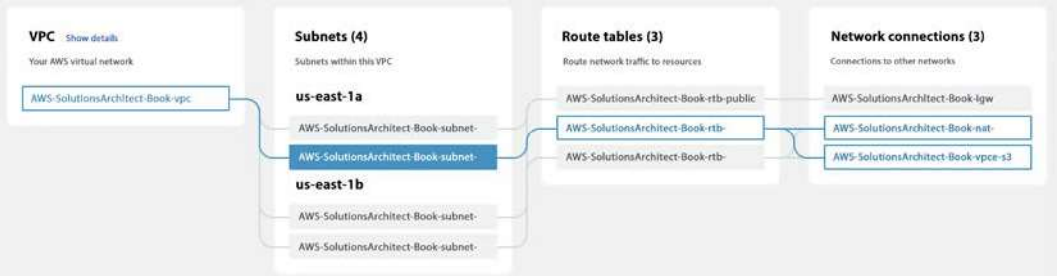

Figure 4.2: AWS VPC configuration with a private subnet flow

In the preceding diagram, you can see VPCs spread across two AZs, where each AZ has two subnets—one public and one private. The highlighted flow shows the data flow of a server deployed into a private subnet of the **us-east-1** AZ. Before going into further details, let's look at key VPC concepts to understand them better:

- **Classless Inter-Domain Routing (CIDR) blocks**: CIDR is the IP address range allocated to your VPC. When you create a VPC, you specify its set of IP addresses with CIDR notation. CIDR notation is a simplified way of showing a specific range of IP addresses. For example, `10.0.0.0/16` covers all IPs from `10.0.0.0` to `10.0.255.255`, providing 65,535 IP addresses to use. All resources in your VPC must fall within the CIDR range.
- **Subnets**: As the name suggests, the subnet is the VPC CIDR block subset. Partitions of the network are divided by the CIDR range within the range of IP addresses in your VPC. A VPC can have multiple subnets for different kinds of services or functions, like a frontend subnet (for internet access to a web page), a backend subnet (for business logic processing), and a database subnet (for database services).

Subnets create trusted boundaries between private and public resources. You should organize your subnets based on internet accessibility. A subnet allows you to define clear isolation between public and private resources. The majority of resources on AWS can be hosted in private subnets. You should use public subnets under controlled access and use them only when it is necessary. As you will keep most of your resources under restricted access, you should plan your subnets so that your private subnets have substantially more IPs available than your public subnets.

- **Route tables**: A routing table contains a set of rules called routes. Routes determine where the traffic will flow. By default, every subnet has a routing table. You can manually create a new route table and assign subnets to it. For better security, use the custom route table for each subnet.
- An **Internet Gateway (IGW)**: The IGW sits at the edge of the VPC and provides connectivity between your VPC resources and the public network (the internet). By default, internet accessibility is denied for internet traffic in your environment. An IGW needs to be attached to your public subnet through the subnet's route table, defining the rules to the IGW. All of your resources that require direct access to the internet (public-facing load balancers, NAT instances, bastion hosts, and so on) would go into the public subnet.
- **Network Address Translation (NAT) gateways**: A NAT gateway provides outbound internet access to the private subnet and prevents connections from being initiated from outside to your VPC resources. A private subnet blocks all incoming and outgoing internet traffic, but servers may need outgoing internet traffic for software and security patch installation. A NAT gateway enables instances in a private subnet to initiate outbound traffic to the internet and protects resources from incoming internet traffic. All restricted servers (such as database and application resources) should deploy inside your private subnet.
- **Security Groups (SGs)**: SGs are the virtual firewalls for your instances to control inbound and outbound packets. You can only use allow statements in the SG, and everything else is denied implicitly. SGs control inbound and outbound traffic as designated resources for one or more instances from the CIDR block range or another SG. As per the principle of least privilege, deny all incoming traffic by default and create rules that can filter traffic based on TCP, UDP, and **Internet Control Message Protocol (ICMP)**.
- **Network Access Control List (NACL)**: A NACL is another firewall that sits at the subnet boundary and allows or denies incoming and outgoing packets. The main difference between a NACL and an SG is that the NACL is stateless – therefore, you need to have rules for incoming and outgoing traffic. With an SG, you need to allow traffic in one direction, and return traffic is, by default, allowed.

You should use an SG in most places as it is a firewall at the EC2 instance level, while a NACL is a firewall at the subnet level. You should use a NACL where you want to put control at the VPC level and also deny specific IPs, as an SG cannot have a deny rule for network traffic coming for a particular IP or IP range.

- **Egress-only IGWs**: These provide outbound communication from **Internet Protocol version 6 (IPv6)** instances in your VPC to the internet and prevent the inbound connection from the internet to your instances on IPv6. IPv6, the sixth iteration of the Internet Protocol, succeeds IPv4 and employs a 128-bit IP address. Like IPv4, it facilitates the provision of unique IP addresses required for internet-connected devices to communicate.
- **DHCP option sets**: This is a group of network information, such as DNS name server and domain name used by EC2 instances when they launch.
- **VPC Flow Logs**: These enable you to monitor traffic flow to your system VPC, such as accepted and rejected traffic information for the designated resource to understand traffic patterns. Flow Logs can also be used as a security tool for monitoring traffic reaching your instance. You can create alarms to notify you if certain types of traffic are detected. You can also create metrics to help you identify trends and patterns.

To access servers in a private subnet, you can create a bastion host, which acts like a jump server. It needs to be hardened with tighter security so that only appropriate people can access it. To log in to the server, always use public-key cryptography for authentication rather than a regular user ID and password method.

A VPC resides only within an AWS Region where it can span across one or more AZs within the Region. In the following diagram, two AZs are being utilized within a Region. Furthermore, you can create one or more subnets inside each AZ, and resources like EC2 and RDS are placed inside the VPC in specific subnets. This architecture diagram shows a VPC configuration with a private and public subnet:

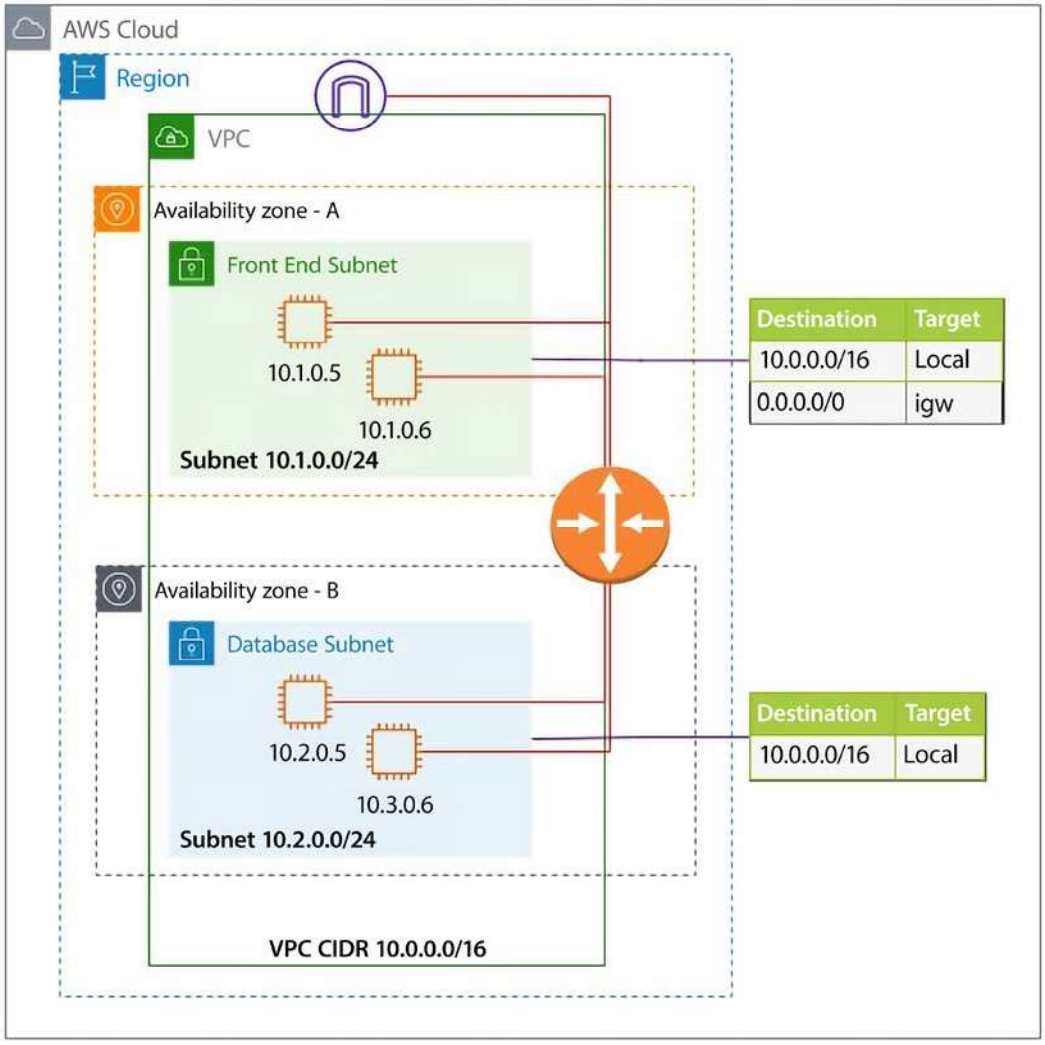

Figure 4.3: AWS VPC network architecture

As shown in this diagram, VPC subnets can be either private or public. As the name suggests, a private subnet doesn't have access to and from the internet, and a public subnet does. By default, any subnet you create is private; what makes it public is the default route – as in, `0.0.0.0/0`, via the IGW.

The VPC's route tables comprise directives for packet routing, and a default route table exists. However, unique route tables can be assigned to individual subnets. By default, all VPC subnets possess interconnectivity. This default behavior can be modified with VPC enhancements for more precise subnet routing, which enables configuration of subnet route tables that direct traffic between two subnets in a VPC through virtual appliances like intrusion detection systems, network firewalls, and protection systems.

As you can see, AWS provides multiple layers for network configuration and security at each layer that can help to build and protect your infrastructure. If attackers can access one component, they must restrict to limited resources by keeping them in their isolated subnet. Due to the ease of VPC creation and building tighter security, organizations tend to create multiple VPCs, which makes things more complicated when these VPCs need to communicate with each other. To simplify this, AWS provides **Transit Gateway (TGW)**. Let's learn more about it.

AWS TGW

As customers are spinning more and more VPCs in AWS, there is an ever-increasing need to connect various VPCs. Before TGW, you could connect VPCs using VPC peering, but VPC peering is a one-to-one connection, which means that resources within peered VPCs only can communicate with each other. If multiple VPCs need to communicate with each other, which is often the case, it results in a complex mesh of VPC peering. For example, a shown in the diagram below, if you have 5 VPCs, you need 10 peering connections.

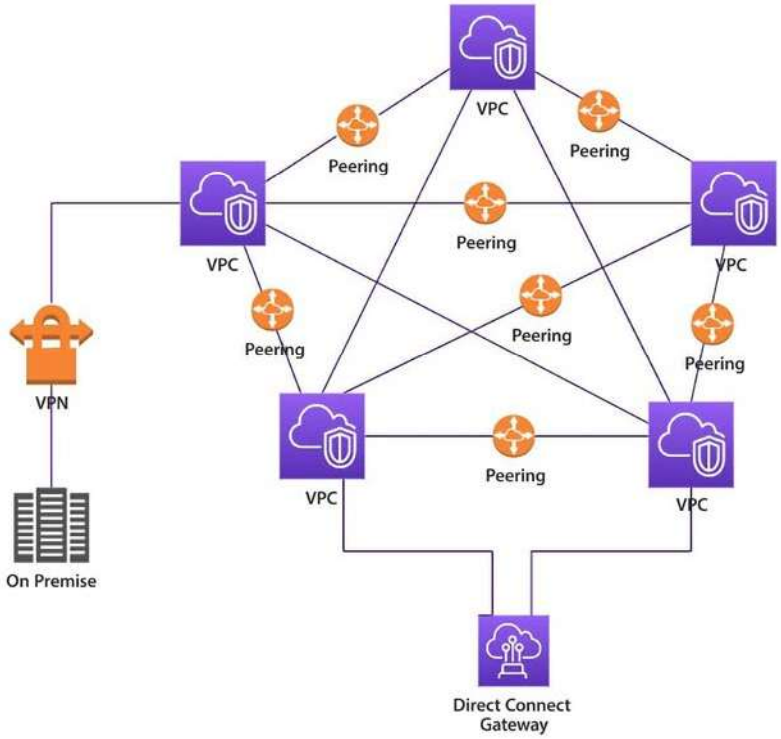

Figure 4.4: VPC connectivity using VPC peering without TGW

As you can see in the preceding diagram, managing so many VPC peering connections will become challenging, and there is also a limit on the number of peering connections per account. To overcome this challenge, AWS released TGW. TGW needs one connection called an attachment to a VPC, and you can establish full- or part-mesh connectivity easily without maintaining so many peering connections.

The following diagram shows simplified communication between five VPCs using TGW.

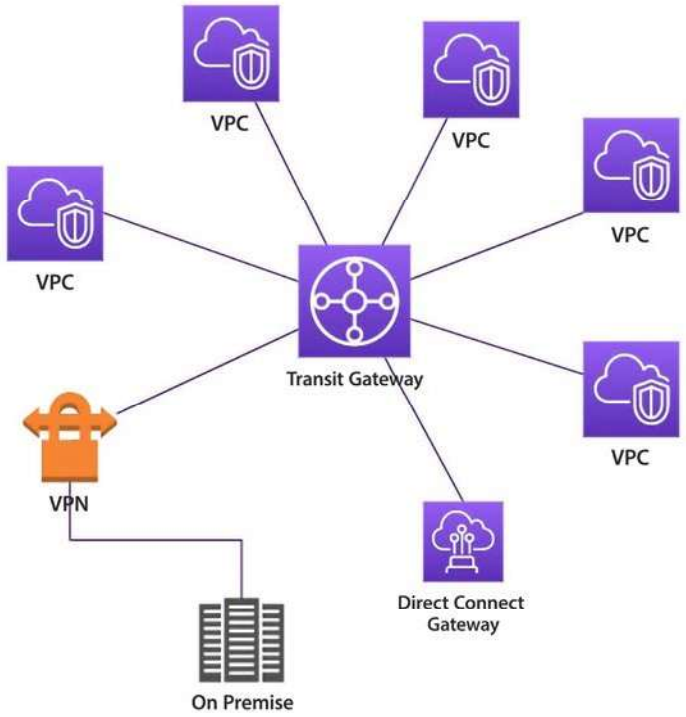

Figure 4.5: VPC connectivity with TGW

AWS TGW is a central aggregation service spanned within a Region, which can be used to connect your VPCs and on-premises networks. TGW is a managed service that takes care of your availability and scalability and eliminates complex VPN or peering connection scenarios when connecting with multiple VPCs and on-premises infrastructure. You can connect TGWs in different Regions by using TGW peering.

TGW is a Regional entity, meaning you can only attach VPCs to the TGW within the same Region, and per VPC, the bandwidth reserved is 50 Gbps. However, one TGW can have up to 5,000 VPC attachments.

In this section, you have learned how to establish network communication between VPCs, but what about securely connecting to resources such as Amazon S3 that live outside of a VPC or other AWS accounts? AWS provides PrivateLink to establish a private connection between VPCs and other AWS services. Let's learn more about AWS PrivateLink.

AWS PrivateLink

AWS PrivateLink establishes secure connectivity between VPCs and AWS services, preventing exposure of traffic to the internet. PrivateLink allows for the private connection of a VPC with supported AWS services hosted by different AWS accounts.

You can access AWS services from a VPC using the Gateway VPC endpoint and Interface VPC endpoint. Gateway endpoints do not support PrivateLink but allow for connection to Amazon S3 and DynamoDB without the need for an IGW or NAT device in your VPC. For other AWS services, an interface VPC endpoint can be created to establish a connection to services through AWS PrivateLink.

Enabling PrivateLink in AWS requires the creation of an endpoint network interface within the desired subnet, and assignment of a private IP address from the subnet address range for each specified subnet in the VPC. You can view the endpoint network interface in your AWS account, but you can't manage it yourself.

PrivateLink essentially provides access to the resources hosted in other VPC or other AWS accounts within the same subnet as the requester. This eliminates the need to use any NAT gateway, IGW, public IP address, or VPN. Therefore, it provides better control over your services, which are reachable via a client VPC.

As shown in the following diagram, AWS PrivateLink enables private connectivity between the **Service Provider** and **Service Consumer** using AWS infrastructure to exchange data without going over the public internet. To achieve this, the **Service Provider** creates an **Endpoint Service** in a private subnet.

In contrast, the **Service Consumer** creates an endpoint in a private subnet with the **Service Provider**'s service API as the target.

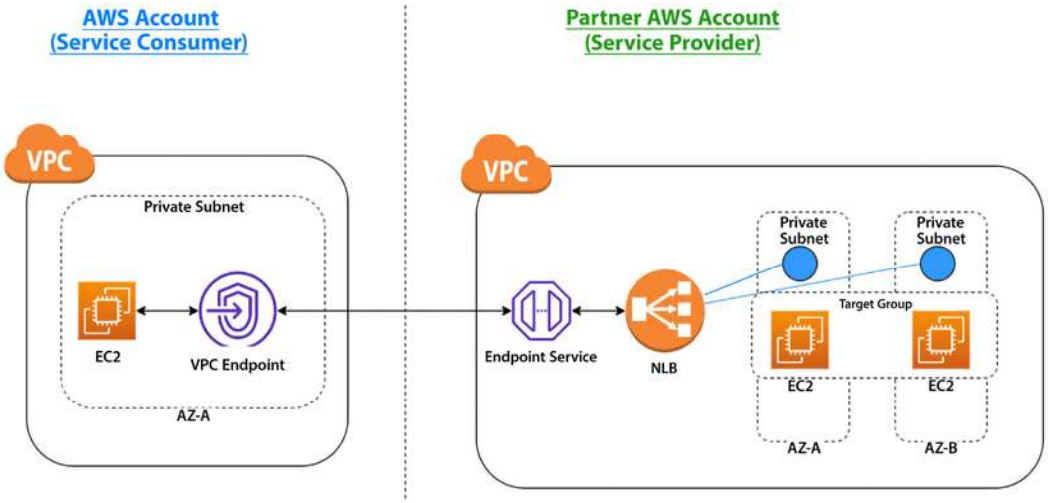

Figure 4.6: PrivateLink between partner Service Provider and Service Consumer accounts

As shown in the preceding architecture diagram, the partner sets up an **Endpoint Service** to expose the service running behind the load balancer (**NLB**). An **NLB** is created in each **Private Subnet**. These services are running on **EC2** instances hosted inside a **Private Subnet**. The client can then create a **VPC Endpoint** with the target as the **Endpoint Service** and use it to consume the service.

Let's look at another pattern shown in the following diagram, which depicts the use of PrivateLink between a **Service Consumer** on an AWS account and an on-premises **Service Provider**.

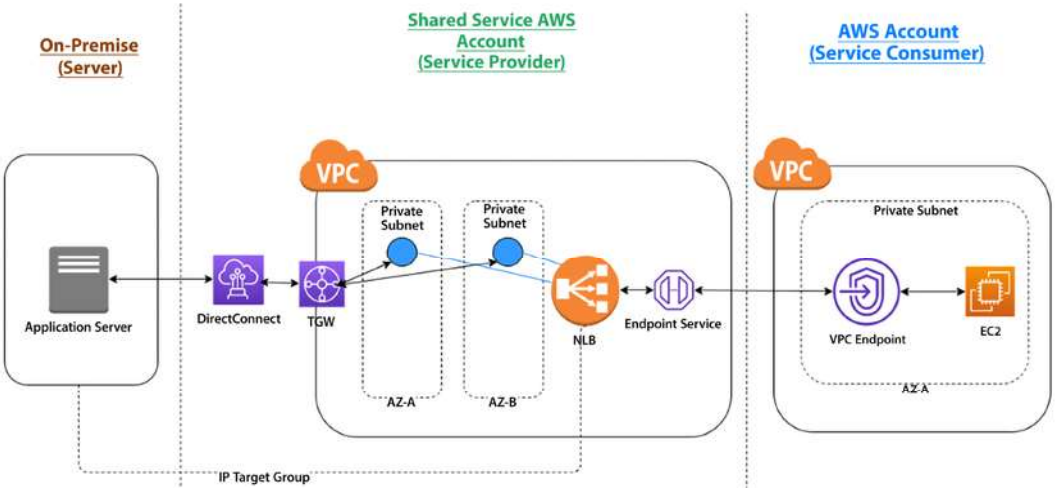

Figure 4.7: PrivateLink between a shared Service Provider, on-premise server, and a Service Consumer account

In this setup, the on-premise servers are the service providers. The **NLB** in the **Shared Service** account is configured with an auto-scaling group with targets referencing the IP addresses of the on-premise servers. The **NLB** is then exposed as an **Endpoint Service**. The **Service Consumer** account can consume this **Endpoint Service** by creating a **VPC Endpoint**. Here, **DirectConnect** provides a dedicated high-speed fiber optics line between the on-premises server and AWS Regions. You will learn more about DirectConnect in this chapter in the *Building Hybrid Cloud Connectivity in AWS* section.

Now, many applications run globally and target to harness users in every corner of the world to accelerate their business. In such a situation, it becomes essential that your users have the same experience while accessing your application regardless of their physical location. AWS provides various edge networking services to handle global traffic. Let's learn more about this.

Edge networking

Edge networking is like last-mile delivery in the supply chain world. When you have users across the world, from the USA to Australia and India to Brazil, you want each user to have the same experience regardless of the physical location of your server where the application is hosted. There are serval components that play their role in building last mile networking. Let's explore them in detail.

Route 53

Amazon Route 53 is a fully managed, simple, fast, secure, highly available, and scalable DNS service. It provides a reliable and cost-effective means for systems and users to translate names like `www.example.com` into IP addresses like `1.2.3.4`. Route 53 is a domain register where you can register a new domain. You can choose an available domain and add it to the cart from the AWS Console and define contacts for the domain. AWS allows you to transfer your domains to AWS and between accounts.

In Route 53, AWS assigns four name servers for all domains, as shown in the screenshot: one for **.com**, one for **.net**, one for **.co.uk**, and one for **.org**. Why? For higher availability! If there is an issue with the **.net** DNS services, the other three continue to provide high availability for your domains.

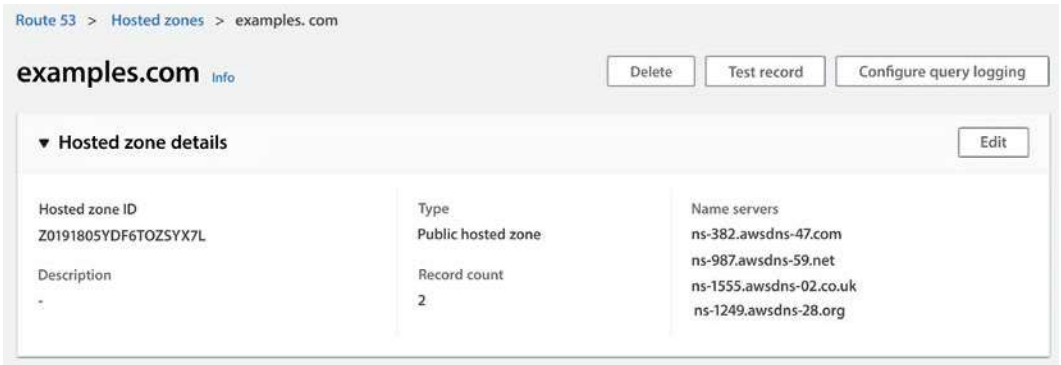

Figure 4.8: Route 53 name server configuration

Route 53 supports both public and private hosted zones. Public hosted zones have a route to internet-facing resources and resolve from the internet using global routing policies. Meanwhile, private hosted zones have a route to VPC resources and resolve from inside the VPC. It helps to integrate with on-premises private zones using forwarding rules and endpoints.

Route 53 provides IPV6 support with end-to-end DNS resolution and support for IPv6 forward (AAAA) and reverse (PTR) DNS records, along with health check monitoring for IPv6 endpoints. For PrivateLink support, when configuring, you can specify a private DNS name, and the Route 53 resolver will resolve it to the PrivateLink endpoint.

Route 53 provides the following seven types of routing policies for traffic:

- **Simple routing policy** – This is used for a single resource (for example, a web server created for the `www.example.com` website).
- **Failover routing policy** – This is used to configure active-passive failover.
- **Geolocation routing policy** – This routes traffic based on the user's location.
- **Geoproximity routing policy** – This is used for geolocation when users are shifting from one location to another.
- **Latency routing policy** – This optimizes the best latency for the resources deployed in multiple AWS Regions.
- **Multivalue answer routing policy** – This is used to respond to DNS queries with up to eight healthy, randomly selected records.
- **Weighted routing policy** – This is used to route traffic to multiple resource properties as defined by you (for example, you want to say 80% traffic to site A and 20% to site B).

You can build advanced routing policies by nesting these primary routing policies into traffic policies; the following diagram shows a nested policy architecture.

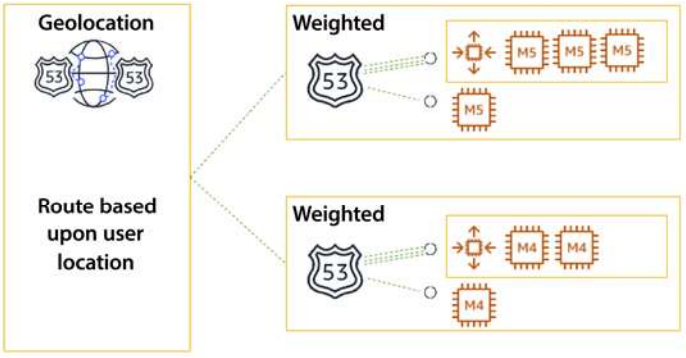

Figure 4.9: Route 53 nested routing policy

In the preceding diagram, you can see the policy is **Geolocation**-based, which routes traffic based on the user's location and its proximity to the nearest Region. In the second level, you have a nested policy defined as a **Weighted** policy within the region that routes traffic to servers based on the weight you have defined to route traffic to individual application servers. Advanced routing policies can be built by nesting the seven primary routing policies into traffic policies. You can find more details on this routing policy in the AWS user document here – `https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html`.

Route 53 Resolver rules tell Route 53 to query a domain. For DNS zones that should resolve on-premises, add a forward rule and point toward the appropriate outbound resolver. Public hosted zones route traffic to internet-facing resources and resolve from the internet using global routing policies. Private hosted zones route traffic to VPC resources and resolve from inside the VPC. Private hosted zones integrate with on-premises private zones using forwarding rules and endpoints.

Route 53 is the only service in which AWS offers 100% SLA, which means AWS makes its best effort to ensure it is 100% available. If Route 53 does not meet the availability commitment, you will be eligible to receive a Service Credit.

While Route 53 helps to direct global traffic to your server, there could be latency if you wanted to deliver significant static assets, such as images and videos, to users far from your servers' deployment Region. AWS provides CloudFront as a content distribution network to solve these latency problems. Let's learn more about it.

Amazon CloudFront

Amazon CloudFront is a content delivery service that accelerates the distribution of both static and dynamic content like image files, video files, and JavaScript, CSS, or HTML files, through a network of data centers spread across the globe. These data centers are referred to as **edge locations**. When you use CloudFront to distribute your content, users requesting content get served by the nearest edge location, providing lower latency and better performance.

As of 2022, AWS has over 300 high-density edge locations spread across over 90 cities in 47 countries. All edge locations are equipped with ample cache storage space and intelligent routing mechanisms to increase the edge cache hit ratio. AWS content distribution edge locations are connected with high-performance 100 GbE network devices and are fully redundant, with parallel global networks with default physical layer encryption.

Suppose you have an image distribution website, `www.example.com`, hosted in the US, which serves art images. Users can access the URL `www.example.com/art.png`, and the image is loaded. If your server is close to the user, then the image load time will be faster, but if users from other locations like Australia or South Africa want to access the same URL, the request has to cross multiple networks before delivering the content to the user's browser. The following diagram shows the HTTP request flow with Amazon CloudFront.

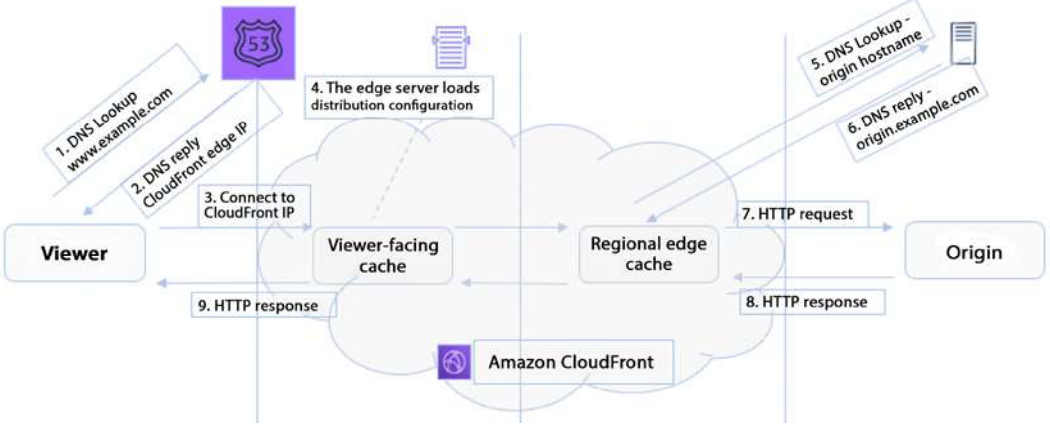

Figure 4.10: HTTP request flow with Amazon CloudFront

As shown in the preceding diagram, when a viewer requests access to page content from the origin server—in this case, `www.example.com`—Route 53 replies with the CloudFront edge IP and redirects the user to the CloudFront location. CloudFront uses the following rules for content distribution:

- If the requested content is already in the edge data center, which means it is a "cache hit," it will be served immediately.
- If the content is not at the edge location (a "cache miss"), CloudFront will request the content from the original location (the web server or S3). The request flows through the AWS backbone, is delivered to the customer, and a copy is kept for future requests.
- If you are using CloudFront, it also provides an extra layer of security since your origin server is not directly exposed to the public network.

CloudFront eliminates the need to go to the origin server for user requests, and content is served from the nearest location. CloudFront provides security by safeguarding the connection between end-users and the content edge, as well as between the edge network and the origin. By offloading SSL termination to CloudFront, the performance of applications is improved since the burden of processing the required negotiation and SSL handshakes is removed from the origins.

CloudFront is a vast topic, and you can learn more about it here – `https://aws.amazon.com/cloudfront/features/`.

In this section, you learned how CloudFront improves performance for both cacheable content and a wide range of applications over TCP, UDP, and MQTT. To address this traffic, AWS provides **AWS Global Accelerator (AGA)**, which improves the availability and performance of your applications with local or global users. Let's learn more about it.

AWS Global Accelerator (AGA)

AGA enhances application availability and performance by offering fixed static IP addresses as a single entry points, or multiple entry points, to AWS Regions, including ALBs, NLBs, and EC2 instances. AGA utilizes the AWS global network to optimize the path from users to applications, thereby improving the performance of TCP and UDP traffic. AGA continuously monitors the health of application endpoints and promptly redirects traffic to healthy endpoints within 1 minute, in the event of an unhealthy endpoint detection.

AGA and CloudFront are distinct services offered by AWS that employ the AWS global network and its edge locations. While CloudFront accelerates the performance of both cacheable (e.g., videos and images) and dynamic (e.g., dynamic site delivery and API acceleration) content, AGA enhances the performance of various applications over TCP or UDP. Both services are compatible with AWS Shield, providing protection against DDoS attacks. You will learn more about AWS Shield in *Chapter 8*, *Best Practices for Application Security, Identity, and Compliance*.

AGA automatically reroutes your traffic to the nearest healthy endpoint to avoid failure. AGA health checks will react to customer backend failure within 30 seconds, which is in line with other AWS load-balancing solutions (such as NLB) and Route 53. Where AGA raises the bar is with its ability to shift traffic to healthy backends in as short a timeframe as 30 seconds, whereas DNS-based solutions can take minutes to hours to shift the traffic load. Some key reasons to use AGA are:

- **Accelerate your global applications** – AGA intelligently directs TCP or UDP traffic from users to the AWS-based application endpoint, providing consistent performance regardless of their geographic location.
- **Improve global application availability** – AGA constantly monitors your application endpoints, including but not limited to ALBs, NLBs, and EC2 instances. It instantly reacts to changes in their health or configuration, redirecting traffic to the next closest available endpoint when problems arise. As a result, your users experience higher availability. AGA delivers inter-Region load balancing, while ELB provides intra-Region load balancing.

ELB in a Region is a suitable candidate for AGA as it evenly distributes incoming application traffic across backends, such as Amazon EC2 instances or ECS tasks, within the Region. AGA complements ELB by expanding these capabilities beyond any single Region, enabling you to create a global interface for applications with application stacks located in a single Region or multiple Regions.

- **Fixed entry point** – AGA provides a set of static IP addresses for use as a fixed entry point to your AWS application. Announced via anycast and delivered from AWS edge locations worldwide, these eliminate the complexity of managing the IP addresses of multiple endpoints and allow you to scale your application and maintain DDoS resiliency with AWS Shield.
- **Protect your applications** – AGA allows you to serve internet users while keeping your ALBs and EC2 instances private.

AGA allows customers to run global applications in multiple AWS Regions. Traffic destined to static IPs is globally distributed, and end user requests are ingested through AWS's closest edge location and routed to the correct regional resource for better availability and latency. This global endpoint supports TCP and UDP and does not change even as customers move resources between Regions for failover or other reasons (i.e., client applications are no longer tightly coupled to the specific AWS Region an application runs in). Customers will like the simplicity of this managed service.

As technology becomes more accessible with the high-speed networks provided by 5G, there is a need to run applications such as connected cars, autonomous vehicles, and live video recognition with ultra-low latency. AWS provides a service called AWS Wavelength, which delivers AWS services to the edge of the 5G network. Let's learn more about it.

AWS Wavelength

AWS Wavelength is designed to reduce network latency when connecting to applications from 5G-connected devices by providing infrastructure deployments within the Telco 5G network service providers' data centers. It allows application traffic to reach application servers running in Wavelength Zones, as well as AWS compute and storage services. This eliminates the need for traffic to go through the internet, which can introduce latency of up to 10s of milliseconds and limit the full potential of the bandwidth and latency advancements of 5G.

AWS Wavelength allows for the creation and implementation of real-time, low-latency applications, such as edge inference, smart factories, IoT devices, and live streaming. This service enables the deployment of emerging, interactive applications that require ultra-low latency to function effectively.

Some key benefits of AWS Wavelength are:

- **Ultra-low latency for 5G** – Wavelength combines the AWS core services, such as compute and storage, with low-latency 5G networks. It helps you to build applications with ultra-low latencies using the 5G network.
- **Consistent AWS experience** – You can use the same AWS services you use daily on the AWS platform.
- **Global 5G network** – Wavelength is available in popular Telco networks such as Verizon, Vodafone, and SK Telecom across the globe, including the US, Europe, Korea, and Japan, which enables ultra-low latency applications for a global user base.

Wavelength Zones are connected to a Region and provide access to AWS services. Architecting edge applications using a hub-and-spoke model with the Region is recommended for scalable and cost-effective options for less latency-sensitive applications.

As enterprises adopt the cloud, it will not be possible to instantly move all IT workloads to the cloud. Some applications need to run on-premises and still communicate with the cloud. Let's learn about AWS services for setting up a hybrid cloud.

Building hybrid cloud connectivity in AWS

A hybrid cloud comes into the picture when you need to keep some of your IT workload on-premises while creating your cloud migration strategy. You may have decided to keep them out of the cloud for various reasons, such as compliance and the unavailability of out-of-the-box cloud services such as a mainframe, when you need more time to re-architect them, or when you are waiting to complete your license term with an existing vendor. In such cases, you need highly reliable connectivity between your on-premises and cloud infrastructure. Let's learn about the various options available from AWS to set up hybrid cloud connectivity.

AWS Virtual Private Network (VPN)

AWS VPN is a networking service to establish a secure connection between AWS, on-premises networks, and remote client devices. There are two variants of AWS VPN: Site-to-Site VPN and AWS Client VPN. AWS Site-To-Site VPN establishes a secure tunnel between on-premises and AWS Virtual Private Gateways or AWS TGWs.

It offers fully managed and highly available VPN termination endpoints at AWS Regions. You can add two VPN tunnels per VPN connection, which is secured with an IPsec Site-to-Site tunnel with AES-256, SHA-2, and the latest DH groups. The following diagram shows the AWS Site-to-Site VPN connection.

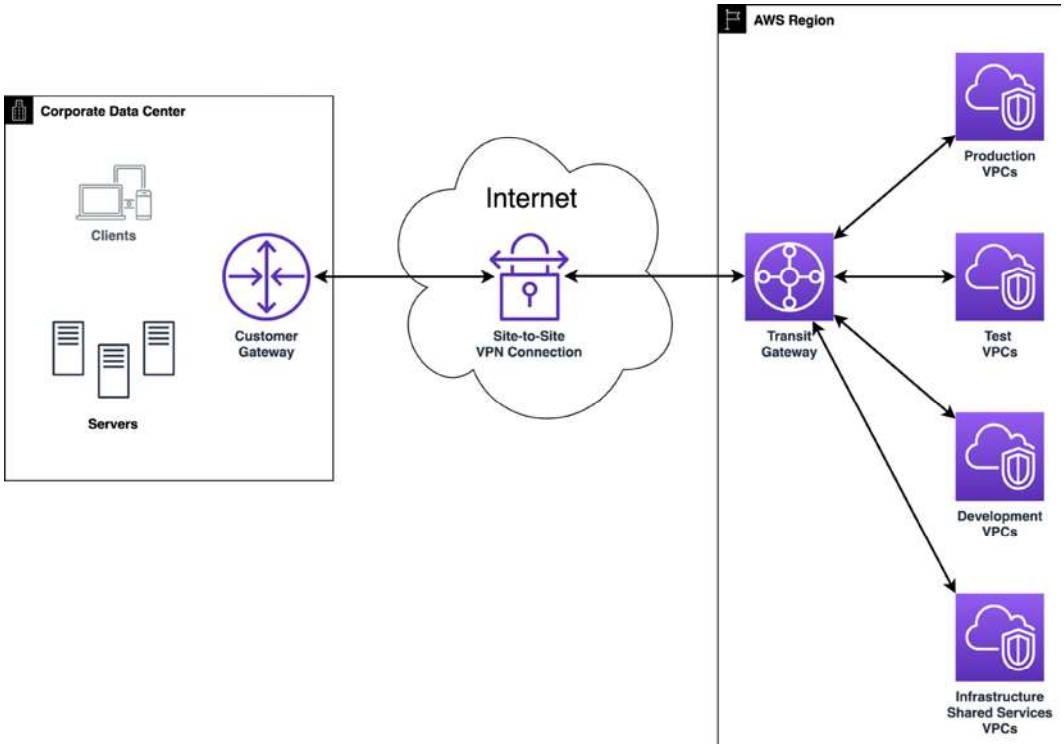

Figure 4.11: AWS Site-to-Site VPN with TGW

As depicted in the preceding diagram, a connection has been established from the customer gateway to the AWS TGW using Site-To-Site VPN, which is further connected to multiple VPCs.

AWS Client VPN can be used with OpenVPN-based VPN client software to access your AWS resources and on-premises resources from any location across the globe.

AWS Client VPN also supports a split tunneling feature, which can be used if you only want to send traffic destined to AWS via Client VPN and the rest of the traffic via a local internet breakout.

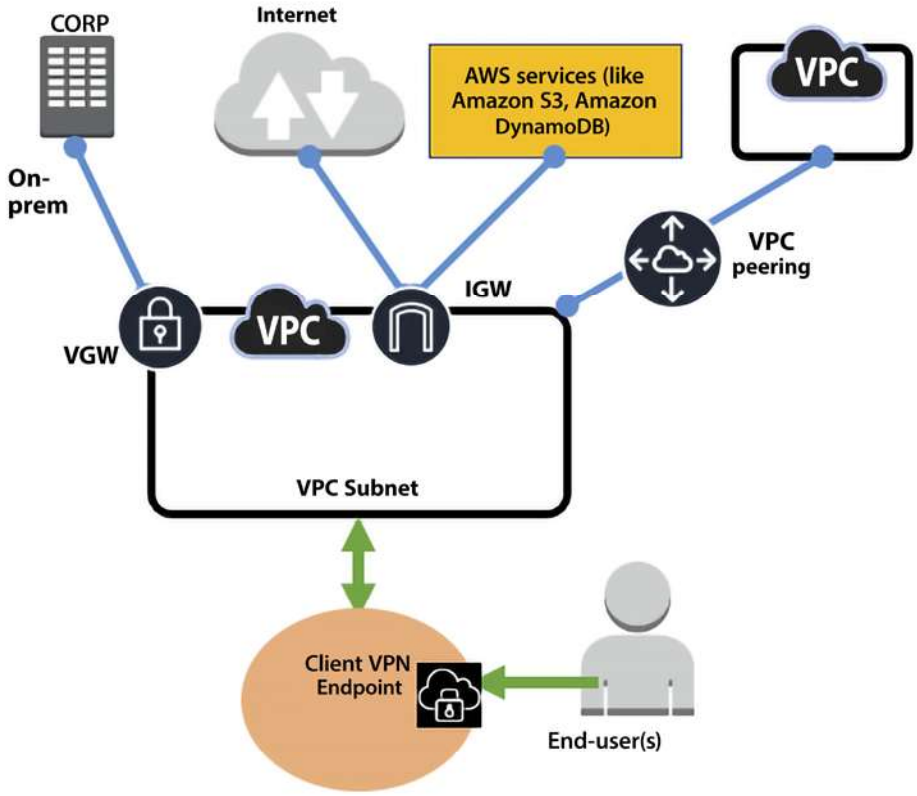

Figure 4.12: AWS Client VPN

AWS Client VPN provides secure access to any resource in AWS and on-premises from anywhere using OpenVPN clients. It seamlessly integrates with existing infrastructure, like Amazon VPC, AWS Directory Service, and so on.

AWS Direct Connect

AWS Direct Connect is a low-level infrastructure service that enables AWS customers to set up a dedicated network connection between their on-premises facilities and AWS. You can bypass any public internet connection using AWS Direct Connect and establish a private connection linking your data centers with AWS. This solution provides higher network throughput, increases the consistency of connections, and, counterintuitively, can often reduce network costs.

There are two variants of Direct Connect, dedicated and hosted. A dedicated connection is made through a 1 Gbps, 10 Gbps, or 100 Gbps dedicated Ethernet connection for a single customer. Hosted connections are obtained via an AWS Direct Connect Delivery Partner, who provides the connectivity between your data center and AWS via the partner's infrastructure.

However, it is essential to note that AWS Direct Connect does not provide encryption in transit by default. Suppose you want to have encryption in transit. In that case, you have two choices – you can either use AWS Site-To-Site VPN to provide IPsec encryption for your packets, or you can combine AWS Direct Connect with AWS Site-to-Site VPN to deliver an IPsec-encrypted private connection while, at the same time, lowering network costs and increasing network bandwidth throughput.

The other option is to activate the MACsec feature, which provides line-rate, bi-directional encryption for 10 Gpbs and 100 Gpbs dedicated connections between your data centers and AWS Direct Connect locations. MACsec is done at the hardware; hence, it provides better performance. To encrypt the traffic, you can also use an AWS technology partner as an alternative solution to encrypt this network traffic.

AWS Direct Connect uses the 802.1q industry standard to create VLANs. These connections can be split into several **virtual interfaces (VIFs)**. This enables us to leverage the same connection to reach publicly accessible services such as Amazon S3 by using an IP address space and private services such as EC2 instances running in a VPC within AWS. The following are AWS Direct Connect interface types:

- **Public virtual interface:** This is the interface you can use to access any AWS public services globally, which are accessible via public IP addresses such as Amazon S3. A public VIF can access all AWS public services using public IP addresses.
- **Private virtual interface**: You can connect to your VPCs using private VIFs using private IP addresses. You can connect to the AWS Direct Connect gateway using a private VIF, allowing you to connect to up to 10 VPCs globally with a single VIF, unlike connecting a private VIF to a Virtual Private Gateway associated with a single VPC.
- **Transit virtual interface**: A transit VIF is connected to your TGW via a Direct Connect gateway. A transit VIF is supported for a bandwidth of 1 Gbps or higher.

The following diagram has put together various AWS Direct Connect interfaces, showing the use of various interfaces while designing your hybrid cloud connectivity.

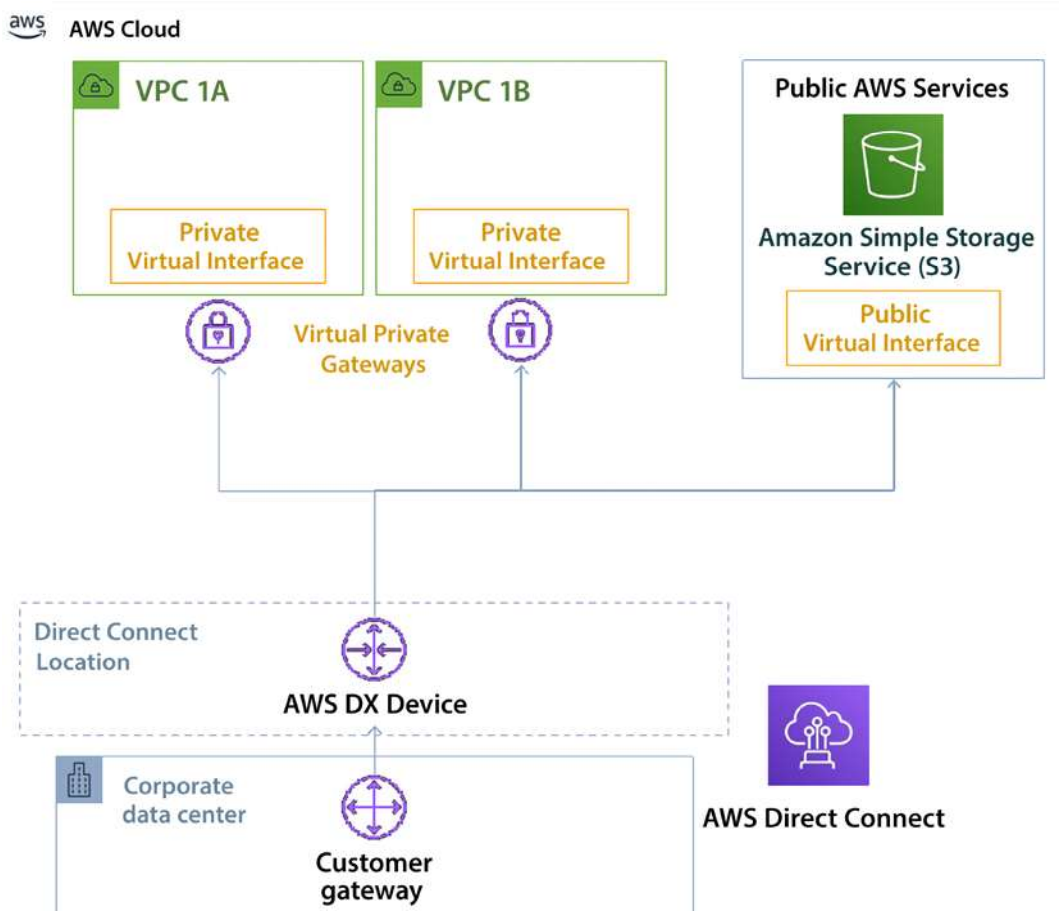

Figure 4.13: AWS Direct Connect interface types

The preceding diagram shows the corporate data center connected to the AWS cloud using the Direct Connect location. Most of your application workloads, such as the web server, app server, and database server, run inside the VPC under a private restricted network, so a private VIF connects to VPCs across different AZs. Conversely, Amazon S3 is in the public domain, where you might host static pages, images, and videos for your applications connected through a public VIF.

AWS Direct Connect can reduce costs when workloads require high bandwidth. It can reduce these costs in two ways:

- It transfers data from on-premises environments to the cloud, directly reducing cost commitments to **Internet Service Providers (ISPs)**.
- The costs of transferring the data using a dedicated connection are billed using the AWS Direct Connect data transfer rates and not the internet data transfer rates, which are lower.

Network latency and responses to requests can be highly variable. Workloads that use AWS Direct Connect have a much more homogenous latency and consistent user experience. Direct connection comes with a cost; you may only sometimes need such high bandwidth and want to optimize costs better. For such cases, you may want to use AWS VPN, as discussed in the *AWS Virtual Private Network (VPN)* section.

You have learned about different patterns of setting up network connectivity within an AWS cloud and to or from an AWS cloud, but for a large enterprise with branch offices or chain stores, connecting multiple data centers, office locations, and cloud resources can be a very tedious task. AWS provides Cloud WAN to simplify this issue. Let's learn more about it.

AWS Cloud WAN

How many of you would be paged if your entire network suddenly went down? Let's take an example, Petco, which has over 1,500 locations. Imagine what happens on a network like that on any given day. One way to connect your data centers and branch offices is to use fixed, physical network connections. These connections are long-lived and not easy to change quickly.

Many use AWS Site-to-Site VPN connections for connectivity between their locations and AWS. Alternatively, you bypass the internet altogether and use AWS Direct Connect to create a dedicated network link to AWS. And some use broadband internet with SD-WAN hardware to create virtual overlay networks between locations. Inside AWS, you build networks within VPCs and route traffic between them with TGW. The problem is that these networks all take different approaches to connectivity, security, and monitoring. As a result, you are faced with a patchwork of tools and networks to manage and maintain.

For example, to keep your network secure, you must configure firewalls at every location, but you are faced with many different firewalls from many different vendors, and each is configured slightly differently than the others. Ensuring your access policies are synced across the entire network quickly becomes daunting. Likewise, managing and troubleshooting your network is difficult when the information you need is kept in many different systems.

Every new location, network appliance, and security requirement makes things more and more complicated. We see many customers struggle to keep up. To solve these problems, network architects need a way to unify their networks so that there is one central place to build, manage, and secure their network. They need easy ways to make and change connections between their data centers, branch offices, and cloud applications, regardless of what they're running on today. And they need a backbone network that can smoothly adapt to these changes.

AWS Cloud WAN is a global network service enabling you to create and manage your infrastructure globally in any AWS Region or on-premises. Consider it a global router that you can use to connect your on-premises networks and AWS infrastructure. AWS Cloud WAN provides network segmentation to group your network or workloads, which can be located across any AWS Region. It takes care of the route propagation to connect your infrastructure without manually maintaining the routing.

AWS Cloud WAN provides network segmentation, which means you can divide your global network into separate and isolated networks. This helps you to control the traffic flow and cross-network access tightly. For example, a corporate firm can have a network segment for invoicing and order processing, another for web traffic, and another for logging and monitoring traffic.

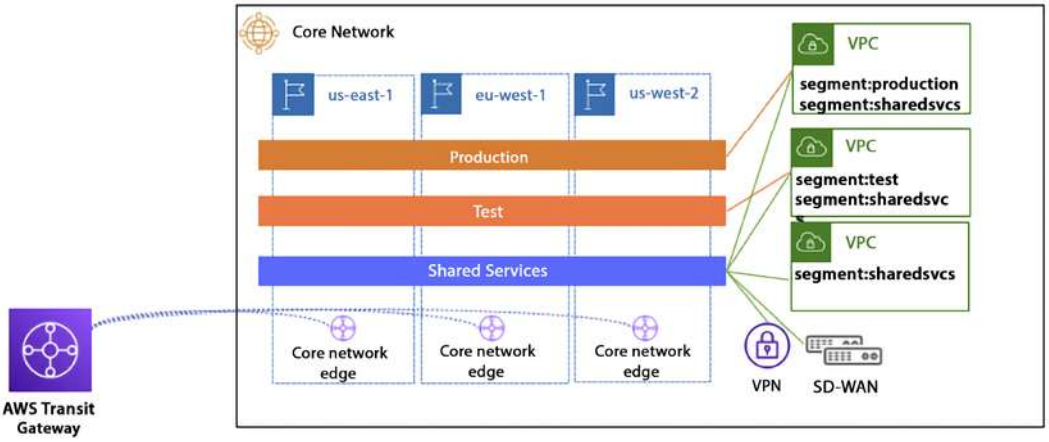

Figure 4.14: AWS Cloud WAN architecture

When using Cloud WAN, you can see your entire network on one dashboard, giving you one place to monitor and track metrics for your entire network. Cloud WAN lets you spot problems early and respond quickly, which minimizes downtime and bottlenecks while helping you troubleshoot problems, even when data comes from separate systems. Security is the top priority, and throughout this chapter, you have learned about the security aspect of network design.

Let's go into more detail to learn about network security best practices.

AWS cloud network security

Security is always a top priority for any organization. As a general rule, you need to protect every infrastructure element individually and as a group – like the saying, "Dance as if nobody is watching and secure as if everybody is." AWS provides various managed security services and a well-architected pillar to help you design a secure solution.

You can implement secure network infrastructure by creating an allow-list of permitted protocols, ports, CIDR networks, and SG sources, and enforcing several policies on the AWS cloud. A NACL is used to explicitly block malicious traffic and segment the edge connectivity to the internet, as shown in the following diagram.

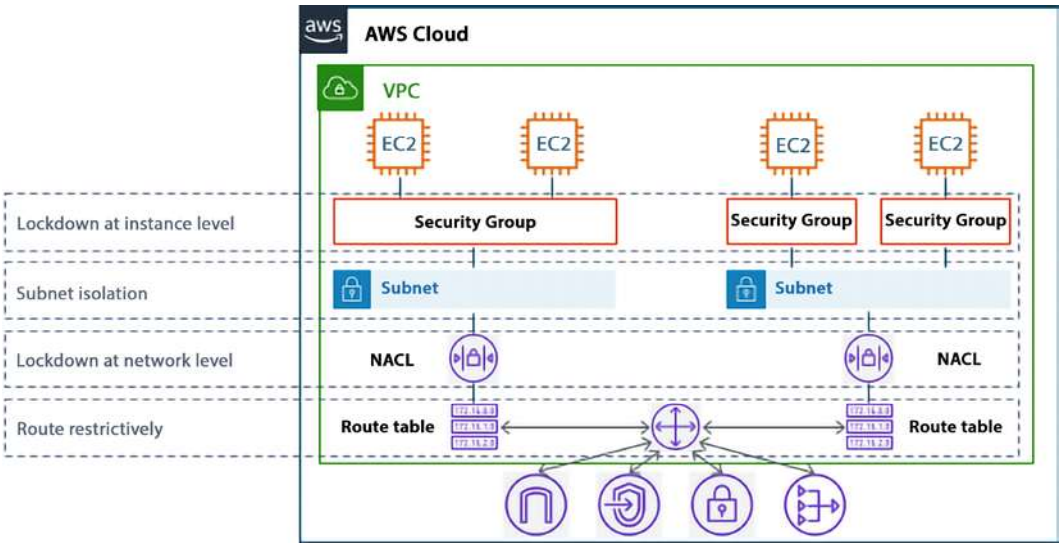

Figure 4.15: SGs and NACLs in network defense

As shown in the preceding diagram, SGs are the host-level virtual firewalls that protect your EC2 instance from the traffic coming to or leaving from a particular instance. An SG is a stateful firewall, which means that you need to create a rule in one direction only, and the return traffic will automatically be allowed. A NACL is also a virtual firewall that sits at the subnet boundary, regulating traffic in and out of one or more subnets. Unlike an SG, a NACL is stateless, which means you need both inbound and outbound rules to allow specific traffic.

The following table outlines the difference between SGs and NACLs.

SG	NACL
The SG is the first layer of defense, which operates at the instance level.	A NACL works at the subnet level inside an AWS VPC.
You can only add "allow" security rules.	You can explicitly add a "deny" rule in addition to allow rules. For example, you can deny access to a specific IP address.
The SG is stateful, which means once you add an inbound allow rule, it automatically adds an outbound allow rule. For example, for a CRM server to respond, you have only to allow a rule to accept traffic from the IP range.	A NACL is stateless, meaning if you add an inbound allow rule, you must add an explicit allow rule. For a CRM server to respond, you must add both allow and deny rules to accept traffic from the IP range.
If you have added multiple rules in the instance, the SG will validate all rules before deciding whether to allow traffic.	In a NACL, you define rule priority by assigning values, such as 100 or 200. The NACL processes the rules in numerical order.
As an SG is at the instance level, it applies to an instance only.	As a NACL is at the subnet level, it automatically applies to all instances in the subnets it's linked to.

Table 4.1: Comparison between security groups and network access control lists

While the SG and NACL provide security inside the VPC, let's learn more about overall AWS network security best practices.

AWS Network Firewall (ANFW)

ANFW is a highly available, fully redundant, and easy-to-deploy managed network firewall service for your Amazon VPC, which offers a service-level agreement with an uptime commitment of 99.99%. ANFW scales automatically based on your network traffic, eliminating the need for the capacity planning, deployment, and management of the firewall infrastructure.

ANFW supports open source Suricata-compatible rules for stateful inspection. It provides fine controls for your network traffic, such as allowing or blocking specific protocol traffic from specific prefixes. ANFW also supports third-party integration to source-managed intelligent feeds.

ANFW also provides alert logs that detail a particular rule that has been triggered. There is native integration with Amazon S3, Amazon Kinesis, and Amazon CloudWatch, which can act as the destination for these logs.

You can learn about various ANFW deployment models by referring to the detailed AWS blog here – `https://aws.amazon.com/blogs/networking-and-content-delivery/deployment-models-for-aws-network-firewall/`.

Let's learn more about network security patterns and anti-patterns, keeping SGs and NACL in mind.

AWS network security patterns – best practices

Several patterns can be used when creating an SG and NACL strategy for your organization, such as:

- **Create the SG before launching the instance(s), resource(s), or cluster(s)** – This will force you to determine if a new SG is necessary or if an existing SG should be used. This enables you to pre-define all the rules and reference the SG at creation time, especially when creating these programmatically or via a CloudFormation/Landing Zone pipeline. Additionally, you should not use the default SG for your applications.
- **Logically construct SGs into functional categories based on their application tier or role they perform** – Consider the number of distinct tiers your application has and then logically construct the SGs to match those functional components. For a typical three-tier architecture, a minimum of three SGs should be used (e.g., a web tier, an app tier, and a DB tier).
- **Configure rules to chain SGs to each other** – In an SG rule, you can authorize network access from a specific CIDR address range or another SG in your VPC. Either option could be appropriate, depending on your environment. Generally speaking, organizations can "chain" the SGs together between application tiers, thus building a logical flow of allowed traffic from one SG to another. However, an exception to this pattern is described in the following pattern.
- **Restrict privileged administrative ports (e.g., SSH or RDP) to internal systems (e.g., bastion hosts)** – As a best practice, administrative access to instances should be blocked or restricted to a small number of protected and monitored instances – sometimes referred to as bastion hosts or jump boxes.
- **Create NACL rule numbers with the future in mind** – NACLs rules are evaluated in numerical order based on the rule number, and the first rule that matches the traffic will be used. For example, if there are 10 rules in an NACL and rule number 2 matches `DENY SSH traffic` on port 22, the other 8 rules never get evaluated, regardless of their content. Therefore, when creating NACL rules, it is best practice to leave gaps between the rule numbers – 100 is typically used. So, the first rule has a rule number of 100, and the second rule has a rule number of 200.

If you have to add a rule between these rules in the future, you can add a new rule with rule number 150, which still leaves space for future planning.

- **In well-architected, high-availability VPCs, share NACLs based on subnet tiers –** The subnet tiers in a well-architected, high-availability VPC should have the same resources and applications deployed in the same subnet tiers (e.g., the web and app tiers). These tiers should have the same inbound and outbound rule requirements. In this case, it is recommended to use the same NACL to avoid making administrative changes in multiple places.
- **Limit inbound rules to the minimum required ports and restrict access to commonly vulnerable ports –** As with hardware firewalls, it is important to carefully determine the minimum baseline for inbound rules required for an application tier to function. Reducing the ports allowed greatly reduces the overall attack surface and simplifies the management of the NACLs.
- **Finally, audit and eliminate unnecessary, unused, or redundant NACL rules and SGs –** As your environment scales, you may find that unnecessary SGs and NACL rules were created or mistakenly left behind from previous changes.

AWS network security anti-patterns

While the previous section described patterns for using VPC SGs and NACLs, there are a number of ways you might attempt to configure or utilize your SGs and NACL in non-recommended ways. These are referred to as "anti-patterns" and should be avoided.

- The default SG is included automatically with your VPC. Whenever you launch an EC2 instance or any other AWS resource in your VPC, it is linked to the default SG. Using the default SG does not provide granular control. You can create custom SGs and add them to instances or resources. You can create multiple SGs as per your applications' needs, such as a web server EC2 instance or an Aurora database server.
- If you already have SGs applied to all instances, **do not** create a rule that references these in other SGs. By referencing an SG applied to all instances, you are defining a source or destination of all instances. This is a wide-open pointer to or from everything in your environment. There are better ways to apply least-privileged access.
- Multiple SGs can be applied to an instance. Within a specific application tier, all instances should have the same set of SGs applied to them. However, ensure a tier's instances have a uniform consistency. Suppose web servers, application servers, and database servers co-mingle in the same tier. In this case, they should be separated into distinct tiers and have tier-specific SGs applied to them. **Do not** create unique SGs for related instances (one-off configurations or permutations).

- Refrain from sharing or reusing NACLs in subnets with different resources – although the NACL rules may be the same now, there is no way to determine future rules required for the resources. Since the resources in the subnet differ (e.g., between different applications), resources in one subnet may need a new rule that isn't needed for the resources in the other subnet. However, since the NACL is shared, opening up the rule applies resources in both subnets. This approach does not follow the principle of least privilege.
- NACLs are stateless, so rules are evaluated when traffic enters and leaves the subnet. You will need an inbound and outbound rule for each two-way communication. This anti-pattern shouldn't be encountered because you are only using NACLs as guardrails. Evaluating large complex rule sets on traffic coming in and out of the subnet will eventually lead to performance degradation.

It is recommended to periodically audit for SGs and NACLs rules that are unnecessary or redundant and delete them. This will reduce complexity and help prevent reaching the service limit accidentally.

AWS network security with third-party solutions

You may not always want to get into all the nitty-gritty details of AWS security configuration and look for more managed solutions. AWS's extensive partner network builds managed AWS solutions to fulfill your network security needs. Some of the most popular network security solutions provided by the following **integrated software vendor (ISV)** partners in AWS Marketplace are below:

- **Palo Alto Networks** – Palo Alto Networks has introduced a Next-Generation Firewall service that simplifies securing AWS deployments. This service enables developers and cloud security architects to incorporate inline threat and data loss prevention into their application development workflows.
- **Aviatrix** – The Aviatrix Secure Networking Platform is made up of two components: the Aviatrix controller (which manages the gateways and orchestrates all connectivity) and the Aviatrix Gateways that are deployed in VPCs using AWS IAM roles.
- **Check Point** – Check Point CloudGuard Network Security is a comprehensive security solution designed to protect your AWS cloud environment and assets. It provides advanced, multi-layered network security features such as a firewall, IPS, application control, IPsec VPN, antivirus, and anti-bot functionality.
- **Fortinet** – This provides firewall technology to deliver complete content and network protection, including application control, IPS, VPN, and web filtering. It also provides more advanced features like vulnerability management and flow-based inspection work.

- **Cohesive Networks** – Cohesive's VNS3 is a software-only virtual appliance for connectivity, federation, and security in AWS.

In addition to these, many more AWS-managed solutions are available through partners like Netskope, Valtix, IBM, and Cisco. You can find the complete list of network security solutions available in the AWS Marketplace using this link – `https://aws.amazon.com/marketplace/search/results?searchTerms=network+security`.

AWS cloud security has multiple components, starting with networking as a top job. You can learn more about AWS security by visiting their security page at `https://aws.amazon.com/security/`.

Summary

In this chapter, you started with learning about the AWS Global Infrastructure and understanding the details of AWS Regions, AZs, and LZs. You also learned about the various benefits of using the AWS Global Infrastructure.

Networking is the backbone of any IT workload, whether in the cloud or in an on-premises network. To start your cloud journey in AWS, you must have good knowledge of AWS networking. When you start with AWS, you create your VPC within AWS. You learned about using an AWS VPC with various components such as an SG, a NACL, a route table, an IGW, and a NAT gateway. You learned how to segregate and secure your IT resources by putting them into private and public subnets.

With the ease of creating VPC in AWS organizations, multiple VPCs tend to be created, whether it is intentional to give each team their own VPC, or unintentional when the dev team creates multiple test workloads. Often, these VPCs need to communicate with each other; for example, the finance department needs to get information from accounting. You learned about setting up communication between multiple VPCs using VPC peering and TGW. You learned to establish secure connections with services on the public internet or other accounts using AWS PrivateLink.

Further, you learned about AWS edge networking to address the global nature of user traffic. These services include Route 53, CloudFront, AGA, and AWS Wavelength. You then learned about connecting an on-premises server and an AWS cloud. Finally, you closed the chapter with the network security best practices, patterns and anti-patterns, and third-party managed network security solutions available via the AWS partner network.

As you start your journey of learning about the AWS core services, the next topic is storage. AWS provides various types of storage to match your workload needs. In the next chapter, you will learn about storage in AWS and how to choose the right storage for your IT workload needs.

7

Selecting the Right Database Service

Building applications is all about data collection and management. If you design an e-commerce application, you want to show available inventory catalog data to customers and collect purchase data as they make a transaction. Similarly, if you are running an autonomous vehicle application, you want to analyze data on the surrounding traffic and provide the right prediction to cars based on that data. As of now, you have learned about networking, storage, and compute in previous chapters. In this chapter, you will learn the choices of database services available in AWS to complete the core architecture tech stack.

With so many choices at your disposal, it is easy to get analysis paralysis. So, in this chapter, we will first lay a foundation of how the databases and their use cases can be classified and then use these classifications to help us pick the right service for our particular use case and our circumstances. In this chapter, you will navigate the variety of options, which will give you the confidence that you are using the right tool for the job.

In this chapter, you will learn the following topics:

- A brief history of databases and data-driven innovation trends
- Database consistency model
- Database usages model
- AWS relational and non-relational database services
- Benefits of AWS database services

- Choosing the right tool for the job
- Migrating databases to AWS

By the end of this chapter, you will learn about different AWS database service offerings and how to choose a suitable database for your workload.

A brief history of databases

Relational databases have been around for over 50 years. Edgar F. Codd created the first database in 1970. The main feature of a relational database is that data is arranged in rows and columns, and rows in tables are associated with other rows in other tables by using the column values in each row as relationship keys. Another important feature of relational databases is that they normally use **Structured Query Language** (**SQL**) to access, insert, update, and delete records. SQL was created by IBM researchers Raymond Boyce and Donald Chamberlin in the 1970s. Relational databases and SQL have served us well for decades.

As the internet's popularity increased in the 1990s, we started hitting scalability limits with relational databases. Additionally, a wider variety of data types started cropping up. **Relational Database Management Systems** (**RDBMSs**) were simply not enough anymore. This led to the development of new designs, and we got the term **NoSQL databases**. As confusing as the term is, it does convey the idea that it can deal with data that is not structured, and it deals with it with more flexibility.

The term **NoSQL** is attributed to Carlo Strozzi and was first used in 1998 for a relational database that he developed but that didn't use the SQL language. The term was then again used in 2009 by Eric Evans and Johan Oskarsson to describe databases that were not relational.

The main difference between relational and non-relational databases is the way they store data and query it. Let's see an example of making a choice between a relational and non-relational database. Take an example of a banking transaction; it is critical for financial transactions in every customer's bank account to always be consistent and roll back in case of any error. In such a scenario, you want to use a relational database. For a relational database, if some information is not available, then you are forced to store null or some other value. Now take an example of a social media profile, which may have hundreds of attributes to store the user's name, address, education, jobs, personal choices, preferences, etc. However, many users do not fill in all the information; some users may add just their name, while others add more details like their address and education. In such cases, you want to use a non-relational database and store only the information provided by the user without adding null values where the user doesn't provide details (unlike in relational databases).

The following tables demonstrate the difference between relational and non-relational databases that have the same user data.

First Name	**Last Name**	**City**	**Country**
Maverick	Doe	Seattle	USA
Goose	Henske	NULL	NULL
John	Gayle	London	NULL

Table 7.1: Relational database

First Name	**Last Name**	**City**	**Country**
Maverick	Doe	Seattle	USA
Goose	Henske		
John	Gayle	London	

Table 7.2: Non-relational database

In the tables above, 3 users provided their information with their first name, last name, city, and country. You can see that only Maverick provided their full information while the other users left out either their city, country, or both. In a relational database, you need to fill in missing information with a `NULL` value across all columns, while in a non-relational database, that column doesn't exist at all.

It is nothing short of amazing what has occurred since then. Hundreds of new offerings have been developed, each trying to solve a different problem. In this environment, deciding the best service or product to solve your problem becomes complicated. And you must consider not only your current requirements and workloads, but also take into account that your choice of database will be able to cover your future requirements and new demands. With so much data getting generated, it is natural that much of innovation is driven by data. Let's look in detail at how data is driving innovation.

Data-driven innovation trends

Since high-speed internet became available in the last decade, more and more data is getting generated. Before we proceed, let's discuss three significant trends that influence your perspective on data:

- **The surge of data**: Our current era is witnessing an enormous surge in data generation. Managing the vast amount of data originating from your business applications is essential.

However, the exponential growth primarily stems from the data produced by network-connected intelligent devices, amplifying the data's diversity and quantity. These "smart" devices, including but not limited to mobile phones, connected vehicles, smart homes, wearable technologies, household appliances, security systems, industrial equipment, machinery, and electronic gadgets, constantly generate real-time data. Notably, over one-third of mobile sign-ups on cellular networks result from built-in cellular connections in most modern cars. In addition, applications generate real-time data, such as purchase data from e-commerce sites, user behavior from mobile apps, and social media posts/tweets. The data volume is expanding tenfold every five years, necessitating cloud-based solutions to manage and exploit vast data efficiently.

- **Microservices change analytics requirements**: The advent of microservices is revolutionizing organizations' data and analytics requirements. Rather than developing monolithic applications, companies are shifting towards a microservices architecture that divides complex problems into independent units. This approach enables developers to operate in smaller groups with minimal coordination, respond more efficiently, and work faster. Microservices enable developers to break down their applications into smaller parts, providing them with the flexibility to use multiple databases for various workloads, each suited for its specific purpose. The importance of analytics cannot be overstated, and it must be incorporated into every aspect of the business, rather than just being an after-the-fact activity. Monitoring the organization's operations in real time is critical to fuel innovation and quick decision-making, whether through human intervention or automated processes. Today's well-run businesses thrive on the swift utilization of data.
- **DevOps driving fast changes**: The fast-paced rate of change, driven by DevOps, is transforming how businesses approach IT. To keep up with the rapid innovation and the velocity of IT changes, organizations are adopting the DevOps model. This approach employs automated development tools to facilitate continuous software development, deployment, and enhancement. DevOps emphasizes effective communication, collaboration, and integration between software developers and IT operations. It also involves a rapid rate of change and change management, enabling businesses to adapt to evolving market needs and stay ahead of the competition.

While you see the trend that the industry is adopting, let's learn some basics of databases and learn about the database consistency model in more detail.

Database consistency model

In the context of databases, ensuring transaction data consistency involves restricting any database transaction's ability to modify data in unauthorized ways. When data is written to the database, it must adhere to a set of predefined rules and constraints. These rules are verified, and all checks must be successfully passed before the data can be accessed by other users. This stringent process ensures that data integrity is maintained and that the information stored in the database is accurate and trustworthy. Currently, there are two popular data consistency models. We'll discuss these models in the following subsections.

ACID data consistency model

When database sizes were measured in megabytes, we could have stringent requirements that enforced strict consistency. Since storage has become exponentially cheaper, databases can be much bigger, often measured in terabytes and even petabytes. For this reason, making databases ACID-compliant for storage reasons is much less prevalent. The ACID model guarantees the following:

- **Atomicity**: For an operation to be considered atomic, it should ensure that transactions within the operation either succeed or fail. If one of the transactions fails, all operations should fail and be rolled back. Could you imagine what would happen if you went to the ATM and the machine gave you money but didn't deduct it from your account?
- **Consistency**: The database is structurally sound and consistent after completing each transaction.
- **Isolation**: Transactions are isolated and don't contend with each other. Access to data from multiple users is moderated to avoid contention. Isolation guarantees that two transactions cannot coincide.
- **Durability**: After a transaction is completed, any changes a transaction makes should be durable and permanent, even in a failure such as a power failure.

The ACID model came before the BASE model, which we will describe next. If performance were not a consideration, using the ACID model would always be the right choice. BASE only came into the picture because the ACID model could not scale in many instances, especially with internet applications that serve a worldwide client base.

BASE data consistency model

ACID was taken as the law of the land for many years, but a new model emerged with the advent of bigger-scale projects and implementations. In many instances, the ACID model is more pessimistic than required, and it's *too safe* at the expense of scalability and performance.

In most NoSQL databases, the ACID model is not used. These databases have loosened some ACID requirements, such as data freshness, immediate consistency, and accuracy, to gain other benefits, such as scale, speed, and resilience. Some exceptions for a NoSQL database that uses the ACID models are the NET-based RavenDB database and Amazon DynamoDB within a single AWS account and region.

The acronym **BASE** can be broken down as follows – **Basic Availability**, **Soft-state**, and **Eventual consistency**. Let's explore what this means further:

- **Basic availability**: The data is available for the majority of the time (but not necessarily all the time). The BASE model emphasizes availability without guaranteeing the consistency of data replication when writing a record.
- **Soft-state**: The database doesn't have to be write-consistent, and different replicas don't always have to be mutually consistent. Take, for example, a system that reports sales figures in real-time to multiple destinations and uses multiple copies of the sales figures to provide fault tolerance. As sales come in and get written into the system, different readers may read a different copy of the sales figures. Some of them may be updated with the new numbers, and others may be a few milliseconds behind and not have the latest updates. In this case, the readers will have different results, but if they rerun the query soon after, they probably would get the new figures. In a system like this, not having the latest and greatest numbers may not end the world and may be good enough. The trade-off between getting the results fast versus being entirely up to date may be acceptable.
- **Eventual consistency**: The stored data exhibits consistency eventually and maybe not until the data is retrieved at a later point.

The BASE model requirements are looser than the ACID model ones, and a direct one-for-one relationship does not exist between ACID and BASE. The BASE consistency model is used mainly in aggregate databases (including wide-column databases), key-value databases, and document databases.

Let's look at the database usage model, which is a crucial differentiator when storing your data.

Database usage model

Two operations can be performed with a database: first, ingest data (or write data into the database), and second, retrieve data (or read data from the database). These two operations will always be present.

On the ingestion side, the data will be ingested in two different ways. It will either be a data update or brand-new data (such as an insert operation). To retrieve data, you will analyze the **change data capture** (**CDC**) set, which is changes in existing data or accessing brand new data. But what drives your choice of database is not the fact that these two operations are present but rather the following:

- How often will the data be retrieved?
- How fast should it be accessed?
- Will the data be updated often, or will it be primarily new?
- How often will the data be ingested?
- How fast does ingestion need to be?
- Will the ingested data be sent in batches or in real time?
- How many users will be consuming the data?
- How many simultaneous processes will there be for ingestion?

The answers to these questions will determine what database technology to use. Two technologies have been the standards to address these questions for many years: **online transaction processing** (**OLTP**) systems and **online analytics processing** (**OLAP**) systems. The main question that needs to be answered is - *is it more important for the database to perform during data ingestion or retrieval?* These databases can be divided into two categories depending on the use case; they need to be read-heavy or write-heavy.

Online Transaction Processing (OLTP) systems

OLTP databases' main characteristics are the fact that they process a large number of transactions (such as inserts and updates). The focus in OLTP systems is placed on fast ingestion and modification of data while maintaining data integrity, typically in a multi-user environment with less emphasis on the retrieval of the data. OLTP performance is generally measured by the number of transactions executed in a given time (usually seconds). Data is typically stored using a schema that has been normalized, usually using the **3rd normal form** (**3NF**). Before moving on, let's quickly discuss 3NF. 3NF is a state that a relational database schema design can possess.

A table using 3NF will reduce data duplication, minimize data anomalies, guarantee referential integrity, and increase data management. 3NF was first specified in 1971 by *Edgar F. Codd*, the inventor of the relational model for database management.

A database relation (for example, a database table) meets the 3NF standard if each table's columns only depend on the table's primary key. Let's look at an example of a table that fails to meet 3NF. Let's say you have a table that contains a list of employees. This table, in addition to other columns, contains the employee's supervisor's name as well as the supervisor's phone number. A supervisor can undoubtedly have more than one employee under supervision, so the supervisor's name and phone number will be repeated for employees working under the same supervisor. To resolve this issue, we could add a supervisor table, put the supervisor's name and phone number in the supervisor table, and remove the phone number from the employee table.

Online Analytical Processing (OLAP) systems

Conversely, OLAP databases do not process many transactions. Once data is ingested, it is usually not modified. OLTP systems are not uncommon to be the source systems for OLAP systems. Data retrieval is often performed using some query language (the **Structured Query Language** (**SQL**)). Queries in an OLAP environment are often complex and involve subqueries and aggregations. In the context of OLAP systems, the performance of queries is the relevant measure. An OLAP database typically contains historical data aggregated and stored in multi-dimensional schemas (typically using the star schema).

For example, a bank might handle millions of daily transactions, storing deposits, withdrawals, and other banking data. The initial transactions will probably be stored using an OLTP system. The data might be copied to an OLAP system to run reporting based on the daily transactions and, once aggregated, for more extended reporting periods.

The following table shows a comparison between OLTP and OLAP:

	OLTP	OLAP
Focus	Insertion and modification of data.	Retrieval and analysis of data.
Data	OLTP data is normally the source of truth and original data.	OLAP systems are fed by OLTP systems.
Transaction	OLTP has short transactions. Usually a combination of updates and inserts.	OLAP has long transactions. Usually just inserts.
Time	Low processing time of transactions.	High processing time of transactions.
Queries	Simpler queries.	Complex queries.
Normalization	Usually normalized (3NF).	Usually not normalized.
Integrity	Important. Normally ACID.	Not as important. BASE can be used.

Figure 7.1: Comparison between OLTP systems and OLAP systems

As you have learned about the database consistency model and its uses, you must be wondering which model is suitable when combining these properties; ACID is a must-have for OLTP, and BASE can be applied for OLAP.

Let's go further and learn about the various kinds of database services available in AWS and how they fit to address different workload needs.

AWS database services

AWS offers a broad range of database services that are purpose-built for every major use case. These fully managed services allow you to build applications that scale quickly. All these services are battle-tested and provide deep functionality, so you get the high availability, performance, reliability, and security required by production workloads.

The suite of AWS fully managed database services encompasses relational databases for transactional applications, such as Amazon RDS and Amazon Aurora, non-relational databases like Amazon DynamoDB for internet-scale applications, an in-memory data store called Amazon ElastiCache for caching and real-time workloads, and a graph database, Amazon Neptune, for developing applications with highly connected data. Migrating your existing databases to AWS is made simple and cost-effective with the AWS Database Migration Service. Each of these database services is so vast that going into details warrants a book for each of these services itself. This section will show you various database services overviews and resources to dive further.

Relational databases

There are many offerings in the database space, but relational databases have served us well for many years without needing any other type of database. A relational database is probably the best, cheapest, and most efficient option for any project that does not store millions of records. So, let's analyze the different relational options that AWS offers us.

Amazon Relational Database Service (Amazon RDS)

Given what we said in the previous section, it is not surprising that Amazon has a robust lineup of relational database offerings. They all fall under the umbrella of Amazon RDS. It is certainly possible to install your database into an EC2 instance and manage it yourself. Unless you have an excellent reason to do so, it may be a terrible idea; instead, you should consider using one of the many flavors of Amazon RDS. You may think running your instance might be cheaper, but if you consider all the costs, including system administration costs, you will most likely be better off and save money using Amazon RDS.

Amazon RDS was designed by AWS to simplify the management of crucial transactional applications by providing an easy-to-use platform for setting up, operating, and scaling a relational database in the cloud. With RDS, laborious administrative tasks such as hardware provisioning, database configuration, patching, and backups are automated, and a scalable capacity is provided in a cost-efficient manner. RDS is available on various database instance types, optimized for memory, performance, or I/O, and supports six well-known database engines, including Amazon Aurora (compatible with MySQL and PostgreSQL), MySQL, PostgreSQL, MariaDB, SQL Server, and Oracle.

If you want more control of your database at the OS level, AWS has now launched **Amazon RDS Custom**. It provisions all AWS resources in your account, enabling full access to the underlying Amazon EC2 resources and database environment access.

You can install third-party and packaged applications directly onto the database instance as they would have in a self-managed environment while benefiting from the automation that Amazon RDS traditionally provides.

Amazon RDS's flavors fall into three broad categories:

- **Community** (Postgres, MySQL, and MariaDB): AWS offers RDS with three different open-source offerings. This is a good option for development environments, low-usage deployments, defined workloads, and non-critical applications that can afford some downtime.
- **Amazon Aurora** (Postgres and MySQL): As you can see, Postgres and MySQL are here, as they are in the community editions. Is this a typo? No, delivering these applications within the Aurora *wrapper* can add many benefits to a community deployment. Amazon started offering the MySQL service in 2014 and added the Postgres version in 2017. Some of these are as follows:
    a. Automatic allocation of storage space in 10 GB increments up to 64 TBs
    b. Fivefold performance increase over the vanilla MySQL version
    c. Automatic six-way replication across availability zones to improve availability and fault tolerance
- **Commercial** (Oracle and SQLServer): Many organizations still run Oracle workloads, so AWS offer RDS with an Oracle flavor (and a Microsoft SQL Server flavor). Here, you will get all the benefits of a fully managed service. However, keep in mind that, bundled with the cost of this service, there will be a licensing cost associated with using this service, which otherwise might not be present if you use a community edition.

Let's look at the key benefits of Amazon RDS.

Amazon RDS Benefits

Amazon RDS offers multiple benefits as a managed database service offered by AWS. Let's look at its key attributes to make your database more resilient and performant.

Multi-AZ deployments - Multi-AZ deployments in RDS provide improved availability and durability for database instances, making them an ideal choice for production database workloads. With Multi-AZ DB instances, RDS synchronously replicates data to a standby instance in a different **Availability Zone (AZ)** for enhanced resilience. You can change your environment from Single-AZ to Multi-AZ at any time. Each AZ runs on its own distinct, independent infrastructure and is built to be highly dependable.

In the event of an infrastructure failure, RDS initiates an automatic failover to the standby instance, allowing you to resume database operations as soon as the failover is complete. Additionally, the endpoint for your DB instance remains the same after a failover, eliminating manual administrative intervention and enabling your application to resume database operations seamlessly.

Read replicas - RDS makes it easy to create read replicas of your database and automatically keeps them in sync with the primary database (for MySQL, PostgreSQL, and MariaDB engines). Read replicas are helpful for both read scaling and disaster recovery use cases. You can add read replicas to handle read workloads, so your master database doesn't become overloaded with reading requests. Depending on the database engine, you may be able to position your read replica in a different region than your master, providing you with the option of having a read location that is closer to a specific locality. Furthermore, read replicas provide an additional option for failover in case of an issue with the master, ensuring you have coverage in the event of a disaster.

While both Multi-AZ deployments and read replicas can be used independently, they can also be used together to provide even greater availability and performance for your database. In this case, you would create a Multi-AZ deployment for your primary database and then create one or more read replicas of that primary database. This would allow you to benefit from the automatic failover capabilities of Multi-AZ deployments and the performance improvements provided by read replicas.

Automated backup - With RDS, a scheduled backup is automatically performed once a day during a time window that you can specify. The backup job is monitored as a managed service to ensure its successful completion within the specified time window. The backups are comprehensive and include both your entire instance and transaction logs. You have the flexibility to choose the retention period for your backups, which can be up to 35 days. While automated backups are available for 35 days, you can retain longer backups using the manual snapshots feature provided by RDS. RDS keeps multiple copies of your backup in each AZ where you have an instance deployed to ensure their durability and availability. During the automatic backup window, storage I/O might be briefly suspended while the backup process initializes, typically for less than a few seconds. This may cause a brief period of elevated latency. However, no I/O suspension occurs for Multi-AZ DB deployments because the backup is taken from the standby instance. This can help achieve high performance if your application is time-sensitive and needs to be always on.

Database snapshots - You can manually create backups of your instance stored in Amazon S3, which are retained until you decide to remove them. You can use a database snapshot to create a new instance whenever needed. Even though database snapshots function as complete backups, you are charged only for incremental storage usage.

Data storage - Amazon RDS supports the most demanding database applications by utilizing **Amazon Elastic Block Store** (**Amazon EBS**) volumes for database and log storage. There are two SSD-backed storage options to choose from: a cost-effective general-purpose option and a high-performance OLTP option. Amazon RDS automatically stripes across multiple Amazon EBS volumes to improve performance based on the requested storage amount.

Scalability - You can often scale your RDS database compute and storage resources without downtime. You can choose from over 25 instance types to find the best fit for your CPU, memory, and price requirements. You may want to scale your database instance up or down, including scaling up to handle the higher load, scaling down to preserve resources when you have a lower load, and scaling up and down to control costs if you have regular periods of high and low usage.

Monitoring - RDS offers a set of 15-18 monitoring metrics that are automatically available for you. You can access these metrics through the RDS or CloudWatch APIs. These metrics enable you to monitor crucial aspects such as CPU utilization, memory usage, storage, and latency. You can view the metrics in individual or multiple graphs or integrate them into your existing monitoring tool. Additionally, RDS provides Enhanced Monitoring, which offers access to more than 50 additional metrics. By enabling Enhanced Monitoring, you can specify the granularity at which you want to view the metrics, ranging from one-second to sixty-second intervals. This feature is available for all six database engines supported by RDS.

Amazon RDS Performance Insights is a performance monitoring tool for Amazon RDS databases. It allows you to monitor the performance of your databases in real-time and provides insights and recommendations for improving the performance of your applications. With Performance Insights, you can view a graphical representation of your database's performance over time and detailed performance metrics for specific database operations. This can help you identify any potential performance bottlenecks or issues and take action to resolve them.

Performance Insights also provides recommendations for improving the performance of your database. These recommendations are based on best practices and the performance data collected by the tool and can help you optimize your database configuration and application code to improve the overall performance of your application.

Security - Controlling network access to your database is made simple with RDS. You can run your database instances in **Amazon Virtual Private Cloud** (**Amazon VPC**) to isolate them and establish an industry-standard encrypted IPsec VPN to connect with your existing IT infrastructure. Additionally, most RDS engine types offer encryption at rest, and all engines support encryption in transit. RDS offers a wide range of compliance readiness, including HIPAA eligibility.

You can learn more about RDS by visiting the AWS page: `https://aws.amazon.com/rds/`.

As you have learned about RDS, let's dive deeper into AWS cloud-native databases with Amazon Aurora.

Amazon Aurora

Amazon Aurora is a relational database service that blends the availability and rapidity of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. Aurora is built with full compatibility with MySQL and PostgreSQL engines, enabling applications and tools to operate without necessitating modifications. It offers a variety of developer tools to construct serverless and **machine learning (ML)**-driven applications. The service is completely managed and automates time-intensive administration tasks, including hardware provisioning, database setup, patching, and backups. It provides commercial-grade databases' reliability, availability, and security while costing only a fraction of the price.

Amazon Aurora has many key features that have been added to expand the service's capabilities since it launched in 2014. Let's review some of these key features:

- **Serverless configuration** - Amazon Aurora Serverless is a configuration of Aurora that offers auto-scaling features on-demand. With this configuration, your database will automatically start up, shut down, and adjust its capacity based on the needs of your application. Amazon Aurora Serverless v2 scales almost instantly to accommodate hundreds of thousands of transactions in seconds. It fine-tunes its capacity in small increments to ensure the right resources for your application. You won't have to manage the database capacity, and you'll only pay for your application's resources. Compared to peak load provisioning capacity, you could save up to 90% of your database cost with Amazon Aurora Serverless.
- **Global Database** - To support globally distributed applications, you can leverage the Global Database feature of Aurora. This enables you to span a single Aurora database across multiple AWS regions, allowing for faster local reads and rapid disaster recovery. Global Database utilizes storage-based replication to replicate your database across various regions, typically resulting in less than one-second latency. By utilizing a secondary region, you can have a backup option in case of a regional outage or degradation and can quickly recover. Additionally, it takes less than one minute to promote a database in the secondary region to full read/write capabilities.

- **Encryption** - With Amazon Aurora, you can encrypt your databases by using keys you create and manage through **AWS Key Management Service (AWS KMS)**. When you use Amazon Aurora encryption, data stored on the underlying storage and automated backups, snapshots, and replicas within the same cluster are encrypted. Amazon Aurora secures data in transit using SSL (AES-256).
- **Automatic, continuous, incremental backups and point-in-time restore** - Amazon Aurora offers a backup feature that enables you to recover your instance to any specific second during your retention period, up to the last five minutes. This capability is known as point-in-time recovery. You can configure your automatic backup retention period for up to thirty-five days. The automated backups are stored in **Amazon Simple Storage Service (Amazon S3)**, which is designed for 99.999999999% durability. The backups are incremental, continuous, and automatic, and they have no impact on database performance.
- **Multi-AZ Deployments with Aurora Replicas** - In the event of an instance failure, Amazon Aurora leverages Amazon RDS Multi-AZ technology to perform an automated failover to one of the up to 15 Amazon Aurora Replicas you have established across three AZs. If you have not provisioned any Amazon Aurora Replicas, in the event of a failure, Amazon RDS will automatically attempt to create a new Amazon Aurora DB instance for you.
- **Compute Scaling** - You can scale the provisioned instances powering your deployment up or down using either the Amazon RDS APIs or the AWS Management Console. The process of compute scaling typically takes only a few minutes to complete.
- **Storage auto-scaling** - Amazon Aurora automatically scales the size of your database volume to accommodate increasing storage requirements. The volume expands in 10 GB increments, up to 128 TB, as your storage needs grow. There's no need to provision extra storage to handle the future growth of your database.
- **Fault-tolerant and self-healing storage** - Each 10 GB chunk of your database volume is replicated six times across three Availability Zones, making Amazon Aurora storage fault-tolerant. It can handle the loss of up to two data copies without affecting write availability, and up to three copies without affecting read availability. Amazon Aurora storage is also self-healing, continuously scanning data blocks and disks for errors and replacing them automatically.
- **Network isolation** - Amazon Aurora operates within Amazon VPC, providing you with the ability to segregate your database within your own virtual network and connect to your existing on-premises IT infrastructure via industry-standard encrypted IPsec VPNs. Additionally, you can manage firewall configurations through Amazon RDS and govern network access to your DB instances.

- **Monitoring and metrics** - Amazon Aurora offers a range of monitoring and performance tools to help you keep your database instances running smoothly. You can use Amazon CloudWatch metrics at no additional cost to monitor over 20 key operational metrics, such as compute, memory, storage, query throughput, cache hit ratio, and active connections. If you need more detailed insights, you can use Enhanced Monitoring to gather metrics from the operating system instance that your database runs on. Additionally, you can use Amazon RDS Performance Insights, a powerful database monitoring tool that provides an easy-to-understand dashboard for visualizing database load and detecting performance problems, so you can take corrective action quickly.
- **Governance** - AWS CloudTrail keeps track of and documents account activity across your AWS infrastructure, providing you with oversight over storage, analysis, and corrective actions. You can ensure your organization remains compliant with regulations such as SOC, PCI, and HIPAA by utilizing CloudTrail logs. The platform enables you to capture and unify user activity and API usage across AWS Regions and accounts in a centralized, controlled environment, which can help you avoid penalties.
- **Amazon Aurora machine learning** - With Amazon Aurora machine learning, you can incorporate machine learning predictions into your applications through SQL programming language, eliminating the need to acquire separate tools or possess prior machine learning experience. It offers a straightforward, optimized, and secure integration between Aurora and AWS ML services, eliminating the need to create custom integrations or move data between them.

Enterprise use cases for Amazon Aurora span multiple industries. Here are examples of some of the key use cases where Amazon Aurora is an excellent fit, along with specific customer references for each:

- **Revamp corporate applications** - Ensure high availability and performance of enterprise applications, including CRM, ERP, supply chain, and billing applications.
- **Build a Software-as-a-Service (SaaS) application** - Ensure flexible instance and storage scaling to support dependable, high-performing, and multi-tenant SaaS applications. Amazon Aurora is a good choice for building a SaaS application, as it provides the scalability, performance, availability, and security features that are essential for successful SaaS applications. Amazon Aurora automatically creates and maintains multiple replicas of your data, providing high availability and failover capabilities. This ensures that your SaaS application is always available, even in the event of an outage or failure.

Amazon Aurora provides several security features, such as encryption at rest and in transit, to help protect your data and ensure compliance with industry standards and regulations.

- **Deploy globally distributed applications** - Achieve multi-region scalability and resilience for internet-scale applications, such as mobile games, social media apps, and online services, with Aurora's flexible instance and storage scaling. To meet high read or write requirements, databases are often split across multiple instances, but this can lead to over-provisioning or under-provisioning, resulting in increased costs or limited scalability. Aurora Serverless solves this problem by automatically scaling the capacity of multiple Aurora instances based on the application's needs, allowing for efficient scaling of databases and enabling the deployment of globally distributed applications. This can provide several benefits, including cost savings, flexibility, and simplicity.
- **Variable and unpredictable workloads** - If you run an infrequently-used application where you need to provision for peak, it will require you to pay for unused resources. A surge in traffic can also be mitigated through the automatic scaling of Aurora Serverless. You will not need to manage or upsize your servers manually. You need to set a min/max capacity unit setting and allow Aurora to scale to meet the load.

Amazon RDS Proxy works together with Aurora to enhance database efficiency and application scalability by enabling applications to share and pool connections established with the database. Let's learn more details about RDS Proxy.

Amazon RDS Proxy

Amazon RDS Proxy is a service that acts as a database proxy for Amazon **Relational Database Service (RDS)**. It is fully managed by AWS and helps to increase the scalability and resilience of applications in the face of database failures, while enhancing the security of database traffic. RDS Proxy sits between your application and your database and automatically routes database traffic to the appropriate RDS instances. This can provide several benefits, including:

- **Improved scalability**: RDS Proxy automatically scales to handle a large number of concurrent connections, making it easier for your application to scale.
- **Better resilience to database failures**: RDS Proxy can automatically failover to a standby replica if the primary database instance becomes unavailable, reducing downtime and improving availability.
- **Enhanced security**: RDS Proxy can authenticate and authorize incoming connections, helping to prevent unauthorized access to your database. It can also encrypt data in transit, providing an extra security measure for your data.

The following diagram shows Amazon EC2 web server provision with an Aurora database where database passwords are managed by AWS Secrets Manager. Aurora put across 2 **Availability Zones (AZs)** to achieve high availability, where AZ1 hosts Aurora's primary database while AZ2 has the Aurora read replica.

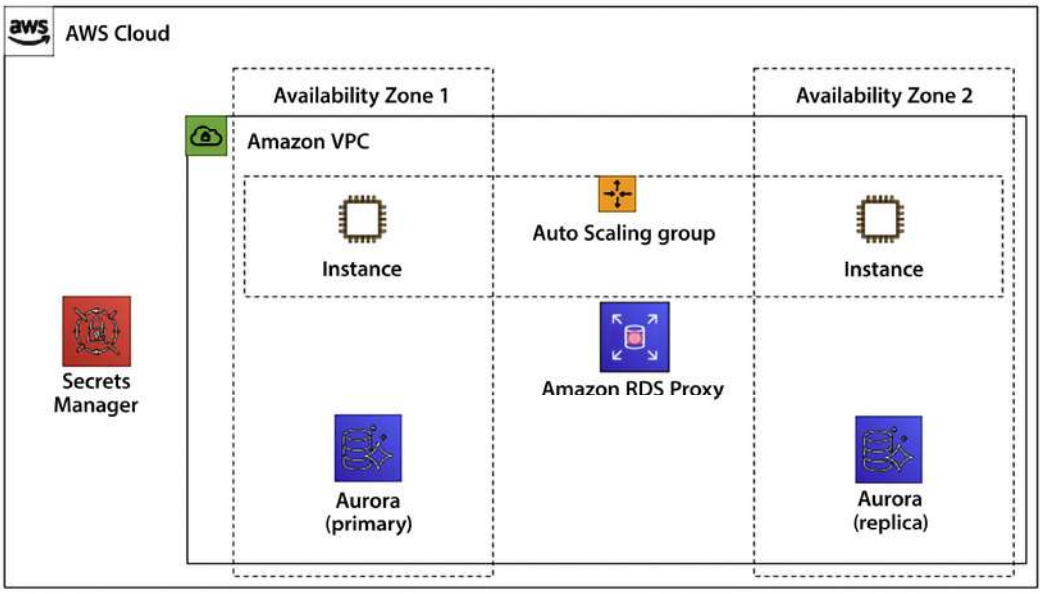

Figure 7.2: Amazon Aurora's high availability with RDS proxy

With RDS Proxy, when a failover happens, application connections are preserved. Only transactions that are actively sending or processing data will be impacted. During failover, the proxy continues to accept new connections. These connections will queue until the database connection comes online; at that point, it then gets sent over to the database.

Amazon RDS Proxy is a useful tool for improving the performance, availability, and security of your database-powered applications. It is fully managed, so you don't have to worry about the underlying infrastructure, and it can help make your applications more scalable, resilient, and secure. You can learn more about RDS Proxy by visiting the AWS page: `https://aws.amazon.com/rds/proxy/`.

High availability and performance are a database's most essential and tricky parts. But this problem can be solved in an intelligent way using machine learning. Let's look at RDS's newly launched feature, Amazon DevOps Guru, to help with database performance issues using ML.

Amazon DevOps Guru for RDS

DevOps Guru for Amazon RDS is a recently introduced capability that uses machine learning to automatically identify and troubleshoot performance and operational problems related to relational databases in an application. The tool can detect issues such as over-utilization of resources or problematic SQL queries and provide recommendations for resolution, helping developers address these issues quickly. DevOps Guru for Amazon RDS utilizes machine learning models to deliver these insights and suggestions.

DevOps Guru for RDS aids in quickly resolving operational problems related to databases by notifying developers immediately via Amazon **Simple Notification Service (SNS)** notifications and EventBridge when issues arise. It also provides diagnostic information, as well as intelligent remediation recommendations, and details on the extent of the issue.

AWS keeps adding innovations for Amazon RDS as a core service. Recently, they launched Amazon RDS instances available on AWS's chip Graviton2, which helps them offer lower prices with increased performance. RDS is now available in Amazon Outpost to fulfill your need to keep the database near your workload in an on-premise environment. You can learn more about RDS Proxy by visiting the AWS page: `https://aws.amazon.com/devops-guru/`.

Besides SQL databases, Amazon's well-known NoSQL databases are very popular. Let's learn about some NoSQL databases.

AWS NoSQL databases

When it comes to NoSQL databases, you need to understand two main categories, key-value and document, as those can be confusing. The key-value database needs high throughput, low latency, reads and writes, and endless scale, while the document database stores documents and quickly accesses querying on any attribute. Document and key-value databases are close cousins. Both types of databases rely heavily on keys that point to a value. The main difference between them is that in a document database, the value stored (the document) will be **transparent** to the database and, therefore, can be indexed to assist retrieval. In the case of a key-value database, the value is **opaque** and will not be scannable, indexed, or visible until the value is retrieved by specifying the key. Retrieving a value without using the key would require a full table scan. Content is stored as the value. To retrieve the content, you query using the unique key, enabling access to the value.

Key-value databases are the simpletons of the NoSQL world. They are pretty easy to use. There are three simple API operations:

- Retrieve the value for a key.
- Insert or update a value using a key reference.
- Delete a value using the key reference.

The values are generally **Binary Large Objects (BLOBs)**. Data stores keep the value without regard for the content. Data in key-value database records is accessed using the key (in rare instances, it might be accessed using other filters or a full table scan). Therefore, performance is high and scalable. The biggest strength of key-value databases is always their biggest weakness. Data access is quick because we use the key to access the data. Still, full table scans and filtering operations are neither supported nor a secondary consideration. Key-value stores often use the hash table pattern to store the keys. No column-type relationships exist, which keeps the implementation details simple. Starting in the key-value category, AWS provides Amazon DynamoDB. Let's learn more about Dynamo DB.

Amazon Dynamo DB

DynamoDB is a fully managed, multi-Region, multi-active database that delivers exceptional performance, with single-digit-millisecond latency, at any scale. It is capable of handling more than 10 trillion daily requests, with the ability to support peaks of over 20 million requests per second, making it an ideal choice for internet-scale applications. DynamoDB offers built-in security, backup and restore features, and in-memory caching. One of the unique features of DynamoDB is its elastic scaling, which allows for seamless growth as the number of users and required I/O throughput increases. You pay only for the storage and I/O throughput you provision, or on a consumption-based model if you choose on-demand. The database can be provisioned with additional capacity on the fly to maintain high performance, making it easy for an application to support millions of users making thousands of concurrent requests every second. In addition, DynamoDB offers fine-grained access control and support for end-to-end encryption to ensure data security.

Some of DynamoDB's benefits are as follows:

- Fully managed
- Supports multi-region deployment
- Multi-master deployment
- Fine-grained identity and access control

- Seamless integration with IAM security
- In-memory caching for fast retrieval
- Supports ACID transactions
- Encrypts all data by default

DynamoDB provides the option of on-demand backups for archiving data to meet regulatory requirements. This feature enables you to create full backups of your DynamoDB table's data. Additionally, you can enable continuous backups for point-in-time recovery, allowing restoration to any point in the last 35 days with per-second granularity. All backups are automatically encrypted, cataloged, and retained until explicitly deleted. This feature ensures backups are easily discoverable and helps meet regulatory requirements.

DynamoDB is built for high availability and durability. All writes are persisted on an SSD disk and replicated to 3 availability zones. Reads can be configured as "strong" or "eventual." There is no latency trade-off with either configuration; however, the read capacity is used differently. Amazon **DynamoDB Accelerator (DAX)** is a managed, highly available, in-memory cache for DynamoDB that offers significant performance improvements, up to 10 times faster than standard DynamoDB, even at high request rates. DAX eliminates the need for you to manage cache invalidation, data population, or cluster management, and delivers microseconds of latency by doing all the heavy lifting required to add in-memory acceleration to your DynamoDB tables.

Defining Dynamo DB Table

DynamoDB is a table-based database. While creating the table, you can specify at least three components:

1. **Keys**: There will be two parts of the key – first, a partition key to retrieve the data, and second, a sort key to sort and retrieve a batch of data in a given range. For example, transaction ID can be your primary key, and transaction date-time can be the sort key.
2. **WCU**: **Write capacity unit** (1 KB/sec) defines at what rate you want to write your data in DynamoDB.
3. **RCU**: **Read capacity unit** (4 KB/sec) defines at what rate you want to read from your given DynamoDB table.

The size of the table automatically increases as you add more items. There is a hard limit of item size at 400 KB. As size increases, the table is partitioned automatically for you. The size and provisioning capacity of the table are equally distributed for all partitions.

As shown in the below diagram, the data is stored in tables; you can think of a table as the database, and within the table, you have items.

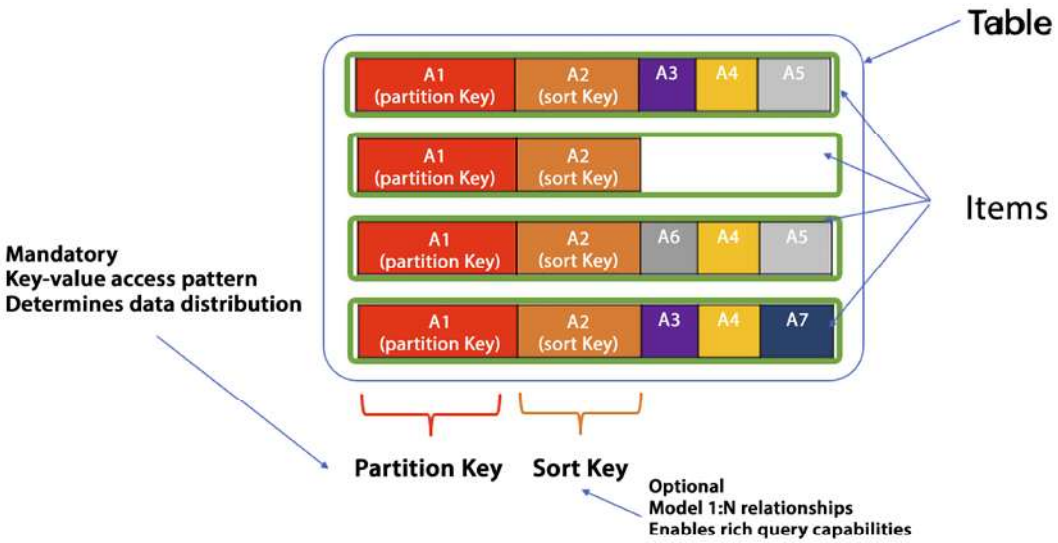

Figure 7.3: DynamoDB table with partition and sort key

As shown in the preceding diagram, the first item has five attributes, and the next item has only two. As more items are added to the table in DynamoDB, it becomes apparent that attributes can differ between items, and each item can have a unique set of attributes. Additionally, the primary key or partition key can be observed, which uniquely identifies each item and determines how the data is partitioned and stored. The partition key is required, while the sort key is optional but useful for establishing one-to-many relationships and facilitating in-range queries.

Sometimes you need to query data using the primary key, and sometimes you need to query by an attribute that is not your primary/secondary key or sort key. To tackle this issue, DynamoDB provides two kinds of indexes:

- **Local Secondary Index (LSI)** - Let's say you want to find out all fulfilled orders. You would have to query for all orders and then look for fulfilled ones in the results, which is not very efficient with large tables. But you have LSIs to help us out. You can create an LSI with the same primary key (`order_ID`) and a different secondary key (`fulfilled`). Now your query can be based on the key of the LSI. This is fast and efficient. The LSI is located on the same partition as the item in the table, ensuring consistency. Whenever an item is updated, the corresponding LSI is also updated and acknowledged.

The same primary key is used to partition both the LSI and the parent table, even though the parent table can have a different sort key. In the index, you can choose to have just the keys or other attributes projected or include all attributes – depending on what attributes you want to be returned with the query. There is a limit of 10 GB on LSI storage as it uses the partition storage of the original table.

- **Global Secondary Index (GSI)** -A GSI is an extension of the concept of indexes, allowing for more complex queries with various attributes as query criteria. In some cases, using the existing primary/sort key does not suffice. To address this, you can define a GSI as a parallel or secondary table with a partition key that is different from the original table and an alternate sort key. When creating a GSI, you must specify the expected workload for read and write capacity units. Similar to an LSI, you can choose to have just the keys or other attributes projected or include all attributes – depending on what attributes you want to be returned with the query.

The followings are the key differences between an LSI and GSI:

Local Secondary Index (LSI)	**Global Secondary Index (GSI)**
You have to define an LSI upfront, and it can be created during table creation only.	You can define a GSI at any time, even after table creation.
LSI shares WCU/RCU with the main table, so you must have enough read/write capacity to accommodate LSI need.	GSI WCU/RCU is independent of the table, so it can scale without impacting the main table.
LSI size is a maximum of 10 GB in sync with the primary table partition, which is limited to 10 GB size per partition.	As GSI is independent of the main table, so it has no size limits.
You can create a maximum of 5 LSIs.	You can create up to 20 GSIs.
As LSIs tie up to the main table, they offer "strong consistency," which means their current access to the most updated data.	GSI offer "eventual consistency," which means there may be a slight lag on data updates and lowers the chances you get stable data.

Table 7.3: DynamoDB Index – LSI vs. GSI

So, you may ask when to use an LSI vs. GSI. In a nutshell, a GSI is more flexible. An LSI is useful when you want to query data based on an alternate sort key within the same partition key as the base table.

It allows you to perform fast, efficient queries with minimal latency, and is best suited for scenarios where you know the specific queries that will be performed on your data. A GSI, on the other hand, allows you to query data based on attributes that are not part of the primary key or sort key of the base table. It's useful when you want to perform ad-hoc queries on different attributes of the data or when you need to support multiple access patterns. A GSI can be used to scale read queries beyond the capacity of the base table and can also be used to query data across partitions.

In general, if your data size is small enough, and you only need to query data based on a different sort key within the same partition key, you should use an LSI. If your data size is larger, or you need to query data based on attributes that are not part of the primary key or sort key, you should use a GSI. However, keep in mind that a GSI comes with some additional cost and complexity in terms of provisioned throughput, index maintenance, and eventual consistency.

If an item collection's data size exceeds 10 GB, the only option is to use a GSI as an LSI limits the data size in a particular partition. If eventual consistency is acceptable for your use case, a GSI can be used as it is suitable for 99% of scenarios.

DynamoDB is very useful in designing serverless event-driven architecture. You can capture the item-level data change, e.g., `putItem`, `updateItem`, and `delete`, by using DynamoDB Streams. You can learn more about Amazon DynamoDB by visiting the AWS page: `https://aws.amazon.com/dynamodb/`.

You may want a more sophisticated database when storing JSON documents for index and fast search. Let's learn about the AWS document database offering.

Amazon DocumentDB

In this section, let's talk about Amazon DocumentDB, *"a fully managed MongoDB-compatible database service designed from the ground up to be fast, scalable, and highly available"*. DocumentDB is a purpose-built document database engineered for the cloud that provides millions of requests per second with millisecond latency and can scale-out to 15 read replicas in minutes. It is compatible with MongoDB 3.6/4.0 and managed by AWS, which means no hardware provisioning, auto-patching, quick setup, good security, and automatic backups are needed.

If you look at the evolution of document databases, what was the need for document databases in this new world? At some point, JSON became the de facto standard for data interchange and data modeling within applications. Using JSON in the application and then trying to map JSON to relational databases introduced friction and complication.

Object Relational Mappers (ORMs) were created to help with this friction, but there were complications with performance and functionality. A crop of document databases popped up to solve the problem. The need for DocumentDB is primarily driven by the following reasons:

- Data is stored in documents that are in a JSON-like format, and these documents are considered as first-class objects within the database. Unlike traditional databases where documents are stored as values or data types, in DocumentDB, documents are the key design point of the database.
- Document databases offer flexible schemas, making it easy to represent hierarchical and semi-structured data. Additionally, powerful indexing capabilities make querying such documents much faster.
- Documents naturally map to object-oriented programming, which simplifies the flow of data between your application and the database.
- Document databases come with expressive query languages that are specifically built for handling documents. These query languages enable ad hoc queries and aggregations across documents, making it easier to extract insights from data.

Let's take user profiles, for example; let's say that Jane Doe plays a new game called ExplodingSnails, you can easily add that information to her profile, and you don't have to design a complicated schema or create any new tables—you simply add a new set of fields to your document. Similarly, you can add an array of Jane's promotions. The document model enables you to evolve applications quickly over time and build applications faster.

In the case of a document database, records contain structured or semi-structured values. This structured or semi-structured data value is called a document. It is typically stored using **Extensible Markup Language (XML)**, **JavaScript Object Notation (JSON)**, or **Binary JavaScript Object Notation (BSON)** format types.

What are document databases suitable for?

- Content management systems
- E-commerce applications
- Analytics
- Blogging applications

When are they not appropriate?

- Requirements for complex queries or table joins
- OLTP applications

Advantages of DocumentDB

Compared to traditional relational databases, Amazon DocumentDB offers several advantages, such as:

- **On-demand instance pricing**: You can pay by the hour without any upfront fees or long-term commitments, which eliminates the complexity of planning and purchasing database capacity ahead of time. This pricing model is ideal for short-lived workloads, development, and testing.
- **Compatibility with MongoDB 3.x and 4.x**: Amazon DocumentDB supports MongoDB 3.6 drivers and tools, allowing customers to use their existing applications, drivers, and tools with little or no modification. By implementing the Apache 2.0 open source MongoDB 4.x API on a distributed, fault-tolerant, self-healing storage system, Amazon DocumentDB offers the performance, scalability, and availability necessary for operating mission-critical MongoDB workloads at scale.
- **Migration support**: You can use the AWS **Database Migration Service (DMS)** to migrate MongoDB databases from on-premises, or on Amazon EC2, to Amazon DocumentDB at no additional cost (for up to six months per instance) with minimal downtime. DMS allows you to migrate from a MongoDB replica set or a sharded cluster to Amazon DocumentDB.
- **Flexible schema**: DocumentDB has a flexible schema, which means that the structure of the documents within the database can vary. This can be useful in situations where the data being stored has a complex or hierarchical structure, or when the data being stored is subject to frequent changes.
- **High performance**: DocumentDB is designed for high performance and can be well suited for applications that require fast read and write access to data.
- **Scalability**: DocumentDB is designed to be horizontally scalable, which means that it can be easily expanded to support large amounts of data and a high number of concurrent users.
- **Easy querying**: Document DB provides powerful and flexible query languages that make it easy to retrieve and manipulate data within the database.

DocumentDB has recently introduced new features that allow for ACID transactions across multiple documents, statements, collections, or databases. You can learn more about Amazon DocumentDB by visiting the AWS page: `https://aws.amazon.com/documentdb/`.

If you want sub-millisecond performance, you need your data in memory. Let's learn about in-memory databases.

In-memory database

Studies have shown that if your site is slow, even for just a few seconds, you will lose customers. A slow site results in 90% of your customers leaving the site. 57% of those customers will purchase from a similar retailer, and 25% of them will never return. These statistics are from a report by Business News Daily (`https://www.businessnewsdaily.com/15160-slow-retail-websites-lose-customers.html`). Additionally, you will lose 53% of your mobile users if a page load takes longer than 3 seconds (`https://www.business.com/articles/website-page-speed-affects-behavior/`).

We're no longer in the world of thinking our users are okay with a few seconds of wait time. These are demanding times for services, and to keep up with demand, you need to ensure that users aren't waiting to purchase your service, products, and offerings to continue growing your business and user base. There is a direct link between customers being forced to wait, and a loss of revenue.

Applications and databases have changed dramatically not only in the past 50 years but also just in the past 10. Where you might have had a few thousand users who could wait for a few seconds for an application to refresh, now you have microservices and IoT devices sending millions of requests per second across the globe that require immediate responses. These changes have caused the application and database world to rethink how data is stored and accessed. It's essential to use the right tool for the job. In-memory data stores are used when there is a need for maximum performance. This is achieved through extremely low latency per request and incredibly high throughput as you are caching data in memory, which helps to increase the performance by taking less time to read from the original database. Think of this as high-velocity data.

In-memory databases, or **IMDBs** for short, usually store the entire data in the main memory. Contrast this with databases that use a machine's RAM for optimization but do not store all the data simultaneously in primary memory and instead rely on disk storage. IMDBs generally perform better than disk-optimized databases because disk access is slower than direct memory access. In-memory operations are more straightforward and can be performed using fewer CPU cycles. In-memory data access seeks time when querying the data, which enables faster and more consistent performance than using long-term storage. To get an idea of the difference in performance, in-memory operations are usually measured in nanoseconds, whereas operations that require disk access are usually measured in milliseconds. So, in-memory operations are usually about a million times faster than operations needing disk access.

Some use cases of in-memory databases are real-time analytics, chat apps, gaming leaderboards, and caching. AWS provides Amazon ElastiCache to fulfill in-memory data caching needs.

As shown in the following diagram, based on your *data access pattern*, you can use either **lazy caching** or **write-through**. In lazy caching, the cache engine checks whether the data is in the cache and, if not, gets it from the database and keeps it in the cache to serve future requests. Lazy caching is also called the **cache aside pattern**.

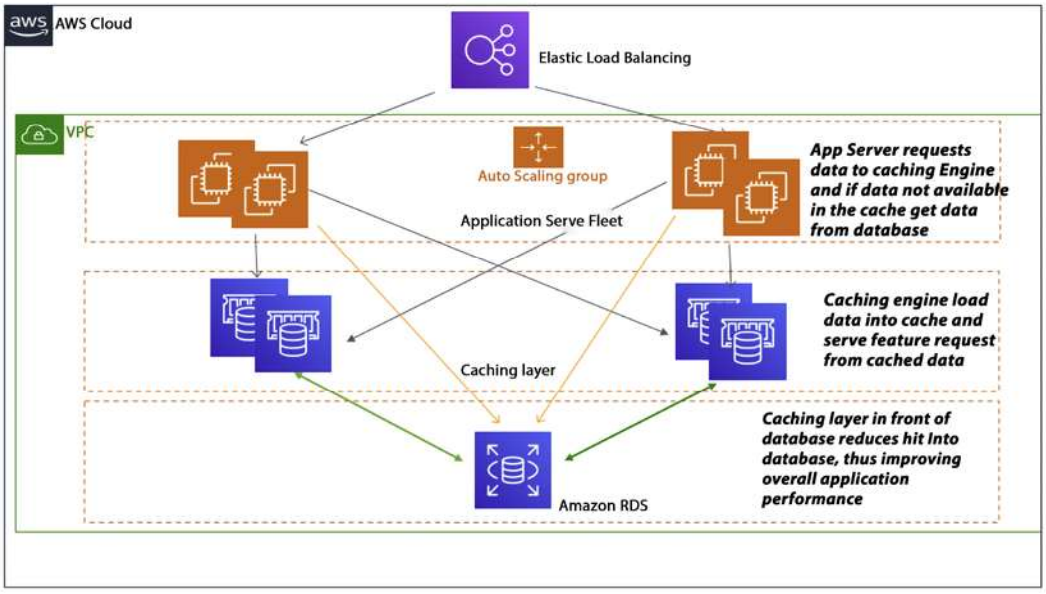

Figure 7.4: Application caching pattern architecture

You can see the caching flow in the above diagram, where the first app server sends a data request to the caching engine, which tries to load data from the cache. If data is not available in the cache, then the cache engine goes to the database and loads the required data into the cache. The Amazon ElastiCache service is an AWS-provided cache database. Let's learn more details about it.

Amazon ElastiCache

Amazon ElastiCache is a cloud-based web service that enables users to deploy and manage an in-memory cache with ease. By storing frequently accessed data in memory, in-memory caches can enhance application performance, enabling faster data access compared to retrieving data from a slower backing store such as a disk-based database. ElastiCache offers support for two popular open-source in-memory cache engines: Memcached and Redis. Both engines are well known for their reliability, scalability, and performance.

With ElastiCache, you can quickly and easily set up, manage, and scale an in-memory cache in the cloud.

ElastiCache automatically handles tasks such as provisioning cache nodes, configuring cache clusters, and monitoring the health of the cache environment, allowing you to focus on developing and deploying your application.

In addition, ElastiCache integrates seamlessly with other AWS services, making it easy to use the cache as a caching layer for other services like Amazon RDS or Amazon DynamoDB. This can help improve the performance and scalability of your overall application architecture.

Amazon ElastiCache enables users of the service to configure, run, and scale an IMDB and to build data-intensive applications. In-memory data storage boosts applications' performance by retrieving data directly from memory.

Redis versus Memcached

Since ElastiCache offers two flavors of in-memory databases, the obvious question is, which one is better? From our research, the answer now appears to be Redis, unless you are already a heavy user of Memcached. If your organization already has committed to Memcached, it is likely not worth porting it to Redis. But for new projects, the better option is Redis. This could change in the future, but as of this writing, Redis continues to gain supporters. Here is a comparison of the features and capabilities of the two:

	Memcached	Redis
Sub-millisecond latency	Yes	Yes
Developer ease of use	Yes	Yes
Data partitioning	Yes	Yes
Support for a broad set of programming languages	Yes	Yes
Advanced data structures	N/A	Yes
Multithreaded architecture	Yes	N/A
Snapshots	N/A	Yes
Replication	N/A	Yes
Transactions	N/A	Yes
Pub/Sub	N/A	Yes
Lua scripting	N/A	Yes
Geospatial support	N/A	Yes

Figure 7.5: Comparison of Redis and Memcached

The preceding table shows the key differences between Redis and Memcached. You can choose by validating your options, and you can learn more about Amazon ElastiCache by visiting the AWS page: `https://aws.amazon.com/elasticache/`.

AWS recently launched Amazon MemoryDB for Redis due to Redis's popularity. It is a durable, in-memory database service that provides ultra-fast performance and is compatible with Redis version 6.2. This service is designed explicitly for modern, microservice-based applications, and it offers Redis's flexible data structures, APIs, and commands. With Amazon MemoryDB, all data is stored in memory, allowing for microsecond read and single-digit millisecond write latency, as well as high throughput. You can learn more about Amazon MemoryDB by visiting the AWS page: `https://aws.amazon.com/memorydb/`.

Now data is getting more complicated with many-to-many relationships and several layers in social media. You need a specific database to drive the relationship, such as friends of friends and their common likes. Let's look at graph databases to solve this problem.

Graph databases

Graph databases are data stores that treat relationships between records as first-class citizens. In traditional databases, relationships are often an afterthought. In the case of relational databases, relationships are implicit and manifest themselves as foreign key relationships. In graph databases, relationships are explicit, significant, and optimized using graph database language; these relationships are called edges.

In some aspects, graph databases are similar to NoSQL databases. They are also schema-less. For certain use cases, they offer much better data retrieval performance than traditional databases. As you can imagine, graph databases are particularly suited for use cases that place heavy importance on relationships among entities.

Accessing data nodes and edges in a graph database is highly efficient. It usually can occur in constant time. With graph databases, it is not uncommon to be able to traverse millions of edges per second.

Graph databases can handle nodes with many edges regardless of the dataset's number of nodes. You only need a pattern and an initial node to traverse a graph database. Graph databases can easily navigate the adjacent edges and nodes around an initial starting node while caching and aggregating data from the visited nodes and edges. As an example of a pattern and a starting point, you might have a database that contains ancestry information. In this case, the starting point might be you, and the pattern might be a parent.

So, in this case, the query would return the names of both of your parents. The following are the components of a graph database:

- **Nodes**: Nodes are elements or entities in a graph. They contain a series of properties, attributes, or key-value pairs. Nodes can be given tags, which constitute roles in the domain. Node labels can be employed to assign metadata (such as indices or constraints) to the nodes.
- **Edges**: Edges supply directed, named, and semantically significant connections between two nodes. An edge has a direction, a type, a start node, and an end node. Like a node, an edge can also have properties. In some situations, an edge can have quantitative properties, such as weight, cost, and strength. Due to the efficient way an edge is stored, two nodes can share edges regardless of the quantity or type without a performance penalty. Edges have a direction, but edges can be traversed efficiently in both directions.

The following diagram shows the relationship between "Follower" and "Influencer" in a social media application, where the nodes depict the entity type (Follower and Influencer) and the edge (Influences) shows their relationship with their level of influence as the property weight. Here, the level of influence is 100, which means the Follower just started following the Influencer and with time, as they give more likes and views to the influencer's post, their level can increase to 200, 300, or 400.

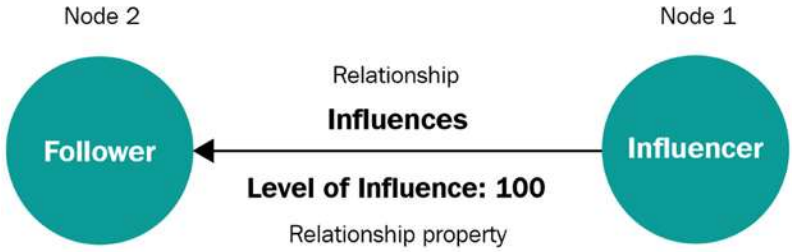

Figure 7.6: Example of a relationship

Two primary graph models are widely used. A property graph is a common name for an attributed, multi-relational graph. The leading property graph API is the open standard Apache TinkerPop™ project. It provides an imperative traversal language, called Gremlin, that can be used to write traversals on property graphs, and many open-source and vendor implementations support it. You may opt for property graphs to represent relational models, and the Apache TinkerPop Gremlin traversal language could be a favorable option as it offers a method to navigate through property graphs. You might also like openCypher, an open-source declarative query language for graphs, as it provides a familiar SQL-like structure to compose queries for graph data.

The second is the **Resource Description Framework (RDF)**, standardized by the W3C in a set of standards collectively known as the Semantic Web. The SPARQL query language for RDF allows users to express declarative graph queries against RDF graph models. The RDF model is also a labeled, directed multi-graph, but it uses the concept of triples, subject, predicate, and object, to encode the graph. Now let's look at Amazon Neptune, which is Amazon's graph database service.

Amazon Neptune

Amazon Neptune is a managed service for graph databases, which uses nodes, edges, and properties to represent and store data, making it a unique type of database. This data model is well suited to represent the complex relationships found in many types of data, such as the relationships between people in a social network or the interactions between different products on an e-commerce website.

Neptune supports the property graph and W3C's RDF standards, making it easy to integrate with other systems and tools that support these standards. Neptune also provides a query language called Gremlin which is powerful and easy to use, which makes it easy to perform complex graph traversals and data manipulation operations on the data stored in the database.

In addition, Neptune is highly scalable and available, with the ability to support billions of vertices and edges in a single graph. It is also fully managed, which means that Amazon takes care of the underlying infrastructure and performs tasks such as provisioning, patching, and backup and recovery, allowing you to focus on building and using your application. You can learn more about Amazon Neptune by visiting the AWS page: `https://aws.amazon.com/neptune/`.

A lot of data comes with a timestamp, and you need a specific database to store time-series data. Let's learn more about it.

Time-series databases

A **time-series database (TSDB)** is a database specifically designed and optimized to store events. What is an event, you ask? It is an action that happens at a specific point in time. With events, it's not only important to track *what* happened but just as important to track *when* it happened. The unit of measure to use for the time depends on the use case. For some applications, it might be enough to know on what day the event happened. But for other applications, it might be required to keep track of the time down to the millisecond. Some examples of projects that might benefit from a TSDB are as follows:

- Performance monitoring
- Networking and infrastructure applications

- Adtech and click stream processing
- Sensor data from IoT applications
- Event-driven applications
- Financial applications
- Log analysis
- Industrial telemetry data for equipment maintenance
- Other analytics projects

A TSDB is optimized to measure changes over time. Time series values can differ from other data types and require different optimization techniques.

Common operations in a TSDB are as follows:

- Millions of inserts from disparate sources potentially per second
- Summarization of data for downstream analytics
- Access to individual events

TSDBs are ideally suited for storing and processing IoT data. Time-series data has the following properties (which might not be present with other data types):

- The order in which the events occur may be necessary.
- Data is only inserted; it is not updated.
- Queries have a time interval component in their filters.

RDBMSes can store this data, but they are not optimized to process, store, and analyze this type of data. Amazon Timestream was purpose-built exclusively for this data type and, therefore, is much more efficient.

Do you feel comfortable about when you should use TSDBs? If you need to store events or track logs or trades, or the time and date when something happened to take center stage, then a TSDB is probably an excellent solution to your problem.

Amazon Timestream

Amazon Timestream is a scalable and fully managed TSDB. Amazon Timestream can persist and analyze billions of transactions per minute at about 1/10 of the cost of RDBMS equivalents. IoT devices and smart industrial machines are becoming more popular by the day. These applications generate events that need to be tracked and measured, sometimes with real-time requirements.

Amazon Timestream has an adaptive query processing engine that can make heads or tails of time-series data as it comes in by inferring data location and data format. Amazon Timestream has features that can automate query rollups, retention, tiering, and data compression. Like many other Amazon services, Timestream is serverless, so it can automatically scale up or down depending on how much data is coming into the streams. Also, because it's serverless and fully managed, tasks such as provisioning, operating system patching, configuration, backups, and tiering are not the responsibility of the DevOps team, allowing them to focus on more important tasks.

Timestream enables you to store multiple measures in a single table row with its multi-measure records feature, instead of one measure per table row, making it easier to migrate existing data from relational databases to Timestream with minimal changes. Scheduled computations are also available, allowing you to define a computation or query and its schedule. Timestream will automatically and periodically run the queries and store the results in a separate table. Additionally, Timestream automatically determines whether data should be written to the memory or magnetic store based on the data's timestamp and configured data retention window, thereby reducing costs. You can learn more about Amazon Timestream by visiting the AWS page: `https://aws.amazon.com/timestream/`.

You often want to make your database tamper-proof and keep a close record of any activity. To serve this purpose, let's learn about ledger databases.

Ledger databases

A **ledger database** (**LDB**) is a database that delivers a cryptographically verifiable, immutable, and transparent transaction log orchestrated by a central authority:

- **LDB immutability**: Imagine you deposit $1,000 in your bank. You see on your phone that the deposit was carried out, and it now shows a balance of $1,000. Then imagine you recheck it tomorrow, and this time, it says $500. You would not be too pleased, would you? The bank needs to ensure that the transaction is immutable and that no one can change it after the fact. In other words, only inserts are allowed, and updates cannot be performed. This assures that transactions cannot be changed once they are persisted.
- **LDB transparency**: In this context, transparency refers to the ability to track changes to the data over time. The LDB should be able to keep an audit log. This audit log, at a minimum, should include who changed the data, when the data was changed, and what the value of the data was before it was changed.

- **LDB cryptographic verifiability**: How can we ensure that our transaction will be immutable? Even though the database might not support update operations, what's stopping someone from using a backdoor and updating the record? If we use cryptography when the transaction is recorded, the entire transaction data is hashed. In simple terms, the string of data that forms the transaction is whittled down into a smaller string of unique characters. Whenever the transaction is hashed, it needs to match that string. In the ledger, the hash comprises the transaction data and appends the previous transaction's hash. Doing this ensures that the entire chain of transactions is valid. If someone tried to enter another transaction in between, it would invalidate the hash, and it would detect that the foreign transaction was added via an unauthorized method.

The prototypical use case for LDBs is bank account transactions. If you use an LDB for this case, the ledger records all credits and debits related to the bank account. It can then be followed from a point in history, allowing us to calculate the current account balance. With immutability and cryptographic verification, we are assured that the ledger cannot be deleted or modified. With other methods, such as RDBMSes, all transactions could be changed or erased.

Amazon Quantum Ledger Database (QLDB)

Amazon QLDB is a fully managed service that provides a centralized trusted authority to manage an immutable, transparent, and cryptographically verifiable ledger. QLDB keeps track of application value changes and manages a comprehensive and verifiable log of changes to the database.

Historically, ledgers have been used to maintain a record of financial activities. Ledgers can keep track of the history of transactions that need high availability and reliability. Some examples that need this level of reliability are as follows:

- Financial credits and debits
- Verifying the data lineage of a financial transaction
- Tracking the location history of inventory in a supply chain network

Amazon QLDB offers various blockchain services, such as anonymous data sharing and smart contracts, while still using a centrally trusted transaction log.

QLDB is designed to act as your system of record or source of truth. When you write to QLDB, your transaction is committed to the journal. The journal is what provides immutability through append-only interactions and verifiability through cryptographic hashing. QLDB treats all interactions, such as reads, inserts, updates, and deletes, like a transaction and catalogs everything sequentially in this journal.

Once the transaction is committed to the journal, it is immediately materialized into tables and indexes. QLDB provides a current state table and indexed history as a default when you create a new ledger. Leveraging these allows customers to, as the names suggest, view the current states of documents and seek out specific documents and their revisions easily. You can learn more about Amazon QLDB by visiting the AWS page: `https://aws.amazon.com/qldb/`.

Sometimes you need a database for large-scale applications that need fast read and write performance, which a wide-column store can achieve. Let's learn more about it.

Wide-column store databases

Wide-column databases can sometimes be referred to as column family databases. A wide-column database is a NoSQL database that can store petabyte-scale amounts of data. Its architecture relies on persistent, sparse matrix, multi-dimensional mapping using a tabular format. Wide-column databases are generally not relational.

When is it a good idea to use wide-column databases?

- Sensor logs and IoT information
- Geolocation data
- User preferences
- Reporting
- Time-series data
- Logging applications
- Many inserts, but not many updates
- Low latency requirements

When are wide-column databases not a good fit? They are good when the use case calls for ad hoc queries:

- Heavy requirement for joins
- High-level aggregations
- Requirements change frequently
- OLTP uses cases

Apache Cassandra is probably the most popular wide-column store implementation today. Its architecture allows deployments without single points of failure. It can be deployed across clusters and data centers.

Amazon Keyspaces (formerly Amazon Managed Apache Cassandra Service, or Amazon MCS) is a fully managed service that allows users to deploy Cassandra workloads. Let's learn more about it.

Amazon Keyspaces (for Apache Cassandra)

Amazon Keyspaces (formerly known as Amazon Cassandra) is a fully managed, scalable, and highly available NoSQL database service. NoSQL databases are a type of database that does not use the traditional table-based relational database model and is well suited for applications that require fast, scalable access to large amounts of data.

Keyspaces is based on Apache Cassandra, an open-source NoSQL database that is widely used for applications that require high performance, scalability, and availability. Keyspaces provides the same functionality as Cassandra, with the added benefits of being fully managed and integrated with other AWS services.

Keyspaces supports both table and **Cassandra Query Language** (**CQL**) APIs, making it easy to migrate existing Cassandra applications to Keyspaces. It also provides built-in security features, such as encryption at rest and network isolation, using Amazon VPC and integrates seamlessly with other AWS services, such as Amazon EMR and Amazon SageMaker.

Servers are automatically spun up or brought down, and, as such, users are only charged for the servers Cassandra is using at any one time. Since AWS manages it, users of the service never have to provision, patch, or manage servers, and they don't have to install, configure, or tune software. Cassandra in AWS can be configured to support thousands of user requests per second.

Cassandra is a NoSQL database; as expected, it doesn't support SQL. The query language for Cassandra is the CQL. The quickest method to interact with Apache Cassandra is using the CQL shell, which is called **cqlsh**. With cqlsh, you can create tables, insert data into the tables, and access the data via queries, among other operations.

Keyspaces supports the Cassandra CQL API. Because of this, the current code and drivers developed in Cassandra will work without changing the code. Using Amazon Keyspaces instead of just Apache Cassandra is as easy as modifying your database endpoint to point to an Amazon MCS service table.

In addition to Keyspaces being wide-column, the major difference from DynamoDB is that it supports composite partition keys and multiple clustering keys, which are not available in DynamoDB. However, DynamoDB has better connectivity with other AWS services, such as Athena, Kinesis, and Elasticsearch.

Keyspaces provides an SLA for 99.99% availability within an AWS Region. Encryption is enabled by default for tables, and tables are replicated three times in multiple AWS Availability Zones to ensure high availability. You can create continuous backups of tables with hundreds of terabytes of data with no effect on your application's performance, and recover data to any point in time within the last 35 days. You can learn more about Amazon Keyspaces by visiting the AWS page: `https://aws.amazon.com/keyspaces/`.

Benefits of AWS database services

In the new world of cloud-born applications, a one-size-fits-all database model no longer works. Modern organizations will not only use multiple types of databases for multiple applications, but many will use multiple types of databases in a single application. To get more value from data, you can choose the following three options available in AWS based on your workload.

Moving to fully managed database services

Managing and scaling databases in a legacy infrastructure, whether on-premises or self-managed in the cloud (on EC2), can be a tedious, time-consuming, and costly process. You have to worry about operational efficiency issues such as:

- The process of installing hardware and software, configuring settings, patching, and creating backups can be time-consuming and laborious
- Performance and availability issues
- Scalability issues, such as capacity planning and scaling clusters for computing and storage
- Security and compliance issues, such as network isolation, encryption, and compliance programs, including PCI, HIPAA, FedRAMP, ISO, and SOC

Instead of dealing with the challenges mentioned above, you would rather spend your time innovating and creating new applications instead of managing infrastructure. With AWS-managed databases, you can avoid the need to over- or under-provision infrastructure to accommodate application growth, spikes, and performance requirements, as well as the associated fixed capital costs such as software licensing, hardware refresh, and maintenance resources. AWS manages everything for you, so you can focus on innovation and application development, rather than infrastructure management. You won't need to worry about administrative tasks such as server provisioning, patching, setup, configuration, backups, or recovery. AWS continuously monitors your clusters to ensure your workloads are running with self-healing storage and automated scaling, allowing you to focus on higher-value tasks such as schema design and query optimization.

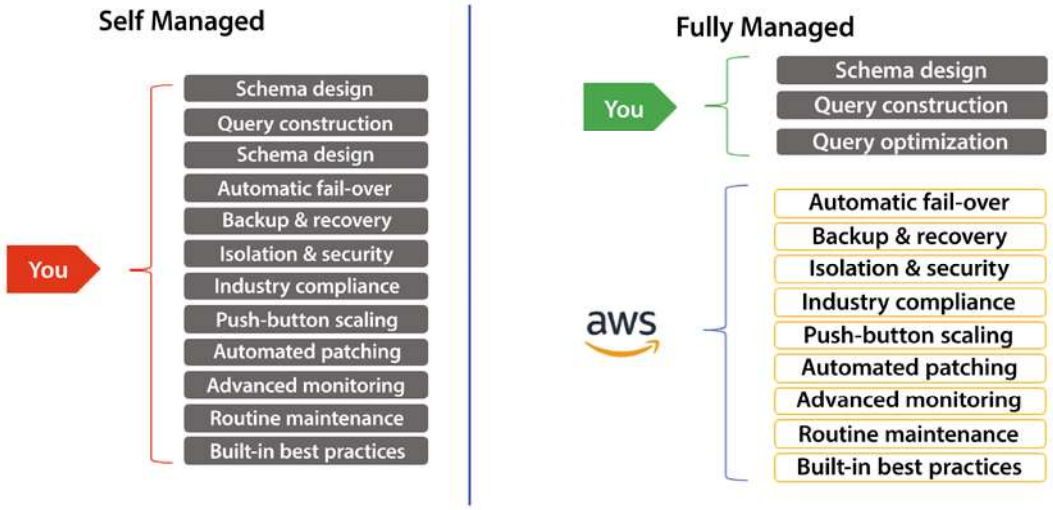

Figure 7.7: Fully managed database services on AWS

Let's look at the next benefit of using a purpose-built database. With purpose-built databases, your application doesn't need a one-fit-for-all architecture and it doesn't have to be molded to accommodate a decade-old relational database. Purpose-built databases help you to achieve maximum output and performance as per the nature of your application.

Building modern applications with purpose-built databases

In the 60s and 70s, mainframes were the primary means of building applications, but by the 80s, client-server architecture was introduced and significantly changed application development. Applications became more distributed, but the underlying data model remained mostly structured, and the database often functioned as a monolith. With the advent of the internet in the 90s, three-tier application architecture emerged. Although client and application code became more distributed, the underlying data model continued to be mainly structured, and the database remained a monolith. For nearly three decades, developers typically built applications against a single database. And that is an interesting data point because if you have been in the industry for a while, you often bump into folks whose mental model is, "Hey, I've been building apps for a long time, and it's always against this one database."

But what has changed? Fast forward to today, and microservice architectures are how applications are built in the cloud. Microservices have now extended to databases, providing developers with the ability to break down larger applications into smaller, specialized services that cater to specific tasks.

This allows developers to choose the best tool for the job instead of relying on a single, overworked database with a single storage and compute engine that struggles to handle every access pattern.

Today's developers and end-users have different expectations compared to the past. They require lower latency and the ability to handle millions of transactions per second with many concurrent users. As a result, data management systems have evolved to include specialized storage and compute layers optimized for specific use cases and workloads. This allows developers to avoid trade-offs between functionality, performance, and scale.

Plus, what we've seen over the last few years is that more and more companies are hiring technical talent in-house to take advantage of the enormous wave of technological innovation that the cloud provides. These developers are building not in the monolithic ways of the past but with microservices, where they compose the different elements together using the right tool for the right job. This has led to the highly distributed, loosely coupled application architectures we see powering the most demanding workloads in the cloud.

Many factors contribute to the performance, scale, and availability requirements of modern apps:

- **Users** – User growth is a common KPI for businesses, and the cloud's global reach enables businesses to touch millions of new customers.
- **Data volume** – Businesses are capturing more data about their customers to either increase the value of their existing products or sell them new products. This results in terabyte- or even petabyte-scale data.
- **Locality** – Businesses are often expanding their presence into new markets to reach new users, which complicates the architectures of their solutions/products.
- **Performance** – Businesses don't want to dilute the user experience to reach new markets or grow their customer base. If customers find a better alternative, they'll use it.
- **Request rate** – As businesses use the cloud's global reach to develop more interactive user experiences in more markets, they need their apps and databases to handle unprecedented levels of throughput.
- **Access/scale** – As businesses embark on their digital transformations, these scale measures are compounded by the number of devices that access their applications. There are billions of smartphones worldwide, and businesses connect smartphones, cars, manufacturing plants, devices in our homes, and more to the cloud. This means many, many billions of devices are connected to the cloud.

- **Economics**—Businesses can't invest millions of dollars in hardware and licenses up front and hope they'll succeed, that model is unrealistic in 2023. Instead, they have to hedge their success by only paying for what they use, without capping how much they can grow. These dynamics combined have changed how businesses build applications. These modern applications have wildly different database requirements, which are more advanced and nuanced than simply running everything in a relational database.

The traditional approach of using a relational database as the sole data store for an application is no longer sufficient. To address this, developers are leveraging their expertise in breaking down complex applications into smaller components and choosing the most appropriate tool for each task. This results in well-architected applications that can scale effectively. The optimal tool for a given task often varies by use case, leading developers to build highly distributed applications using multiple specialized databases.

Now you have learned about the different types of AWS databases, let's go into more detail about moving on from legacy databases.

Moving on from legacy databases

Numerous legacy applications have been developed on conventional databases, and consumers have had to grapple with database providers that are expensive, proprietary, and impose punishing licensing terms and frequent audits. Oracle, for instance, announced that they would double licensing fees if their software is run on AWS or Microsoft. As a result, customers are attempting to switch as soon as possible to open-source databases such as MySQL, PostgreSQL, and MariaDB.

Customers who are migrating to open-source databases are seeking to strike a balance between the pricing, freedom, and flexibility of open-source databases and the performance of commercial-grade databases. Achieving the same level of performance on open-source databases as on commercial-grade databases can be challenging and necessitates a lot of fine-tuning.

AWS introduced Amazon Aurora, a cloud-native relational database that is compatible with MySQL and PostgreSQL to address this need. Aurora aims to provide a balance between the performance and availability of high-end commercial databases and the simplicity and cost-effectiveness of open-source databases. It boasts 5 times better performance than standard MySQL and 3 times better performance than standard PostgreSQL, while maintaining the security, availability, and reliability of commercial-grade databases, all at a fraction of the cost. Additionally, customers can migrate their workloads to other AWS services, such as DynamoDB, to achieve application scalability.

In this section, you learned about the benefits of AWS database services. Now, there are so many database services that you have learned about, so let's put them together and learn how to choose the right database.

Choosing the right tool for the job

In the previous sections, you learned how to classify databases and the different database services that AWS provides. In a nutshell, you learned about the following database services under different categories:

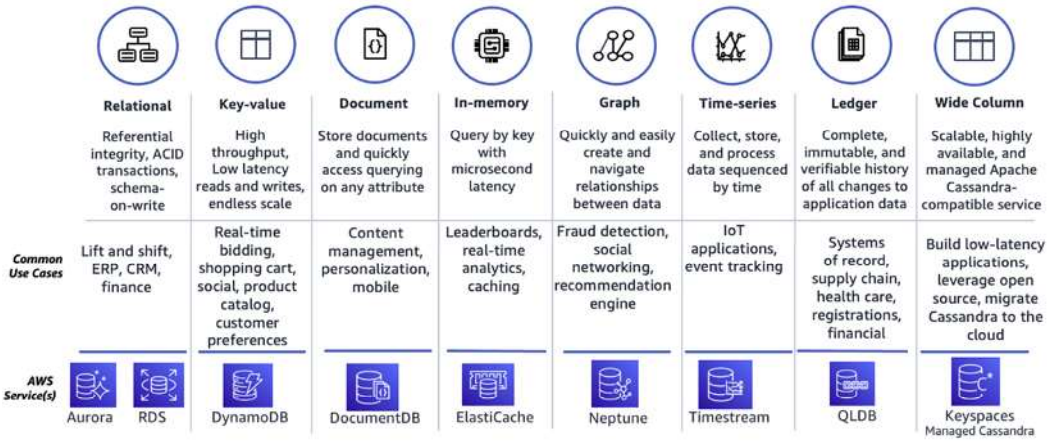

Figure 7.8: AWS database services in a nutshell

When you think about the collection of databases shown in the preceding diagram, you may think, "Oh, no. You don't need that many databases. I have a relational database, and it can take care of all this for you". Swiss Army knives are hardly the best solution for anything other than the most straightforward task. If you want the right tool for the right job that gives you the expected performance, productivity, and customer experience, you want a unified purpose-built database. So no one tool rules the world, and you should have the right tool for the right job to make you spend less money, be more productive, and change the customer experience.

Consider focusing on common database categories to choose the right database instead of browsing through hundreds of different databases. One such category is 'relational,' which many people are familiar with. Suppose you have a workload where strong consistency is crucial, where you will collaborate with the team to define schemas, and you need to figure out every single query that will be asked of the data and require consistent answers. In that case, a relational database is a good fit.

Popular options for this category include Amazon Aurora, Amazon RDS, open-source engines like PostgreSQL, MySQL, and MariaDB, as well as RDS commercial engines such as SQL Server and Oracle Database.

AWS has developed several purpose-built non-relational databases to facilitate the evolution of application development. For instance, in the key-value category, Amazon DynamoDB is a database that provides optimal performance for running key-value pairs at a single-digit millisecond latency and at a large scale. On the other hand, if you require a flexible method for storing and querying data in the same document model used in your application code, then Amazon DocumentDB is a suitable choice. This document database is designed to handle JSON documents in their natural format efficiently and can be scaled effortlessly.

Do you recall the era of XML? XML 1.0 was established in 1998. Commercial systems then added an XML data type to become an XML database. However, this approach had limitations, as many database operators needed help working with that data type. Today, document databases have replaced XML databases. Amazon DocumentDB, launched in January 2019, is an excellent example of such a database.

If your application requires faster response times than single-digit millisecond latency, consider an in-memory database and cache that can access data in microseconds. Amazon ElastiCache offers management for Redis and Memcached, making it possible to retrieve data rapidly for real-time processing use cases such as messaging, and real-time geospatial data such as drive distance.

Suppose you have large datasets with many connections between them. For instance, a sports company should link its athletes with its followers and provide personalized recommendations based on the interests of millions of users. Managing all these connections and providing fast queries can be challenging with traditional relational databases. In this case, you can use Amazon Neptune, a graph database designed to efficiently handle complex queries with interconnected data.

Time-series data is not just a timestamp or a data type that you might use in a relational database. Instead, a time-series database's core feature is that the primary axis of the data model is time. This allows for the optimization of data storage, scaling, and retrieval. Amazon Timestream is an example of a purpose-built time-series database that provides fast and scalable querying of time-series data.

Amazon QLDB is a fully managed ledger database service. A ledger is a type of database that is used to store and track transactions and is typically characterized by its immutability and the ability to append data in sequential order.

QLDB is a transactional database that uses an immutable, append-only ledger to store data. This means that once data is written to the ledger, it cannot be changed, and new data can only be added in sequential order. This makes QLDB well suited for applications that require a transparent, auditable, and verifiable record of transactions.

A wide-column database is an excellent choice for applications that require fast data processing with low latency, such as industrial equipment maintenance, trade monitoring, fleet management, and route optimization. Amazon Keyspaces for Apache Cassandra provides a wide-column database option that allows you to develop applications that can handle thousands of requests per second with practically unlimited throughput and storage.

You should spend a significant amount of time clearly articulating the business problem you are trying to solve. Some of the questions the requirements should answer are as follows:

- How many users are expected?
- How many transactions per day will occur?
- How many records need to be stored?
- Will there be more writes or reads?
- How will the data need to be accessed (only by primary key, by filtering, or some other way)?

Why are these questions important? SQL has served us well for several decades now, as it is pervasive, and has a lot of mindshare. So, why would we use anything else? The answer is performance. In instances where there is a lot of data and it needs to be accessed quickly, NoSQL databases might be a better solution. SQL vendors realize this and are constantly trying to improve their offerings to better compete with NoSQL, including adopting techniques from the NoSQL world. For example, Aurora is a SQL service, and it now offers Aurora Serverless, taking a page out of the NoSQL playbook.

As services get better, the line between NoSQL and SQL databases keeps on blurring, making the decision about what service to use more and more difficult. Depending on your project, you might want to draw up a Proof of Concept using a couple of options to determine which option performs better and fits your needs better.

Another reason to choose SQL or NoSQL might be the feature offered by NoSQL to create schema-less databases. Creating databases without a schema allows for fast prototyping and flexibility. However, tread carefully. Not having a schema might come at a high price.

Allowing users to enter records without the benefit of a schema may lead to inconsistent data, which becomes too variable and creates more problems than it solves. Just because we can create databases without a schema in a NoSQL environment, we should not forgo validation checks before creating a record. If possible, a validation scheme should be implemented, even when using a NoSQL option.

It is true that going schema-less increases implementation agility during the data ingestion phase. However, it increases complexity during the data access phase. So, make your choice by making a required trade-off between data context vs. data performance.

Migrating databases to AWS

If you find it challenging to maintain your relational databases as they scale, consider switching to a managed database service such as Amazon RDS or Amazon Aurora. With these services, you can migrate your workloads and applications without the need to redesign your application, and you can continue to utilize your current database skills.

Consider moving to a managed relational database if:

- Your database is currently hosted on-premises or in EC2.
- You want to reduce the burden of database administration and allocate DBA resources to application-centric work.
- You prefer not to rearchitect your application and wish to use the same skill sets in the cloud.
- You need a straightforward path to a managed service in the cloud for database workloads.
- You require improved performance, availability, scalability, and security.

Self-managed databases like Oracle, SQL Server, MySQL, PostgreSQL, and MariaDB can be migrated to Amazon RDS using the lift and shift approach. For better performance and availability, MySQL and PostgreSQL databases can be moved to Amazon Aurora, which offers 3-5 times better throughput. Non-relational databases like MongoDB and Redis are popularly used for document and in-memory databases in use cases like content management, personalization, mobile apps, catalogs, and real-time use cases such as caching, gaming leaderboards, and session stores. To maintain non-relational databases at scale, organizations can move to a managed database service like Amazon DocumentDB for self-managed MongoDB databases or Amazon ElastiCache for self-managed in-memory databases like Redis. These services provide a straightforward solution to manage the databases without rearchitecting the application and enable the same DB skill sets to be leveraged while migrating workloads and applications.

As you understand the different choices of databases, then the question comes of how to migrate your database to the cloud; there are five types of database migration paths available in AWS:

- **Self-service** - For many migrations, the self-service path using the DMS and **Schema Conversion Tool (SCT)** offers the tools necessary to execute with over 250,000 migrations completed through DMS, customers have successfully migrated their instances to AWS. Using the **Database Migration Service (DMS)**, you can make homogeneous migrations from your legacy database service to a managed service on AWS, such as from Oracle to RDS Oracle. Alternatively, by leveraging DMS and the SCT, heterogeneous conversions are possible, such as converting from SQL Server to Amazon Aurora. The **Schema Conversion Tool (SCT)** assesses the source compatibility and recommends the best target engine.
- **Commercially licensed to aws databases** - This type of migration is best for customers looking to move away from the licensing costs of commercial database vendors and avoid vendor lock-in. Most of these migrations have been from Oracle and SQL Server to open-source databases and Aurora, but there are use cases for migrating to NoSQL databases as well. For example, an online store may have started on a commercial or open-source database but now is growing so fast that it would need a NoSQL database like DynamoDB to scale to millions of transactions per minute. Refactoring, however, typically requires application changes and takes more time to migrate than the other migration methods. AWS provides a Database Freedom program to assist with such migration. You can learn more about the AWS Database Freedom program by visiting the AWS page: `https://aws.amazon.com/solutions/databasemigrations/database-freedom/`.
- **MySQL Database Migrations** - Standard MySQL import and export tools can be used for MySQL database migrations to Amazon Aurora. Additionally, you can create a new Amazon Aurora database from an Amazon RDS for MySQL database snapshot with ease. Migration operations based on DB snapshots typically take less than an hour, although the duration may vary depending on the amount and format of data being migrated.
- **PostgreSQL Database Migrations** - For PostgreSQL database migrations, standard PostgreSQL import and export tools such as `pg_dump` and `pg_restore` can be used with Amazon Aurora. Amazon Aurora also supports snapshot imports from Amazon RDS for PostgreSQL, and replication with AWS DMS.

- **The AWS Data Lab** is a service that helps customers choose their platform and understand the differences between self-managed and managed services. It involves a 4-day intensive engagement between the customer and AWS database service teams, supported by AWS solutions architecture resources, to create an actionable deliverable that accelerates the customer's use and success with database services. Customers work directly with Data Lab architects and each service's product managers and engineers. At the end of a Lab, the customer will have a working prototype of a solution that they can put into production at an accelerated rate.

  A Data Lab is a mutual commitment between a customer and AWS. Each party dedicates key personnel for an intensive joint engagement, where potential solutions will be evaluated, architected, documented, tested, and validated. The joint team will work for four days to create usable deliverables to enable the customer to accelerate the deployment of large AWS projects. After the Lab, the teams remain in communication until the projects are successfully implemented.

In addition to the above, AWS has an extensive Partner Network of consulting and software vendor partners who can provide expertise and tools to migrate your data to AWS.

Summary

In this chapter, you learned about many of the database options available in AWS. You started by revisiting a brief history of databases and innovation trends led by data. After that, you explored the database consistency model learning ACID vs. BASE and OLTP vs. OLAP.

You further explored different types of databases and when it's appropriate to use each one, and you learned about the benefits of AWS databases and the migration approach. There are multiple database choices available in AWS, and you learned about making a choice to use the right database service for your workload.

In the next chapter, you will learn about AWS's services for cloud security and monitoring.

9

Driving Efficiency with CloudOps

Organizations migrating to the cloud need management and governance to ensure best practices in their cloud IT operations. Whether you manage modern applications built using microservices-based architectures, containers, serverless stacks, or legacy applications that have been re-hosted or re-architected for the cloud, you will realize that traditional application development and operations processes could be more effective.

Automation has always been vital for managing cloud operations, increasing efficiency and avoiding human error disruption. However, you will still observe many manual tasks in most organizations, especially IT workload management. With the rise of the cloud, automation has become more critical due to its pay-as-you-go model. Automation helps you improve productivity, resulting in substantial cost savings in human effort and IT resource expenditure. Automation has become key to reducing daily operational costs and cloud organizations spending more on operations than upfront capital investment.

Automation is crucial for cost and other aspects, such as enduring application security and reliability. Automation goes hand in hand with monitoring and alerts. Automation will only work if you have proper monitoring, alerting your automation script when to take a certain action, for example, if you want to run your production app server without compromising the user experience due to a capacity crunch. It's always recommended to monitor server capacity, such as if the server exhausted memory capacity to 80% or CPU capacity to 70%, and send an alert to autoscaling for server scaling as needed.

AWS provides several services and tools to automate your cloud infrastructure and application with CloudOps fully or if you want to automate security using DevSecOps. In this chapter, you will learn the following topics in detail to understand the need for cloud automation, monitoring, and alerts:

- What is CloudOps?
- AWS CloudOps pillars
- DevOps and DevSecOps in AWS
- AWS cloud management tools for automation
- Cloud automation best practice

By the end of this chapter, you will understand various automation strategies to manage cloud operations. You will learn about various AWS services available at your disposal for cloud automation, monitoring, and alerts and how to use them in your workload.

What is the cloud operation (CloudOps) model, and what role does automation play?

CloudOps, or the cloud operational model, encompasses a collection of guidelines and safeguards that are established, tracked, and adjusted as needed to manage expenses, boost productivity, and mitigate potential security risks. It can help guide your people, processes, and the technology associated with your cloud infrastructure, security, and operations. An operational model also helps you develop and implement controls to manage security, budget, and compliance across your workloads in the cloud.

Implementing cloud automation empowers organizations to construct streamlined cloud operational models through the automated creation, modification, and removal of cloud resources. Although the concept of cloud computing initially promised the ability to utilize services as required, many organizations still rely on manual processes to provision resources, conduct testing, recognize resource redundancy, and decommission resources. This approach can result in substantial labor, error-proneness, and expense.

Businesses migrating to the cloud need management and governance to ensure best practices in their cloud IT operations. AWS management and governance services provide faster innovation and firm control over cost, compliance, and security.

The following are the key benefits of the cloud operation model:

- Organizations can unlock the speed and agility that comes with the cloud and accelerate their cloud adoption and application modernization efforts as part of their digital transformation journey.
- Use the power of automation for routine tasks to reduce manual errors and interventions.
- Continue to scale your businesses with the certainty that cloud governance spans all different environments uniformly and at scale.
- Use your skilled personnel effectively to deliver business outcomes.
- Avoid unexpected cost overruns.

While implementing cloud automation can initially demand significant effort, the potential rewards are considerable. Once the initial hurdles are overcome, the ability to perform intricate tasks with just a single click can transform an organization's operations. In addition to minimizing manual labor, cloud automation offers several other advantages, including:

- **Improved security and resilience**: Automation helps you improve security as there are always chances of security lapses due to human error or changes that can be left out, which is outside of individual knowledge for setting up security credentials for newly added dev environment. Also, automation helps to improve resiliency by automated recovery of the environment; for example, if your server reaches over capacity, you can take proactive action by adding more CPU or memory to avoid downtime.
- **Improved backup processes**: Automated backup is one of the most important things for your business process continuation. Protecting your data helps minimize business loss by winning customer trust and rebuilding your environment quickly in case of a disaster recovery event. Automated backup helps you ensure all backups are secure and do not rely on one individual who can forget to take a backup.
- **Improved governance**: Automation helps you to improve governance by making sure all activity is captured across the environment. For example, you need to know what servers and database inventories are running across your company's IT workload or who is accessing those environments. All these things are possible through an automated governance model.

AWS provides a set of services and third-party tools for modern enterprises as they adopt the cloud operation model. It can help you drive more innovation, faster cloud adoption, improved application performance, and quicker response times to customer feedback while maintaining governance and compliance.

Let's look at the CloudOps pillars and the services provided to fulfill the requirements of each pillar.

CloudOps pillars

While planning your CloudOps model, you need to take a 360-degree look. You want to provision and operate your environment for business agility and governance control. Establishing a CloudOps model, regardless of your cloud migration journey, helps you attain consistent governance and efficient operations across different infrastructure environments. This helps free up critical resources to deliver business outcomes and time-to-market faster while improving safety, ease, efficiency, and cost control. The following diagram shows the key pillars of cloud operation for complete coverage of your IT workload automation.

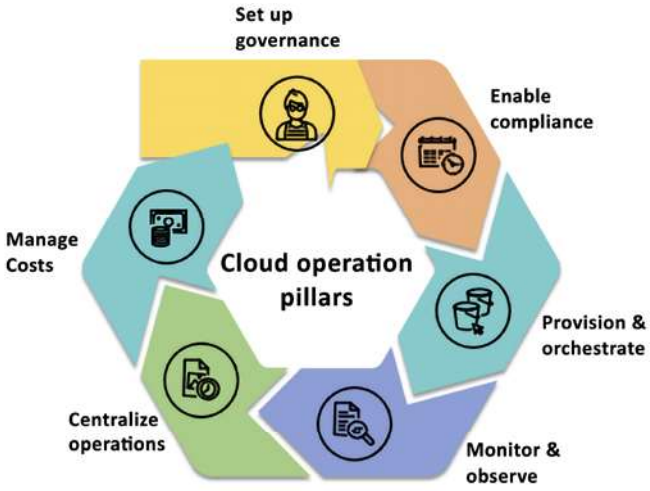

Figure 9.1: The pillars of CloudOps

As shown in the preceding diagram, the following are the key pillars of CloudOps:

1. **Set up Governance**: Set up a well-architected, multi-account AWS environment with guardrails to build the foundation for governance. AWS environments with a Well-Architected Framework checklist ensure you have covered all best practices to monitor and set alerts on your environment for security, operational excellence, cost, reliability, and performance.
2. **Enable Compliance**: Continuously monitor compliance and configurations for your resources, remediate failures in an automated fashion, and gather evidence for audits.
3. **Provision & Orchestrate**: Speed up application and resource provisioning with **infrastructure-as-code (IaC)** while maintaining consistency and compliance.

4. **Monitor & Observe**: Measure and manage your applications and resources to identify and resolve issues quickly.
5. **Centralize Operations**: Take seamless and automated operational actions across your entire application portfolio while maintaining safety, security, and compliance.
6. **Manage Costs**: Helps manage costs with transparency, control, regular forecasting, and optimization, thus enabling businesses to achieve greater financial efficiency and transform their operations.

Let's look at what each of these pillars helps you to achieve in your efforts to enable governance in cloud environments. You will also learn the AWS services that predominantly fulfill the requirement of each pillar.

First pillar – Set up governance

The first and the best place to start is by laying a very strong foundation for your governance. In the AWS environment, it begins by setting up a well-architected, multi-account AWS environment and setting up guardrails in each account. You learned about the Well-Architected Framework in *Chapter 2*, *Understanding the AWS Well-Architected Framework and Getting Certified*, where you saw that AWS has a comprehensive checklist to make sure your environment is set up properly to monitor cost, security, performance, reliability, and high availability. You can refer to AWS's well-architected labs here at `https://www.wellarchitectedlabs.com/`, which provide a very comprehensive, practical, hands-on guide to enable those guardrails against each well-architected pillar. The environment you build must be secure and extensible so that you don't halt experimentation and innovation as you grow your footprint on AWS. You need it to scale with your usage.

Your business needs to evolve continuously, so you should keep yourself from a single mode of architecting and operating in AWS. Most customers' environments don't remain static. They tend to grow with their business. You want to ensure your landing zone grows with your business without encumbrance while adhering to organizational policies. AWS Landing Zone is an offering that assists clients in swiftly configuring a secure and multi-account AWS environment, built around AWS's industry-leading practices. The solution delivers a preconfigured and secure infrastructure that encompasses key services, standardized AWS account architecture, and robust security controls. The goal of Landing Zone is to provide a secure, well-architected multi-account environment that serves as a starting point for new AWS accounts. It helps customers get started with AWS faster by providing a set of reusable blueprints for common patterns and practices, such as account VPCs, security controls, and identity and access management.

You need a well-defined AWS environment to accommodate the following needs:

- **Many Teams**: Multiple teams could be in the same account, overstepping one another.
- **Isolation**: Each team could have different security needs and want to isolate themselves from one another with a different security profile.
- **Security Controls**: Different applications might have different controls around them to address security and compliance. For example, talking to an auditor is far easier than pointing to a single account hosting the PCI solution. But even within an organization, security controls provide the ability to isolate certain things based on security isolation needs.
- **Business Process**: There are completely different **business units (BUs)** or products. For example, the Sales BU is different from the HR BU with an entirely different business process.
- **Billing**: An account is the primary way to divide items at a billing level. Each AWS account is billed separately and has its own set of resources and associated charges. This means that if you have multiple accounts within an organization, each account will have its own billing and cost allocation data.

To enable account control, AWS provides **AWS Organizations** that help you to establish a multi-account structure for centralized governance. You can use it to establish granular control over your AWS accounts, manage across accounts easily, and apply policies as broadly or as narrowly as you need. In the previous chapter, *Chapter 8*, *Best Practices for Application Security, Identity, and Compliance*, you learned about AWS Organizations.

For automated setup, AWS provides **AWS Control Tower**, a self-service solution to set up and govern a secure, compliant multi-account AWS environment. It abstracts multiple AWS services under the covers, so you can use it to set up your environment, based on best practices, without needing a lot of AWS knowledge. AWS Control Tower provides the following benefits:

- Automate the setup of your landing zone based on best-practice blueprints
- Apply guardrails for ongoing governance over your AWS workloads
- Automate your account provisioning workflow with an account factory
- Get dashboard visibility into your organizational units, accounts, and guardrails

You learned about AWS Control Tower in the previous chapter, *Chapter 8*, *Best Practices for Application Security, Identity, and Compliance*.

AWS professional services provide AWS **Landing Zone Accelerator (LZA)**, which is a set of tools and resources provided by AWS to help customers accelerate the deployment of a secure, multi-account AWS environment.

LZA builds on top of the AWS Landing Zone service, which provides a pre-built, opinionated framework for setting up a secure, multi-account environment.

LZA provides a modular set of landing zone components that can be customized to meet specific requirements and leverages automation to speed up the deployment process. It also provides access to AWS experts who can provide guidance and best practices to help ensure a successful deployment. You can learn more about LZA by referring AWS user guide here: `https://aws.amazon.com/solutions/implementations/landing-zone-accelerator-on-aws/`.

LZA is designed to accelerate the process of setting up a landing zone for workloads that are migrating to AWS. LZA provides a modular set of landing zone components that can be customized to meet specific requirements and leverages automation to speed up the deployment process. This makes LZA a good fit for customers who need to set up a landing zone quickly and want more control over the individual components of their environment.

On the other hand, Control Tower is designed to help customers set up and govern a multi-account AWS environment. Control Tower provides a pre-built set of rules and policies to enforce governance and security best practices across multiple AWS accounts. CT also provides a central dashboard for managing and monitoring multiple accounts, making it easier for customers to maintain governance and compliance across their environment. This makes CT a good fit for customers who need to manage multiple AWS accounts and want a pre-built set of governance policies to enforce best practices.

It is extremely important to have the correct foundation when you are starting with your cloud journey. AWS services like Landing Zone and Control Tower, in combination with the Well-Architected Framework, help you with an automated way to establish the right environment. The teams building this out are not the bottleneck and enable you to be flexible in your approach and know that you might need new accounts, processes, or isolation solutions. That flexibility is what allows us to succeed in the long term. After setting up governance, you must ensure your applications meet compliance requirements. Let's learn more about automating compliance.

Second pillar — Managing Configuration, Compliance, and Audit

As you migrate workloads to the cloud, you need to know that you can maintain cloud compliance and get assurance for your workloads. Once compliance mechanisms and processes are in place, you can empower your development teams to build and innovate while having peace of mind that they are staying compliant.

A resource inventory helps you maintain environment configurations, track change history, depend on one another, and ensure your resources are correctly configured. Once you establish proper configurations, you want to be able to audit, manage, and remediate them quickly.

How many hours do you spend today collecting evidence in response to an audit? Whether internal or external? Wouldn't it be easier if you were able to keep a running log of all auditable events and remediation actions? And that's where AWS configuration, compliance, and auditing tools come in. You must continuously monitor configuration and compliance changes within your AWS environment and keep a running audit log to get visibility into your organization's resource configurations.

Many customers follow the **Institute of Internal Auditors (IIA)** guidance for the three lines of defense:

- **1st Line**: *How to automate compliance management and manage risk* – The implementation of AWS CloudTrail, AWS Config, AWS Control Tower, and AWS License Manager can aid in the automation of compliance management and risk mitigation within an AWS environment.
- **2nd Line**: *How to implement continuous oversight and oversee risk* – By utilizing Amazon CloudWatch and AWS Security Hub, it is possible to understand the operational health and security status of AWS accounts.
- **3rd Line**: *How to assess and independently gather assurance of risk management* – AWS Audit Manager helps assess their security, change management, and software licensing controls.

You learned about AWS Security Hub and Control Tower in the previous chapter, *Chapter 8*, *Best Practices for Application Security, Identity, and Compliance*. Let's learn about the other services, depending on mentioned above for managing cloud audits and compliance. You can use a combination of these services your IT workload needs, to automate configuration, compliance, and audit.

AWS CloudTrail

In AWS, all interactions with AWS services and resources are handled through AWS API calls, and these API calls are monitored and logged by AWS CloudTrail. AWS CloudTrail records all API calls made in your AWS account and provides a complete history of all user activity and API usage. This information can be used for security analysis, compliance auditing, and troubleshooting.

CloudTrail stores all generated log files in an Amazon S3 bucket that you define. These log files are encrypted using **Amazon S3 server-side encryption (SSE)**, which provides an additional layer of security for your logs. It's also worth noting that CloudTrail logs all API calls, regardless of whether they come directly from a user or on behalf of a user by an AWS service.

This lets you understand all API activity in your AWS environment, which can be crucial for security and compliance purposes.

AWS CloudTrail is a service that enables governance, compliance, operations, and risk auditing for AWS accounts by logging and monitoring account activity related to actions across the AWS infrastructure. By using CloudTrail, you can continuously monitor and retain logs of all account activity, providing you with a complete history of your AWS account's event history. This event history includes actions taken through the AWS Management Console, AWS SDKs, command-line tools, and other AWS services.

These logs can be used to aid in governance, compliance, and risk management, providing a clear record of activity across your AWS infrastructure. With CloudTrail, you can also create custom alerts and notifications to help you identify and respond to potential security issues and compliance risks in real time.

Once enabled, CloudTrail will automatically track all Management Events at no charge. Then, you also have several different data event sources you can opt into depending on your application and compliance needs. This event history is another source of observability data, simplifying security analysis, resource change tracking, and troubleshooting. You can learn more about AWS CloudTrail by visiting the AWS page here: `https://aws.amazon.com/cloudtrail/`.

AWS Config

You learned about AWS Config in *Chapter 5*, *Storage in AWS – Choosing the Right Tool for the Job*, in the context of S3. AWS Config records and evaluates your AWS resources configuration. AWS Config performs the following activities for AWS resources: record, evaluate, and visualize. Let's learn about these in more detail in the context of CloudOps:

Record

- **Configuration history of AWS resources**: AWS Config records the details of changes made to your AWS resources, providing you with a configuration history timeline. This enables you to track any changes made to a resource's configuration at any time in the past.
- **Resource relationship tracking**: AWS Config can discover, map, and track relationships between AWS resources in your account. For example, if a new **Amazon Elastic Compute Cloud** (**Amazon EC2**) security group is associated with an Amazon EC2 instance, AWS Config will record the updated configurations of both the Amazon EC2 security group and the Amazon EC2 instance.

- **Configuration history of software**: AWS Config can also record software configuration changes within your Amazon EC2 instances and servers running on-premises or with other cloud providers. It provides a history of both OS and system-level configuration changes and infrastructure configuration changes recorded for Amazon EC2 instances.

Evaluate

- **Configurable and customizable rules**: Assess your resource configurations and resource changes for compliance against built-in or custom rules and automate the remediation of non-compliant resources. You can customize pre-built rules provided by AWS Config or create your own custom rules with AWS Lambda to define your internal guidelines and best practices for resource configurations.
- **Conformance packs**: Simplifies organization-wide deployment and reporting of compliance. It deploys a pack of config rules and remediation actions to your AWS Organization.
- **Automatic remediation** enables you to remediate non-compliant resources using Systems Manager Automation documents.

Visualize

- **Cloud governance dashboard**: This feature provides a visual dashboard that lets you easily identify non-compliant resources and take the necessary corrective action. You can customize the dashboard to monitor resources based on cost and security.
- **Multi-account, multi-region data aggregation**: AWS Config allows you to aggregate data from multiple AWS accounts and regions, providing you with a centralized view of your resources and their compliance status with AWS Config rules. This feature is particularly useful for enterprise-scale organizations.
- **Configuration snapshots**: AWS Config can take snapshots of your resource configurations at specific points in time. This allows you to quickly identify changes to your resources and compare their configurations across different points in time.

Here is an example AWS Config rule that checks whether Amazon EC2 instances have an associated security group with inbound rules that allow traffic on port 22 (SSH):

```
{
  "Name": "ec2-security-group-has-inbound-rules-on-port-22",
  "Description": "Checks whether the security group associated with an EC2
instance has inbound rules that allow traffic on port 22",
```

```
  "Scope": {
    "ComplianceResourceTypes": [
      "AWS::EC2::Instance"
    ]
  },
  "Source": {
    "Owner": "AWS",
    "SourceIdentifier": "EC2_INSTANCE_HAS_SECURITY_GROUP_WITH_INBOUND_
RULES_ON_PORT_22"
  },
  "InputParameters": "{\"allowedProtocols\":\"tcp\",\"portNumber\":22}"
}
```

This rule checks whether the security group associated with each Amazon EC2 instance has inbound rules that allow traffic on port 22. If any instances do not have such a security group, they will be flagged as non-compliant.

AWS Config help to keep AWS resources compliant. You can learn more about AWS Config by visiting the AWS page here: `https://aws.amazon.com/config/`. Let's look at the next service for tracking and auditing licenses.

AWS License Manager

AWS License Manager is a one-stop solution for managing licenses from various software vendors across hybrid environments. This helps you to stay compliant within your organizational structure and processes. There are no additional charges for using AWS License Manager.

AWS License Manager is targeted at IT administrators who manage licenses and software assets. This includes license administrators, procurement administrators, or asset managers who are responsible for managing license true-ups and vendor audits. In contrast, users spin up instances and use the licensed software on those instances. With AWS License Manager, the administrators can now easily manage licenses. Users in the organization are not required to do additional work to manage licenses and can focus on business as usual.

You can complete your licensing true-ups and audits using AWS License Manager. Administrators start by creating rules based on their enterprise agreements. They can do this using the AWS Management Console, CLI, or API. Furthermore, administrators can enforce licensing rules by attaching them to instance launches. Once rules are enforced, the service automatically keeps track of instances as users spin them up and down.

The organization stays compliant based on its license terms, and administrators can discover users' software after spinning up instances. Finally, administrators can keep track of usage through the AWS License Manager's built-in dashboard.

AWS License Manager automatically keeps track of instance launches, and the built-in dashboard is populated. Administrators can view usage limit alerts and take actions such as procuring more licenses as needed. When there is an upcoming license true-up or audit, administrators no longer have to determine which instances use which licenses. With AWS License Manager's built-in dashboard, figuring out how many licenses they are using and which resources are using them is no longer a challenge. You can learn more about AWS License Manager by visiting the AWS page here: `https://aws.amazon.com/license-manager/`.

Amazon CloudWatch

CloudWatch is one of the essential services for running your cloud operation. It allows you to monitor your AWS workload and take action based on alerts. In addition, CloudWatch provides observability for your AWS resources on a single platform across applications and infrastructure.

Amazon CloudWatch is a powerful monitoring service that is designed to help you optimize your AWS resources and applications. It offers a wide range of capabilities, including:

- **Data and operational insights**: CloudWatch provides valuable insights into the performance and health of your AWS resources and applications. With CloudWatch, you can collect and track metrics, monitor log files, and set alarms.
- **Resource monitoring**: CloudWatch can monitor a variety of AWS resources, including Amazon EC2 instances, Amazon S3 buckets, and Amazon RDS instances. This allows you to quickly identify and troubleshoot any issues that arise.
- **Custom metrics**: CloudWatch allows you to create custom metrics based on the data generated by your applications. This provides you with greater flexibility and control over the monitoring process.
- **Log monitoring**: CloudWatch can also monitor the log files generated by your applications. This enables you to quickly identify and troubleshoot any issues that are related to your application code.

Amazon CloudWatch is an essential tool for anyone running applications on the AWS cloud platform. Its powerful monitoring capabilities can help you optimize your resources, improve application performance, and maintain the operational health of your systems.

CloudWatch alarms are a powerful feature that enable you to receive notifications or automate actions based on the rules you define. With CloudWatch alarms, you can monitor a wide range of metrics and set up alerts that notify you when certain conditions are met. For example, you can send an email alert to the admin whenever the average network latency of an Amazon RDS database exceeds 10 seconds or when the CPU usage of an Amazon EC2 instance falls below 10%. You can also create more complex alarms that automatically trigger actions, such as launching additional instances to handle increased traffic or scaling down resources during periods of low demand.

CloudWatch alarms provide a flexible and customizable way to monitor your AWS resources and take automated actions based on your specific needs. Whether you need to monitor resource utilization, application performance, or other key metrics, CloudWatch alarms can help you stay on top of your cloud infrastructure and ensure that it is always running at peak performance.

In addition, CloudWatch provides data for the past two weeks so that you can access historical data for analysis of past events. It also integrates with other AWS services, such as Amazon EC2 Auto Scaling, Amazon SNS, and AWS Lambda, enabling you to use CloudWatch to react to changes in your resources and applications.

Some key features of CloudWatch include the following:

- **Metrics**: CloudWatch allows you to collect metrics for your resources and applications, such as CPU usage, network traffic, and the disk reads/writes. You can view these metrics in the CloudWatch console or use the CloudWatch API to retrieve them programmatically.
- **Alarms**: You can set alarms in CloudWatch to be notified when certain thresholds are breached. For example, you can schedule an alarm to send an email or SMS message to you if the CPU usage on one of your Amazon EC2 instances exceeds a certain threshold.
- **Logs**: CloudWatch allows storing and accessing your log files in a centralized location. You can use CloudWatch Logs Insights to search and analyze your log data or use CloudWatch Logs to export your log data to third-party tools for further analysis.
- **Dashboards**: You can use CloudWatch dashboards to create custom views of your metrics and log data to quickly get an overview of your system's health and performance.

Amazon CloudWatch is a powerful and flexible monitoring service that can help you ensure the availability, performance, and efficiency of your AWS resources and applications.

Amazon CloudWatch Events is a service that allows you to respond to changes in state in your AWS resources in real time. With CloudWatch Events, you can monitor for operational changes in your resources and automatically trigger actions based on those changes. For example, you can create a CloudWatch Events rule that triggers an AWS Lambda function whenever a new Amazon EC2 instance is launched in your account. This Lambda function can perform actions like tagging the instance, configuring its security groups, or starting a set of preconfigured applications on the instance.

CloudWatch Events allows you to react to operational changes in your AWS resources quickly and efficiently. Automating your response to these events can save time and reduce the risk of manual errors. With CloudWatch Events, you can easily create rules that define the conditions that trigger your actions and specify the targets for executing those actions.

CloudWatch Events rules enable you to match event patterns and take actions in response to those patterns. A rule can have one or more event patterns, and you can specify the type of action that CloudWatch Events takes when it detects a pattern. For example, you can set up a rule to send an email message when a new Amazon EC2 instance is launched or to stop an Amazon EC2 instance when the CPU utilization is too high.

CloudWatch Events can be used to schedule automated actions that self-trigger at a specific time or when a specified event occurs. For example, you can use CloudWatch Events to schedule the automatic stopping of Amazon EC2 instances so that you don't incur charges for no longer needed instances. You can learn more about AWS CloudWatch by visiting the AWS page here: `https://aws.amazon.com/cloudwatch/`.

Amazon EventBridge

Amazon EventBridge is a fully-managed, serverless event bus service that simplifies connecting your applications, integrated SaaS applications, and AWS services. By creating event-driven architectures, EventBridge allows various applications and services to communicate with each other in a flexible and scalable manner.

The following screenshot shows an EventBridge rule set up to send information to Elasticsearch using Lambda for an instance start failure during autoscaling. This Lambda function then sends information to Elasticsearch, allowing you to analyze and troubleshoot the failure.

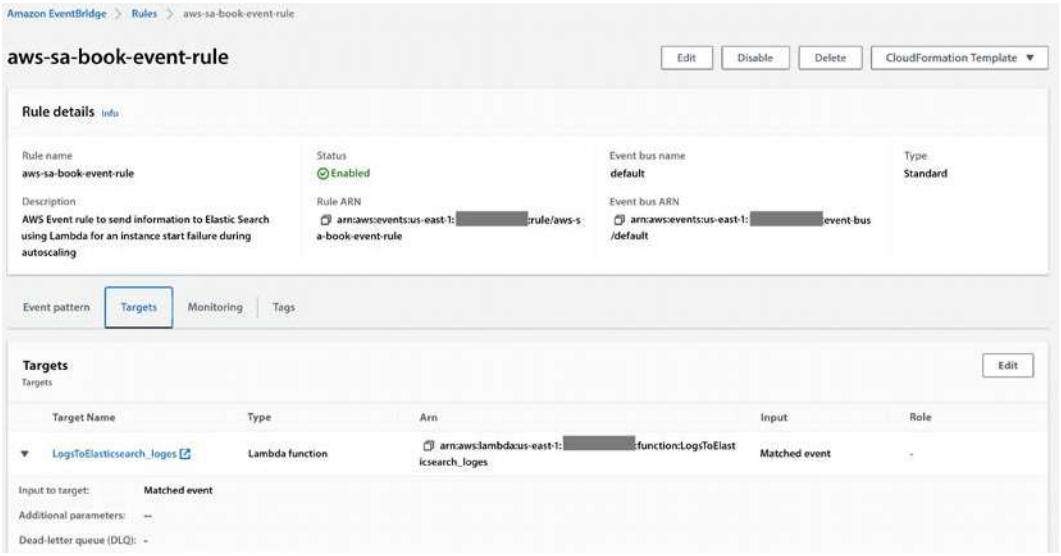

Figure 9.2: An AWS EventBridge rule for an autoscaling failure

EventBridge simplifies the setup and management of rules that dictate event routing and processing, enabling you to create complex and resilient architectures for your applications. With EventBridge, you can easily build event-driven applications and receive notifications about events from AWS DevOps services and automation application deployment pipelines. You can explore more about EventBridge by visiting the AWS page here: `https://aws.amazon.com/eventbridge/`.

AWS Audit Manager

The audit Manager continuously accesses risk and compliance controls and provides the following benefits.

- Easily map your AWS usage to controls: Use pre-built compliance frameworks or build custom frameworks to collect evidence for compliance controls.
- Save time with an automated collection of evidence across accounts: Focus on reviewing the relevant evidence to ensure your controls are working as intended.
- Be continually prepared to produce audit-ready reports: Evidence is continuously collected and organized by control so you can effortlessly search, filter, and review evidence to ensure controls are working as intended.
- Ensure assessment report and evidence integrity: When it is time for an audit, build assessment reports with evidence that has been continuously collected, securely stored, and remains unaltered.

Below you can find a screenshot of AWS Audit Manager:

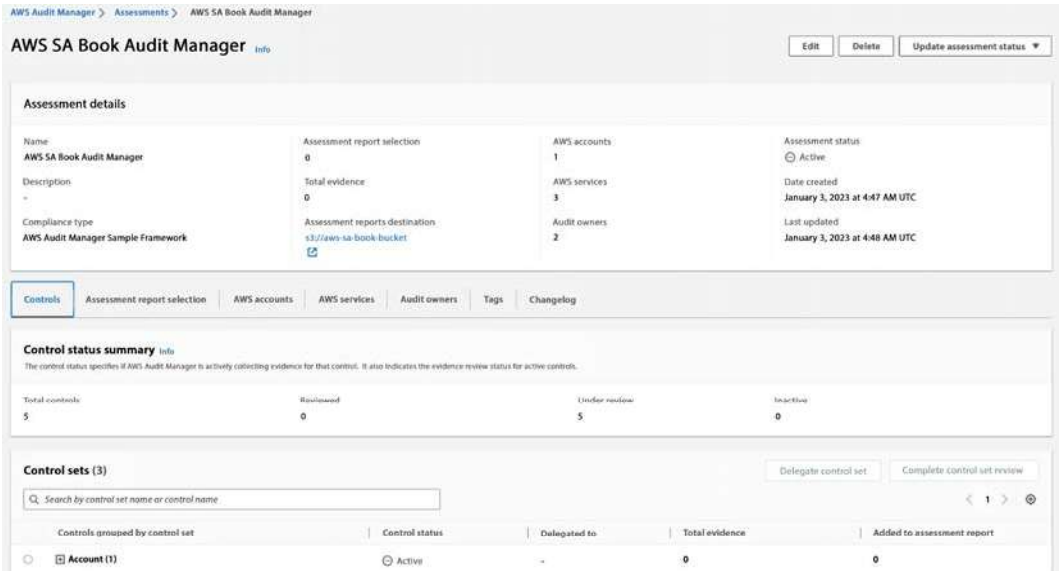

Figure 9.3: AWS Audit Manager

You can learn more about AWS Audit Manager by visiting the AWS page here: `https://aws.amazon.com/audit-manager/`.

AWS Systems Manager

AWS Systems Manager is a management service that allows you to take actions on your AWS resources as necessary. It provides you with a quick view of operational data for groups of resources, making it easy to detect any issues that could impact applications that rely on those resources. You can group resources by various criteria, such as applications, application layers, or production vs. development environments. Systems Manager displays operational data for your resource groups on a single dashboard, eliminating the need to switch between different AWS consoles. For example, you can create a resource group for an application that uses Amazon EC2, Amazon S3, and Amazon RDS. Systems Manager can check for software changes installed on your Amazon EC2 instances, changes in your S3 objects, or stopped database instances.

The following screenshot shows a configuration deployment status for servers in a given AWS account:

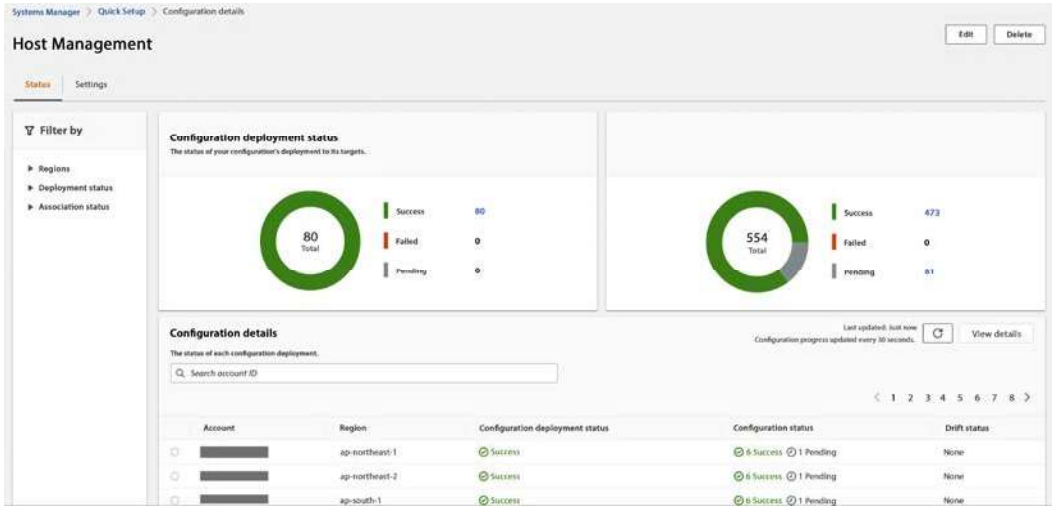

Figure 9.4: AWS Systems Manager host management

AWS Systems Manager provides detailed insights into the current state of your resource groups, allowing you to understand and control them quickly. The Systems Manager Explorer and Inventory dashboards offer various tools to view system configurations, such as operating system patch levels, software installations, and application configurations. Moreover, it is integrated with AWS Config, allowing you to track changes across your resources over time.

AWS Systems Manager offers several features to help maintain security and compliance in your environment. It can scan your instances against your patch, configuration, and custom policies, helping you identify and address potential security issues. With Systems Manager, you can define patch baselines, ensure that your anti-virus definitions are up-to-date, and enforce firewall policies, among other things. Systems Manager also enables you to manage your servers at scale remotely without manually logging in to each server. This feature can be especially helpful in large-scale environments, where managing resources individually can be time-consuming and error-prone.

In addition, Systems Manager provides a centralized store for managing your configuration data, including plain text items such as database strings and secrets like passwords. By separating your secrets and configuration data from your code, you can help reduce the risk of security breaches and simplify your development and deployment processes.

You can use System Manager to achieve all components of the second pillar as it provides a one-stop shop, as shown in the left-hand navigation bar in the below screenshot:

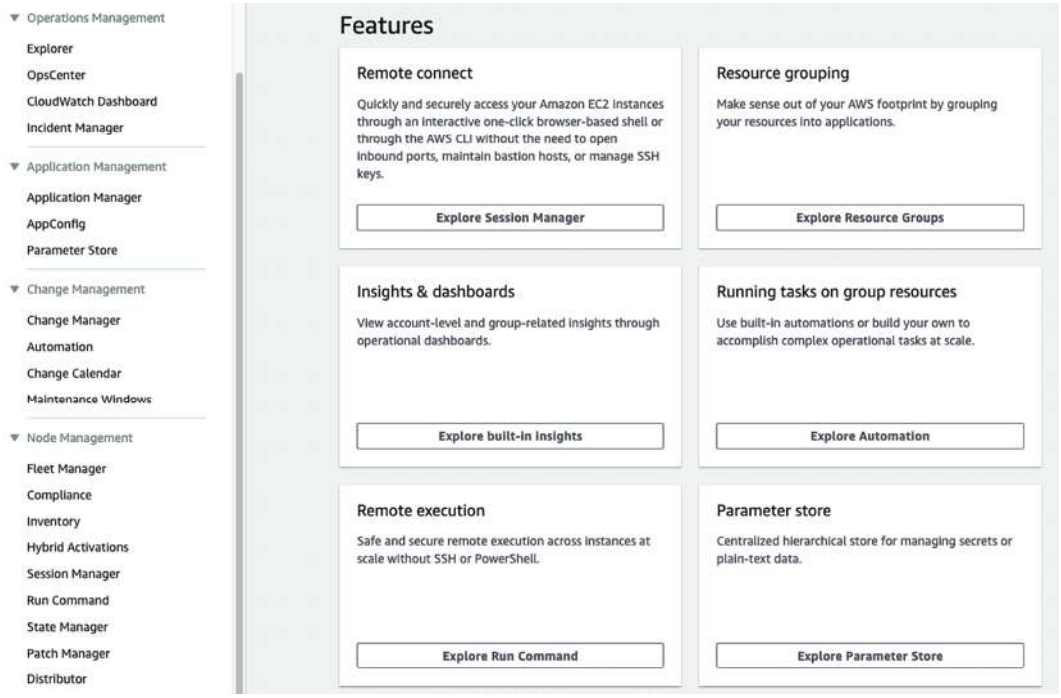

Figure 9.5: AWS Systems Manager for CloudOps monitoring and audit

AWS Systems Manager is the hub of your operation for managing all your AWS applications and resources along with your on-premise environments, keeping everything in one place for easy monitoring and auditing. AWS has done a great job explaining how Systems Manager works, which you can learn about by referring to this link: `https://docs.aws.amazon.com/systems-manager/latest/userguide/what-is-systems-manager.html`.

In this section, you learned about managing cloud configuration and compliance along with various AWS services that can help to achieve that. Now, let's learn about the following step: provision and orchestration.

Third pillar – Provisioning & orchestration

Once you have set up the environment, you need to speed up the provisioning and orchestration of your applications and any associated resources in a repeatable and immutable fashion.

Let's look at history; in the old days, infrastructure deployment started with manual deployments, where you used wikis and playbooks, which were sometimes outdated. Most of us can relate to a project where we had to use them. They were sometimes outdated, or some details needed to be mentioned, and we encountered situations during the provisioning which needed to be stated in the manual. The best way to solve the problem was to ask the person who did it the last few times in the past and hopes that this person was not on vacation and still working at the company!

The next step was scripting everything in Bash. It worked well until the complexity was too great because Bash was not designed to build complex deployment frameworks, so it was hard to maintain. The best advice was: it worked the last time, don't touch it!

As digital transformation increasingly occurs within an organization, more applications either move to or are built on the cloud. These applications themselves solve complex problems and require complex infrastructure. Teams need more tools to manage this complexity, be productive, and innovate. You need highly specialized tools to manage the applications and to be able to choose from a varied set of tools based on their use cases. Managing infrastructure is a big part of managing cloud complexity, and one of the ways to manage infrastructure is by treating infrastructure as code.

Infrastructure-as-code (**IaC**) templates help you to model and provision resources, whether AWS-native, third-party or open source, and applications on AWS and on-premises. So, you can speed up application and resource provisioning while improving consistency and compliance. For example, a developer wants to provision an S3 bucket that meets their company's security requirements. They will no longer have to dig through documentation to determine what bucket resource properties to set or how to set them. Instead, they can reuse a pre-built Secure S3 bucket module to provision a bucket quickly while automatically aligning with their company's requirements.

A cloud application typically consists of many components: networking (i.e., traffic gateways), compute (Amazon EC2, containers), databases, streams, security groups, users, roles, etc. All these servers and resources are the infrastructure components of your cloud application. Managing cloud applications involves managing the life cycle of their resources: create, update, or delete. By codifying your infrastructure, you can manage your infrastructure code in a way that is similar to your application code. This means that you can use a code editor to create it, store it in a version control system, and collaborate with your team members before deploying it to production. The advantages of using IaC are numerous, including:

- A single source of truth for deploying the entire stack.
- The ability to replicate, redeploy, and repurpose your infrastructure.

- The ability to version control both your infrastructure and your application code.
- Automatic rollback to the previous working state in case of failures.
- The ability to build and test your infrastructure as part of your CI/CD pipeline.

AWS offers multiple services that help customers manage their infrastructure. AWS CloudFormation is the provisioning engine. It helps speed up cloud provisioning with IaC.

AWS Service Catalog allows organizations to create and maintain a list of IT services authorized to be used on the AWS platform. These lists can include everything from VM images, servers, software, and databases to complete multi-tier application architectures. Users can then browse and launch these pre-approved IT services through a self-service portal, which helps ensure that they are using approved resources that meet compliance and security requirements. Additionally, administrators can set up workflows to automatically provision resources and control access and permissions to specific resources within the catalog.

You can choose to combine these services as per your workload needs. Let's learn about these services in more detail.

AWS CloudFormation

CloudFormation is a tool that supports the implementation of IaC. With CloudFormation, you can write your IaC using the CloudFormation template language, available in YAML and JSON formats. You can start from scratch or leverage any of the pre-existing sample templates to create your infrastructure. You can use the CloudFormation service through a web-based console, command-line tools, or APIs to create a stack based on your template code. Once you have defined your stack and resources in the template, CloudFormation provisions and configures them accordingly.

CloudFormation helps you model, provision, and manage AWS resources. It allows you to use a template to create and delete multiple related AWS resources in a predictable and consistent way. Here is a simple example of a CloudFormation template written in YAML syntax:

```
---
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-101013
      InstanceType: m4.xlarge
```

```
      KeyName: gen-key-pair
      SecurityGroups:
        - !Ref ServerSecurityGroup
  ServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow ssh access
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIp: 10.1.1.16/0
```

This template creates an Amazon EC2 instance with a security group that allows incoming SSH connections. The security group is created first and then referenced when creating the Amazon EC2 instance.

To use this template, save it to a file (e.g., `servertemplate.yml`) and then use the AWS CLI to create a stack:

```
aws cloudformation create-stack --stack-name sa-book-stack --template-body
file://servertemplate.yml
```

You can then use the AWS Management Console, the AWS CLI, or the CloudFormation API to monitor the progress of the stack creation. Once the stack is created, you will have an Amazon EC2 instance running in your AWS account. You can make changes to the stack by updating the template and using the `update-stack` command. You can learn more about AWS CloudFormation by visiting the AWS page here: `https://aws.amazon.com/cloudformation/`.

AWS Service Catalog

AWS Service Catalog provides a centralized platform to manage catalogs of IT services, ensuring adherence to corporate standards and compliance. Organizations can easily control IT service availability, configurations, and access permissions by individual, group, department, or cost center. The platform also simplifies the process of finding and deploying approved IT services for employees. Organizations can define their catalog of AWS services and AWS Marketplace software and make them available for their employees through a self-service portal.

AWS Service Management Connectors allow **IT service management** (**ITSM**) administrators to enhance the governance of provisioned AWS and third-party products.

For instance, by integrating with AWS Service Catalog, ServiceNow and Jira Service Desk can request, provision, and manage AWS and third-party services and resources for their users, streamlining the ITSM process.

AWS Service Catalog AppRegistry is a centralized repository that enables organizations to manage and govern their application resources on AWS. It provides a single place for collecting and managing application metadata, including their name, owner, purpose, and associated resources. This information can be used to improve application visibility and governance and to enable better collaboration between teams that work on different parts of the application stack. With AppRegistry, you can track and manage all your applications' resources, including AWS resources, third-party software, and external resources such as domain names or IP addresses. The following screenshot shows an app registry in the system catalog:

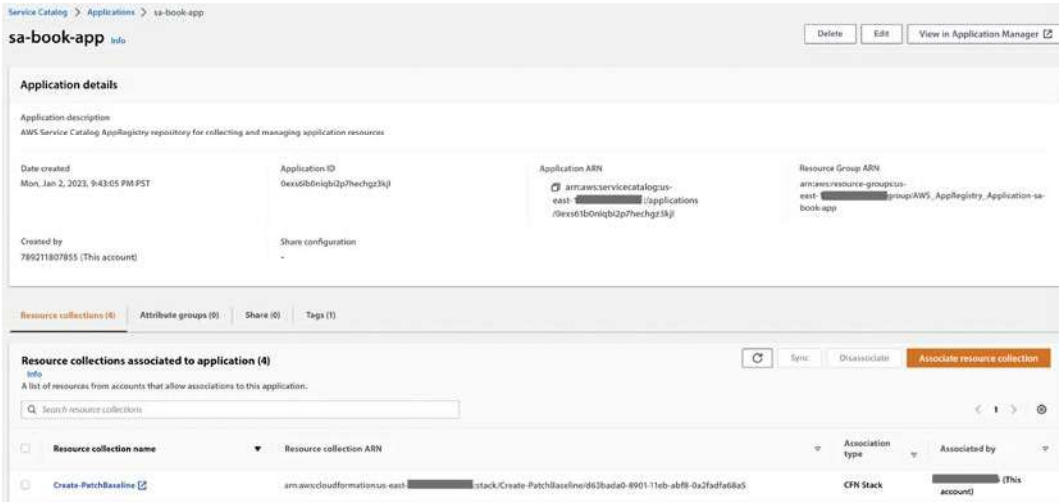

Figure 9.6: AWS System Catalog app registry

With AppRegistry, you can define your application metadata, such as ownership, data sensitivity, and cost centers, and include a reference to your application within the infrastructure code. This helps business stakeholders have up-to-date information about the application's contents and metadata. In addition, AppRegistry provides a central place to track and manage changes to your application resources, helping you maintain a comprehensive view of your applications and their dependencies. You can learn more about AWS Service Catalog by visiting the AWS page here: `https://aws.amazon.com/servicecatalog/`.

AWS Proton

AWS Proton is a managed application deployment service that allows developers to quickly and easily deploy and manage container and serverless applications on AWS. It provides a fully managed, opinionated environment for defining, deploying, and managing applications. With Proton, developers can focus on writing code and building applications while the service handles the underlying infrastructure and deployment workflows. This helps to accelerate the development process, reduce the risk of errors, and improve overall application quality.

AWS provides sample Proton templates that help you start building your application's infrastructure. You can fork those samples using AWS samples code link: (`https://github.com/aws-samples/aws-proton-cloudformation-sample-templates`) and refer to them while building the Proton environment.

AWS Proton can help you update out-of-date applications with a single click when you adopt a new feature or best practice. This helps ensure that your applications remain up-to-date and compliant with industry standards. Additionally, by providing a consistent architecture across your organization, Proton helps improve collaboration and reduces the risk of errors or misconfigurations. You can learn more about AWS Proton by visiting the AWS page here: `https://aws.amazon.com/proton/`.

AWS Cloud Development Kit (CDK)

The AWS CDK is an open-source software development framework that enables developers to define cloud infrastructure and resources using familiar programming languages such as TypeScript, JavaScript, Python, Java, and C#. It provisions and deploys the infrastructure using AWS CloudFormation, providing the benefits of IaC.

With CDK, you will work much faster because you are using your familiar language, concepts, classes, and methods without a context switch. You also have all the tool support from the programming language, such as autocomplete, inline documentation, tests, and a debugger. The most important part is that you can build your abstractions and components of the infrastructure and application. AWS provides many default values, so there is no need to read a lot of documentation; you can start quickly.

The AWS CDK consists of three main components: the core framework, the AWS-construct library, and the CLI. The core framework enables you to define and organize your AWS infrastructure using high-level programming languages. You can create and structure apps that consist of one or multiple stacks. Stacks are the fundamental deployment unit in AWS CDK. They are a logical grouping of AWS resources that are provisioned and managed as a single unit. Each stack is mapped one-to-one to a CloudFormation stack and can be independently deployed, updated, or deleted.

It is good practice to divide resources into stacks with different life cycles: i.e., you would create one stack for network infrastructure such as a VPC, another stack would have an Elastic Container Service cluster, and yet another stack would be the application that is running in this cluster.

The AWS-constructed library in CDK is a collection of pre-built components designed by AWS for creating resources for specific services. This allows for decoupling libraries and using only the necessary dependencies in your project. The library is developed with best practices and security considerations in mind to provide an excellent developer experience, ease of use, and fast iteration cycles. The CDK CLI interacts with the core framework, helping to initialize project structure, inspect deployment differences, and deploy your project quickly to AWS. Here is an example of creating an Amazon S3 bucket using CDK in TypeScript:

```
import * as cdk from 'aws-cdk-lib';
import * as s3 from 'aws-cdk-lib/aws-s3';
class MyBookStack extends cdk.Stack {
  constructor(scope: CDK.App, id: string, props?: CDK.StackProps) {
    super(scope, id, props);
    new s3.Bucket(this, 'BookBucket', {
      bucketName: 'aws-sa-book-bucket',
      publicReadAccess: true
    });
  }
}
const app = new cdk.App();
new MyBookStack(app, 'MyBookStack');
app.synth();
```

This code defines a CDK stack with a single Amazon S3 bucket and synthesizes a CloudFormation template for the stack. You can then use the `cdk deploy` command to deploy the stack to your AWS account.

AWS CDK provides a paradigm shift in how you provision multiple environments. With CloudFormation, you can use one template with parameters for multiple environments, i.e., dev and test. But with CDK, you have a shift where multiple templates are generated for each environment, ending in different stacks. This decoupling helps us to contain and maintain differences between environments by having less expensive resources in the dev environment. You can learn more about AWS CDK by visiting the AWS page here: `https://aws.amazon.com/cdk/`.

AWS Amplify

AWS Amplify is a suite of specialized tools designed to help developers quickly build feature-rich, full-stack web and mobile applications on AWS. It allows developers to utilize a wide range of AWS services as their use cases evolve. With Amplify, developers can configure a backend for their web or mobile app, visually create a web frontend UI, connect the two, and manage app content without needing to access the AWS console. At a high level, AWS Amplify provides the following features, tools, and services:

- **Amplify Libraries**: Frontend developers can use purpose-built Amplify libraries for interacting with AWS services. You can use the case-centric Amplify Libraries for connecting frontend iOS, Android, web, and React Native apps to an AWS backend and UI components for auth, data, and storage. Customers can use Amplify Libraries to build a new app backend or connect an existing backend.
- **Amplify Hosting**: Amplify Hosting is a fully managed CI/CD service for modern web apps. It offers hundreds of global points of presence for fast and reliable hosting of static and server-side rendered apps that scale with your business needs. With Amplify Hosting, you can deploy updates to your web app on every code commit to the Git repository. The app is then deployed and hosted globally using CloudFront. Amplify Hosting supports modern web frameworks such as React, Angular, Vue, Next.js, Gatsby, Hugo, Jekyll, and more.
- **Amplify Studio**: Amplify Studio is a visual development environment that provides an abstraction layer on top of the Amplify CLI. It allows you to create full-stack apps on AWS by building an app backend, creating custom UI components, and connecting a UI to the app backend with minimal coding. With Amplify Studio, you can select from dozens of popular React components, such as buttons, forms, and marketing templates, and customize them to fit your style guide. You can also import UX designs from the popular design prototyping tool, Figma, as clean React code for seamless collaboration. Amplify Studio exports all UI and infrastructure artifacts as code so you can maintain complete control over your app design and behavior.

- **The Amplify CLI**: Provides flexibility and integration with existing CI/CD tools through the new Amplify extensibility features. The Amplify CLI allows frontend developers to set up backend resources in the cloud easily. It's designed to work with the Amplify JavaScript library and the AWS Mobile SDKs for iOS and Android. The Amplify CLI provisions and manages the mobile or web backend with guided workflows for common app use cases such as authentication, data, and storage on AWS. You can reconfigure Amplify-generated backend resources to optimize for specific use cases, leveraging the entire feature set of AWS, or modify Amplify deployment operations to comply with your enterprise DevOps guidelines.

Here's a code example of how to use AWS Amplify in a web application to store and retrieve data from a cloud database:

```
import { API, graphqlOperation } from 'aws-amplify'
// Add a new item to the cloud database
async function addItem(item) {
  const AddItemMutation = `mutation AddItem($item: ItemInput!) {
    addItem(item: $item) {
      id
      name
      description
    }
  }`
  const result = await API.graphql(graphqlOperation(AddItemMutation, {
item }))
  console.log(result)
}
// Retrieve a list of items from the cloud database
async function listItems() {
  const ListItemsQuery = `query ListItems {
    listItems {
      items {
        id
        name
        description
      }
    }
  }`
```

```
    const result = await API.graphql(graphqlOperation(ListItemsQuery))
    console.log(result)
  }
```

The `API` object provided by Amplify enables you to call GraphQL operations to interact with the cloud database. The `addItem` function uses the `addItem` mutation to create a new item in the database, while the `listItems` function uses the `listItems` query to retrieve a list of items.

AWS Amplify ties into a broader array of tools and services provided by AWS. You can customize your Amplify toolkit and leverage AWS services' breadth and depth to service your modern application development needs. You can choose the services that suit your application and business requirements and scale confidently on AWS. You can learn more about AWS Amplify by visiting the AWS page here: `https://aws.amazon.com/amplify/`.

In this section, you were introduced to AWS CDK, which is a provisioning and orchestration solution that facilitates the consistent and repeatable provisioning of resources. By utilizing AWS CDK, you can scale your organization's infrastructure and applications on AWS in a sustainable manner. Additionally, AWS CDK allows you to create your infrastructure as code using programming languages. You can simplify and accelerate the governance and distribution of IaC templates using the AWS Service Catalog to create repeatable infrastructure and application patterns with best practices. Now let's learn about the next step to set up monitoring and observations from your applications.

Fourth pillar – Monitor & observe your applications

As the saying goes, you manage what you measure, so after you've provisioned your application, you have to be able to start measuring its health and performance. Monitoring and observability tools help to collect metrics, logs, traces, and event data. You can quickly identify and resolve application issues for serverless, containerized, or other applications built using microservices-based architectures.

AWS provides native monitoring, logging, alarming, and dashboards with CloudWatch and tracing through X-Ray. When deployed together, they provide the three pillars of an observability solution: metrics, logs, and traces. X-Ray (tracing) is fundamental to observability and is therefore included in the motions alongside CloudWatch. Furthermore, AWS provides open-source observability for Prometheus and Grafana and support for Open Telemetry. A well-defined monitoring and observability strategy implemented with CloudWatch and X-Ray provides insights and data to monitor and respond to your application's performance issues by providing a consolidated view of the operation.

The following are the key AWS services for observability and monitoring, which you can choose to use as per your workload needs or combine them together.

AWS CloudWatch helps you collect, view, and analyze metrics and set alarms to get notified when certain thresholds are breached. Here you can see a screenshot of the billing metrics dashboard in CloudWatch:

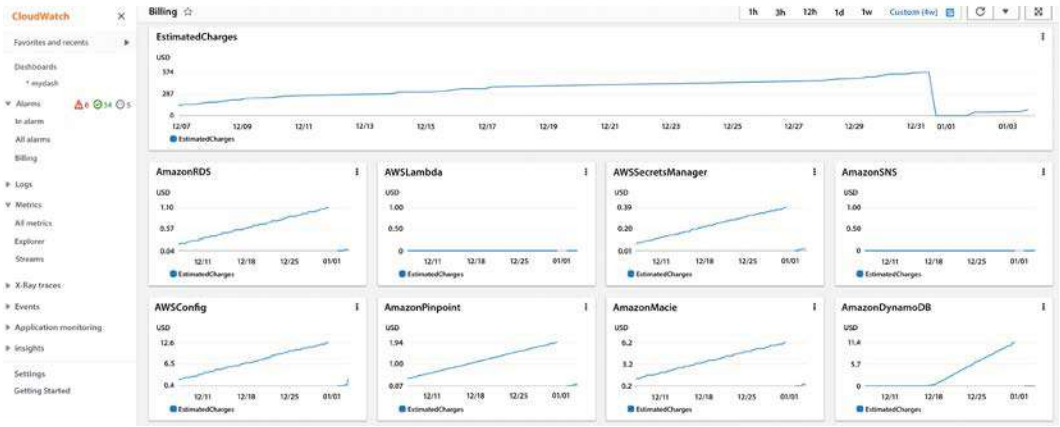

Figure 9.7: AWS CloudWatch billing metrics dashboard

You can also use CloudWatch to track log files from Amazon EC2 instances, Amazon RDS DB instances, and other resources and troubleshoot issues with your applications. You learned about AWS CloudWatch earlier in this chapter.

AWS X-Ray is a distributed tracing service that allows developers to analyze and debug their applications, especially those built using a microservices architecture. It helps identify performance bottlenecks and errors, allowing developers to optimize application performance and enhance the end-user experience.

It allows you to trace requests as they flow through your application and see the performance of each component of your application:

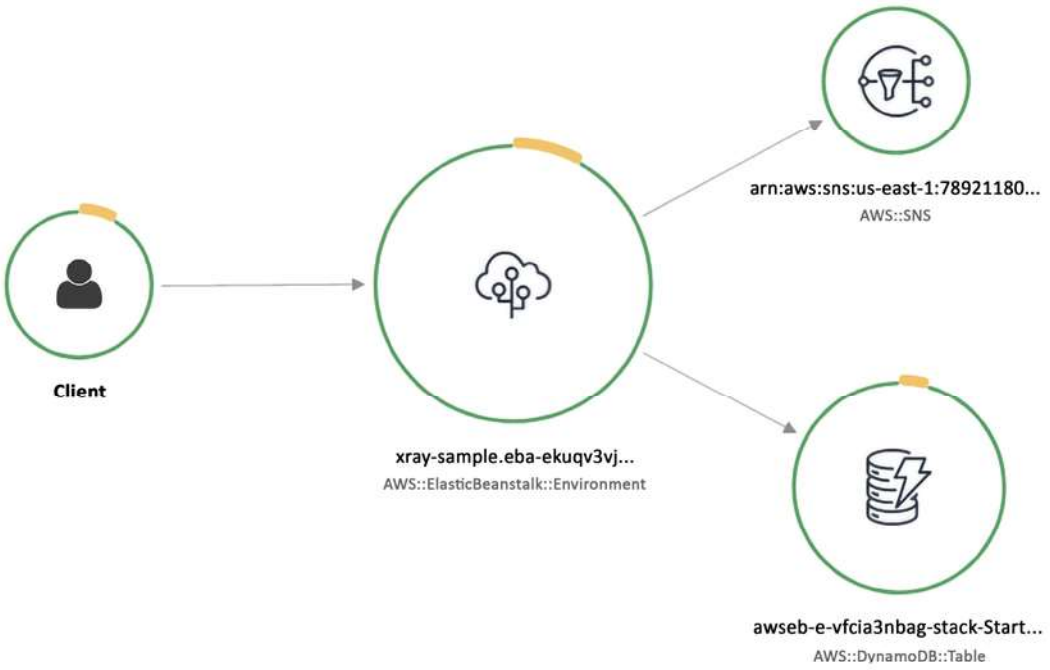

Figure 9.8: AWS X-Ray service map graph

In the AWS X-Ray console, you can view a trace of a request as it flows through your application and see the performance of each trace segment. You can also use the console to search for traces and analyze performance data for your application. To use AWS X-Ray, you first need to instrument your application code to send data to the X-Ray service. This can be done using one of the AWS SDKs or the X-Ray daemon. You can then view and analyze the data using the X-Ray console or the X-Ray API. You can learn more about AWS X-Ray by visiting the AWS page here: `https://aws.amazon.com/xray/`.

Amazon Managed Service for Prometheus (**AMSP**) is a fully managed service that makes it easy to run and scale Prometheus, an open-source monitoring and alerting system, in the cloud. AMSP automatically handles tasks such as scaling and maintenance, allowing you to focus on monitoring your applications. It also includes integration with other AWS services, such as Amazon CloudWatch and Amazon SNS, which allows you to view and analyze your monitoring data and set up alerts and notifications. The followings are the benefits of using AMSP:

- **Fully managed**: AMSP takes care of the underlying infrastructure and maintenance tasks, so you can focus on monitoring your applications.
- **Scalability**: AMSP automatically scales to handle changes in workload, so you don't have to worry about capacity planning.
- **Integration with other AWS services**: AMSP integrates with other AWS services, such as CloudWatch and SNS, which allows you to view and analyze your monitoring data and set up alerts and notifications.
- **Security**: AMSP includes built-in security measures, such as encryption at rest and network isolation, to help protect your monitoring data.
- **Cost-effective**: AMSP is a pay-as-you-go service, which means you only pay for the resources you use. This can be more cost-effective than running and maintaining your own Prometheus infrastructure.

You can learn more about AMSP by visiting the AWS page here: `https://aws.amazon.com/prometheus/`.

Amazon Managed Service for Grafana (**AMG**) is a fully managed service that simplifies the visualization and analysis of operational data at scale. It leverages the popular open-source analytics platform Grafana to query, visualize, and alert on metrics stored across AWS, third-party ISVs, databases, and other IT resources. AMG removes the need for server provisioning, software configuration, and security and scaling concerns, enabling you to analyze your metrics, logs, and traces without the heavy lifting in production. You can learn more about AMG by visiting the AWS page here: `https://aws.amazon.com/grafana/`.

In this section, you learned that an observable environment reduces risk, increases agility, and improves customer experience. Observability provides insights and context about the environment you monitor. AWS enables you to transform from monitoring to observability so that you can have full-service visibility from metrics, logs, and traces by combining AWS CloudWatch and X-Ray. Let's learn about the next step in building centralized operations.

Fifth pillar — Centralized operations management

IT teams need to take operational actions across hundreds, sometimes thousands, of applications while maintaining safety, security, and compliance simultaneously. To help make ops management as easy and efficient as possible, you must safely manage and operate your IT infrastructure at scale. To achieve that, you should have a central location and interface to view operational data from multiple AWS services. You can then automate operational tasks on applications and resources, especially common operational changes, such as rotating certificates, increasing service limits, taking backups, and resizing instances. You laid down all without compromising any safety, security, or compliance guardrails in the foundation stage.

To help enable cloud operations, you can use **AWS Systems Manager**. Systems Manager is a fully managed service that helps customers safely manage and operate their IT infrastructure at scale. It provides a central location and interfaces to view operational data from multiple AWS services. Customers can then use it to automate operational tasks on their applications and resources, especially common operational changes, such as rotating certificates, increasing service limits, taking backups, and resizing instances.

You learned about AWS Systems Manager earlier in this chapter. Here you will learn about Systems Manager's ability to view, manage, operate, and report on cloud operations. There are **four** stages of implementing CloudOps using AWS Systems Manager:

1. **Build a foundation for cloud operations**: To build a foundation for cloud operations, the Systems Manager helps you set up the management of service configurations, IT assets, infrastructure, and platforms. Systems Manager integrates with AWS Config to collect and maintain an inventory of infrastructure resources and application and OS data starting with your environment and account structure. Simple, automated setup processes allow you to quickly enable operational best practices, such as continuous vulnerability scanning and collecting insights into improving an application's performance and availability. It also helps you to automate operational best practices by codifying operations runbooks and defining execution parameters, such as freeze periods and permissions. AWS Config rules enable continuous compliance by enforcing security, risk, and compliance policies.
2. **Enable visibility into applications and infrastructure**: The second stage is to enable visibility into applications and infrastructure by continuously tracking key technical and business measures, providing visibility, and triggering automation and actions if needed.

Systems Manager provides operational visibility and actionable insights to customers through a widget-based, customizable dashboard that can be tailored for users such as IT operators, DevOps engineers, IT leaders, and executives. Operational teams can set up operational event management and reporting for their infrastructure and resources at scale using pre-defined configuration bundles that reflect operational best practices to filter out and capture specific operational events that require an operator's attention for diagnosis and action/remediation. The dashboard provides a holistic view of relevant data across multiple AWS accounts and AWS Regions, such as inventory and CMDB, patch, resource configuration and compliance, support tickets, insights from EC2 Compute Optimizer, Trusted Advisor, Personal Health Dashboard, Amazon CloudWatch, and trends on outstanding operational issues.

3. **Automate operations at scale**: To proactively automate operations at scale, Systems Manager gives teams the ability to automate patch management to keep resources secure. Systems Manager provides change management capabilities with built-in approval workflows and secures automated or manual change execution when making application and environment changes. Only approved changes can be deployed to the environment by authorized resources, and these come with detailed reporting. You can automate server administration, providing central IT teams with a consistent, integrated console to perform common administrative tasks. Additionally, you can manage and troubleshoot resources on AWS and on-premises. Operators can manage their VM fleet when manual actions are required by connecting directly from the console. You can also operationalize risk management by creating rules. Here is an example AWS Systems Manager rule that operationalizes risk management by checking for security vulnerabilities in installed software packages on Amazon EC2 instances:

```
{
  "Name": "ec2-check-for-security-vulnerabilities",
  "Description": "Scans installed software packages on EC2 instances
for security vulnerabilities",
  "ResourceId": "*",
  "ResourceType": "AWS::EC2::Instance",
  "ComplianceType": "NON_COMPLIANT",
  "RulePriority": 1,
  "Operator": "EQUALS",
  "Parameters": {
    "ExecutionFrequency": "OneTime",
```

```
    "OutputS3BucketName": "sa-book-s3-bucket",
    "OutputS3KeyPrefix": "ec2-security-scans/"
  },
  "Actions": [
    {
      "Type": "RunCommand",
      "Properties": {
        "Comment": "Scan installed software packages for security
vulnerabilities",
        "OutputS3BucketName": "sa-book-s3-bucket",
        "OutputS3KeyPrefix": "ec2-security-scans/",
        "DocumentName": "AWS-RunShellScript",
        "Parameters": {
          "commands": [
            "apt update",
            "apt-get install -y unattended-upgrades",
            "apt-get install -y --only-upgrade bash",
            "apt-get install -y --only-upgrade glibc",
            "apt-get install -y --only-upgrade libstdc++6",
            "apt-get install -y --only-upgrade libgcc1",
            "apt-get install -y --only-upgrade libc6",
            "apt-get install -y --only-upgrade libc-bin",
            "apt-get install -y --only-upgrade libpam-modules",
            "apt-get install -y --only-upgrade libpam-runtime",
            "apt-get install -y --only-upgrade libpam0g",
            "apt-get install -y --only-upgrade login",
            "apt-get install -y --only-upgrade passwd",
            "apt-get install -y --only-upgrade libssl1.0.0",
            "apt-get install -y --only-upgrade openssl",
            "apt-get install -y --only-upgrade dpkg",
            "apt-get install -y --only-upgrade apt",
            "apt-get install -y --only-upgrade libapt-pkg4.12",
            "apt-get install -y --only-upgrade apt-utils",
            "apt-get install -y --only-upgrade libdb5.3",
            "apt-get install -y --only-upgrade bzip2",
            "apt-get install -y --only-upgrade libbz2-1.0",
            "apt-get install -y --only-upgrade liblzma5",
```

```
            "apt-get install -y --only-upgrade libtinfo5",
            "apt-get install -y --only-upgrade libreadline7",
            "apt
]
}
```

By running analyses to proactively detect and remediate risks across applications and infrastructure, such as expiring certificates, a lack of database backup, and the use of blocked software, identified risks are assigned to owners and can be remediated using automation runbooks with automated reporting.

4. **Remediate issues and incidents**: Finally, when unexpected issues arise, you must be able to remediate issues and incidents quickly. With the incident, event, and risk management capabilities within Systems Manager, you can employ various AWS services to trigger relevant issues or incidents. It integrates with Amazon GuardDuty for threat detection and AWS Inspector for security assessments, keeps a running check on vulnerabilities in the environment, and allows automated remediation. It provides a consolidated view of incidents, changes, operational risks and failures, operational alarms, compliance, and vulnerability management reports. It allows operations teams to take manual or automated action to resolve issues. Systems Manager speeds up issue resolution by automating common, repeatable remediation actions, such as failover to a backup system and capturing failed state for root cause analysis.

Systems Manager's incident management capability automates a response plan for application issues by notifying the appropriate people to respond, providing them with relevant troubleshooting data, and enabling chat-based collaboration. You can easily access and analyze operational data from multiple AWS services and track updates related to incidents, such as changes in alarm status and response plans. Operators can resolve incidents manually by logging into the instance or executing automation runbooks. Systems Manager integrates with AWS Chatbot to invoke commands in the appropriate channel for an incident so central IT teams can resolve issues quickly. Let's learn about the final and most important step in the cloud operation model: managing cost.

Sixth pillar – Manage your cloud's finance

Cloud adoption has enabled technology teams to innovate faster by reducing approval, procurement, and infrastructure deployment cycles. It also helps finance organizations eliminate the failure cost, as cloud resources can be terminated with just a few clicks or API calls. As a result, technology teams are no longer just builders, but they also operate and own their products.

They are responsible for most activities that were traditionally associated with finance and operations teams, such as procurement and deployment.

Cloud Financial Management (CFM) enables finance, product, technology, and business organizations to manage, optimize and plan costs as they grow their usage and scale on AWS. The primary goal of CFM is to enable customers to achieve their business outcomes cost-efficiently and accelerate economic and business value creation while balancing agility and control. CFM has the following four dimensions to manage and save costs:

Plan and evaluate

When planning for future cloud spending, you should first define a goal for the monthly cost of the individual project. The project team needs to make sure cloud resources related to a project are correctly tagged with cost allocation tags and/or cost categories. This way, they can calculate and track the monthly cost of the project with the cost and usage data available in their AWS Cost Explorer and AWS Cost and Usage Reports. These reports provide the data you need to understand how your AWS costs are incurred and optimize your AWS usage and cost management. These reports can be customized to include only the needed data and can be delivered to an Amazon S3 bucket or an Amazon SNS topic. You can also use the data in these reports to create custom cost and usage reports, set up cost and usage alarms, and create budgets.

You can decide on a project's budget based on the growth trend of the project as well as the available funds set aside for the project. Then, you can set the budget thresholds using AWS Budgets for cost or resource usage. You can also use AWS Budgets to set coverage and utilization targets for the project team's Reserved Instances and Savings Plans. These are two options that allow you to save money on your AWS usage costs. They both enable you to purchase a discounted rate for your AWS usage, in exchange for committing to a certain usage level over a specific period. **Reserved Instances** are a type of pricing model that allows you to save up to 75% on your Amazon EC2 and RDS usage costs by committing to a one- or three-year term. With Reserved Instances, you pay a discounted hourly rate for the usage of a specific instance type in a specific region, and you can choose between Standard and Convertible Reserved Instances.

AWS Savings Plans is a new pricing model that allows you to save up to 72% on your AWS usage costs by committing to a one-year or three-year term. With Savings Plans, you pay a discounted hourly rate for your AWS usage, and you can choose between Compute Savings Plans and Amazon EC2 Instance Savings Plans. Compute Savings Plans offer a discount on a wide range of AWS services, including Amazon EC2, Fargate, and Lambda, while Amazon EC2 Instance Savings Plans only offer a discount on Amazon EC2 usage.

The AWS Budgets Reports dashboard allows you to monitor the progress of your budget portfolio by comparing actual costs with the budgeted costs and forecasted costs with the budgeted costs. You can set up notification alerts to receive updates when the cost and usage are expected to exceed the threshold limit. These alerts can be sent via email or Amazon **Simple Notification Service (SNS)**. You can learn more about AWS Budgets by visiting the AWS page here: `https://aws.amazon.com/aws-cost-management/aws-budgets/`. After planning your budget, let's learn about managing it.

Manage and Control

As businesses grow and scale on AWS, you need to give your team the freedom to experiment and innovate in the cloud while maintaining control over cost, governance, and security. And while fostering innovation and speed is essential, you also want to avoid getting surprised by the bill.

You can achieve this by establishing centralized ownership through a center of excellence. Cost management elements are shared responsibilities across the entire organization. A centralized team is essential to design policies and governance mechanisms, implement and monitor the effort, and help drive company-wide best practices.

You can utilize services like **Identity and Access Management (IAM)** to ensure secure access control to AWS resources. IAM allows you to create and manage user identities, groups, and roles and grants permissions for IAM users to access specific AWS resources. This way, you can control and restrict individual and group access to AWS resources, ensuring the security of your infrastructure and data.

Use **AWS Organizations** to enable automatic policy-based account creation, management, and billing at scale. Finally, using the **AWS Billing Console**, you can easily track overall spending and view cost breakdown by service and account by accessing Billing Dashboard. You can view the overall monthly spending from last month, the current month, and the current forecasted month.

The following screenshot is a sample billing dashboard:

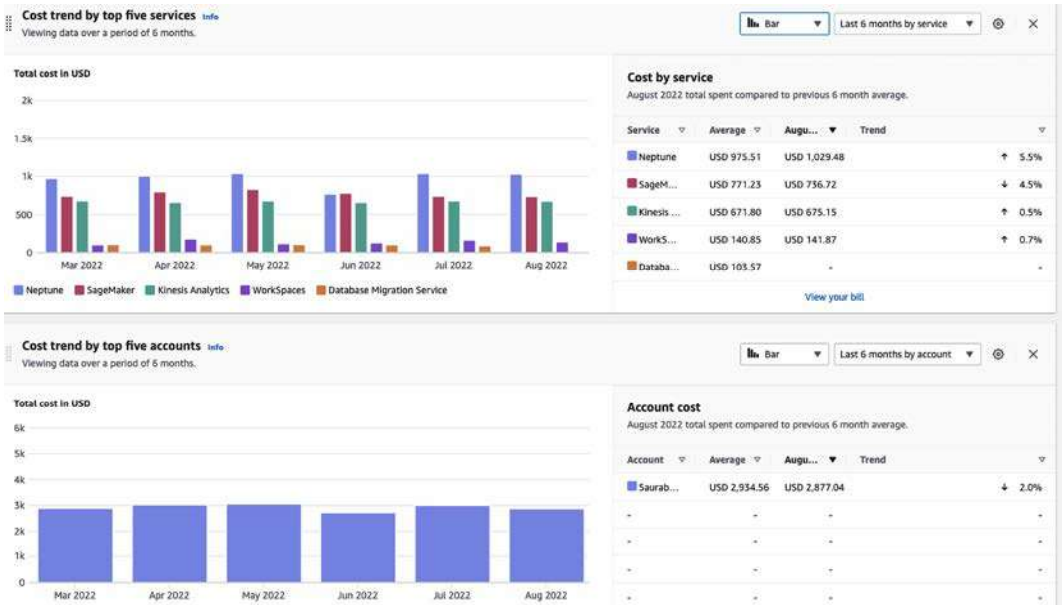

Figure 9.9: AWS Billing Dashboard

The above billing dashboard shows six months spend by service and cost trend. Using the Billing Console, you can receive a unified view of spend in a single bill and establish rules for organizing costs, sharing discount benefits associated with Reserved Instances and Savings Plans, and many other controls. You can learn more about the AWS Billing Console by visiting the AWS page here: `https://aws.amazon.com/aws-cost-management/aws-billing/`.

AWS Purchase Order Management enables you to use **purchase orders (POs)** to procure AWS services and approve invoices for payment. With this service, you can configure multiple POs, map them to your invoices, and access the invoices generated against those POs. Additionally, you can manage the status of your POs, track their balance and expiration, and set up email notifications for contacts to receive alerts when POs are running low on balance or close to their expiration date. You can learn more about AWS Purchase Order Management by visiting the AWS page here: `https://aws.amazon.com/aws-cost-management/aws-purchase-order-management/`.

AWS Cost Anomaly Detection is a machine learning service that automates cost anomaly alerts and root cause analysis. It can save time investigating spending anomalies by providing automated root cause analysis and identifying potential cost drivers, such as specific AWS services, usage types (e.g., data transfer cost), regions, and member accounts. You can learn more about AWS Cost Anomaly Detection by visiting the AWS page here: `https://aws.amazon.com/aws-cost-management/aws-cost-anomaly-detection/`. As you manage your cost, let's learn how to track it.

Track and allocate

You need to ask three questions to understand your billing uses. The **first** is, *What is causing our bill to increase?* AWS provides **AWS Cost Explorer** to help answer this question. Cost Explorer provides a quick visualization of cost and utilization with default reports and creates specific views with filters and grouping, as shown below. This tool can help show which AWS services are leading to increased spending:

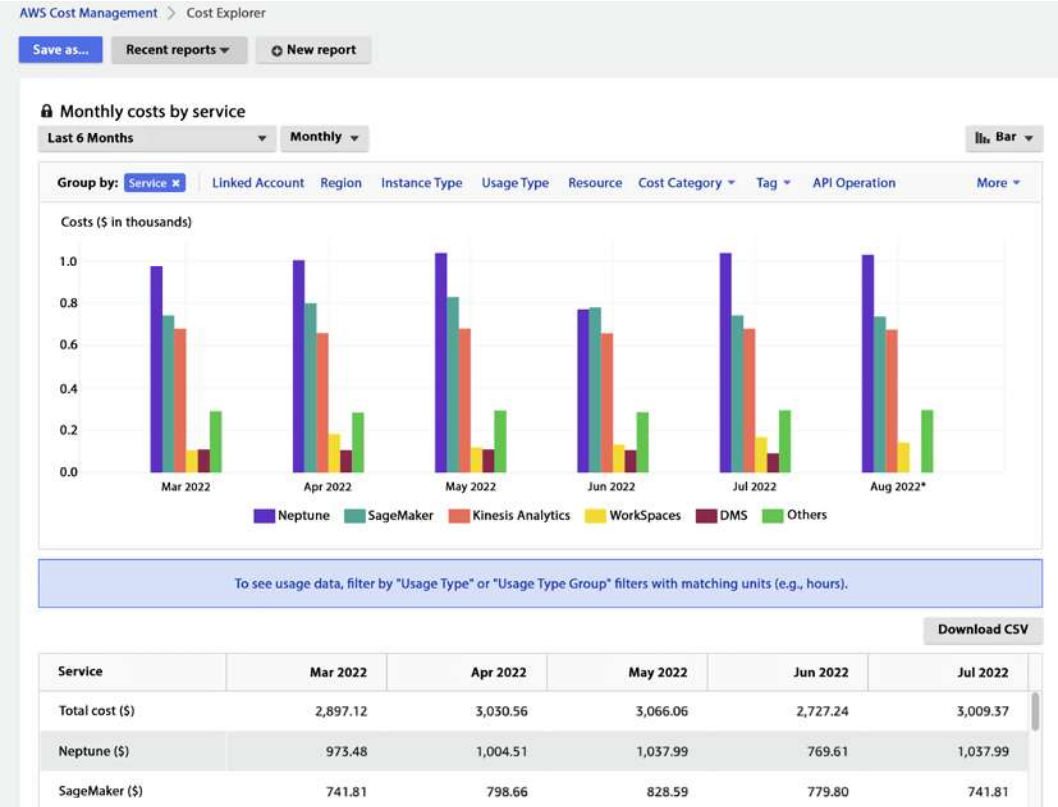

Figure 9.10: AWS Cost Explorer

In the above Cost Explorer dashboard, you can see service expense grouping in the costs chart showing AWS Neptune has the highest cost, followed by SageMaker. It also provides the ability to download CSV files for detailed analysis. You can learn more about AWS Cost Explorer by visiting the AWS page here: `https://aws.amazon.com/aws-cost-management/aws-cost-explorer/`.

The **second** question is a common follow-up: *Which of my lines of business, teams, or organizations drove increased spending and usage?* AWS Cost Explorer's data becomes even more valuable when paired with Cost Allocation Tags. These features allow customers to categorize their resources and spending to fit their organization's needs. Categorizing spending by specific teams, sometimes referred to as show back, allows for better analysis and easier identification of savings opportunities. You can also use the AWS **Cost and Usage Report (CUR)** to bring in cost and usage data, including tags, into your analysis tool of choice. This approach also allows for combining the data with other business-specific data. You can learn more about CUR by visiting the AWS page here: `https://aws.amazon.com/aws-cost-management/aws-cost-and-usage-reporting/`.

Finally, the **third** question is, *How do I understand the cost impact of Reserved Instances?* AWS Cost Explorer provides multiple ways to view and dive deep into this data, such as tag service with meaningful info, e.g., owner, project, application, or you can define cost categories by group accounts, tags, services, and charge types with custom rules. You can use CUR to deliver cost data to the S3 bucket, which can be integrated with Amazon Athena and/or ingested into your ERP system.

Furthermore, **AWS Billing Conductor** is a billing and cost management tool that helps you monitor and optimize your AWS costs and usage and provides insights into how you are using your resources. AWS Billing Conductor provides recommendations for ways to optimize your costs, such as by identifying idle or underutilized resources that can be turned off or scaled down. You can learn more about AWS Billing Conductor by visiting the AWS page here: `https://aws.amazon.com/aws-cost-management/aws-billing-conductor/`.

AWS Application Cost Profiler is a service that allows you to collect and correlate usage data from your multi-tenant applications with your AWS billing information. This enables you to generate detailed, tenant-specific cost reports with hourly granularity delivered daily or monthly. This can be useful for understanding the cost breakdown of your multi-tenant application and allocating costs to individual tenants or customers. You can learn more about AWS Application Cost Profiler by visiting the AWS page here: `https://aws.amazon.com/aws-cost-management/aws-application-cost-profiler/`. Now let's learn about the final step to optimize costs and increase savings.

Optimize and save

Cost optimization is about making sure you pay only for what you need. AWS offers tools and services that make it easier for you to build cost-efficient workloads, which help you to continue saving as you scale on AWS. While there are hundreds of ways you can pull to minimize spending, at its core, two of the following most impactful levers you can pull to optimize spending are detailed next.

Using the right pricing models: AWS offers services through multiple pricing models — on-demand, pay-as-you-go, commitment-based Reserved Instances, and Spot instances for up to 90% discount compared to on-demand pricing. Amazon EC2 Spot Instances are a type of AWS computing resource that is available at a reduced price and can be used to run your applications. These instances are spare capacities in the AWS cloud that can be interrupted at any time, based on the resource demand. Spot Instances are suitable for applications with flexible start and end times and can handle interruptions without causing significant issues. You can launch a Spot Instance by specifying the maximum price you are willing to pay per hour (known as the "bid price"). If the current Spot price is less than your bid price, your Spot Instance will be launched, and you will be charged the current Spot price. However, if the current Spot price exceeds your bid price, your Spot Instance will be interrupted, and you will not be charged for the usage.

Suppose you have predictable, steady workloads on Amazon EC2, Amazon ECS, and Amazon RDS. If you use Reserved Instances, you can save up to 75% over on-demand capacity. Reserved Instances are a pricing option in AWS that allows you to pay up front for a commitment to use a certain amount of resources over a specific period in exchange for a discounted price. There are three options for purchasing Reserved Instances:

- **All up-front (AURI)**: This option requires you to pay for the entire term of the Reserved Instance up front and provides the most significant discount.
- **Partial up-front (PURI)**: This option requires a partial payment up front and provides a smaller discount than the AURI option.
- **No upfront payments (NURI)**: This option does not require any up front payment and provides the smallest discount.

By choosing the AURI option, you can receive the largest discount on your Reserved Instances. The PURI and NURI options offer lower discounts but allow you to pay less up front or avoid upfront payments altogether.

AWS offers Reserved Instances, and Savings Plans purchase recommendations via Cost Explorer based on your past usage. Cost Explorer identifies and recommends the estimated value resulting in the largest savings. It allows you to generate a recommendation specific to your purchase preference. From there, you can track your investment using cost explorer. Any usage above the commitment level will be charged at on-demand rates. You can revisit commitment levels, make incremental purchases, and track coverage and utilization using pre-built reports in cost explorer.

With AWS purchase option recommendations, you can receive tailored Reserved Instance, or Savings Plans purchase recommendations based on your historical usage. You can select parameters for recommendations, such as the type of plan, term commitment, and payment option that makes sense for your business.

Identify and eliminate idle or over-provisioned resources: AWS Cost Explorer resources can be used for top-level **key performance indicators** (**KPIs**), rightsizing, and instance selection. Cost Explorer will estimate monthly savings, which is the sum of the projected monthly savings associated with each recommendation.

Cost Explorer rightsizing recommendations generate recommendations by identifying idle and underutilized instances and searching for smaller instance sizes in the same instance family. Idle instances are defined as CPU utilization of <1%, while underutilized instances are defined as those with CPU utilization between 1% and 40%.

In this section, you learned about various ways to manage your cost and make the cloud more profitable to run your workload and business.

Summary

Working in a cloud environment is different from on-premise. The out-of-the-box tools and services are available in AWS can make a huge difference when operating your workload in the cloud. To realize the full value of the cloud, it's important to understand cloud operation and how to apply automation everywhere. In this chapter, you learned about the cloud operation model and the six pillars of CloudOps, which help you to understand how to plan your cloud operation efficiently.