# AWS® Certified Cloud Practitioner

# STUDY GUIDE

## FOUNDATIONAL (CLF-C01) EXAM

**Includes interactive online learning environment and study tools:**

**Two custom practice exams**

**100 electronic flashcards**

**Searchable key term glossary**

BEN PIPER
DAVID CLINTON

SYBEX
A Wiley Brand

Until you actively launch some services, there won't be anything happening that will actually generate billable events.

## What's Available Under the Free Tier?

There are actually two kinds of free services:

- Light implementations of most AWS services that are available through the first 12 months of a new AWS account
- Light usage of certain AWS services that are always available—even after your initial 12 months are up

You can get a detailed rundown of everything that's available for both classes at `https://aws.amazon.com/free`. To give you some sense of what's out there, the following are some highlights.

### The 12-Month Free Tier

We've already discussed running low-powered EC2 and RDS instances and using 5 GB of S3 storage capacity under the Free Tier. Other available free service features include the following:

- 30 GB of either magnetic or solid-state drive (SSD) volumes from Amazon Elastic Block Storage (EBS). EBS volumes are usually used as the primary data drives for EC2 instances. For context, 8 GB is normally more than enough space to run the Linux OS driving a busy web server.
- 500 MB/month of free storage on the Amazon Elastic Container Registry (ECR). ECR is a service for storing and managing Docker container images. 500 MB is enough space to satisfy the needs of nearly any use case.
- 50 GB of outbound data transfers and 2 million HTTP or HTTPS requests for Amazon CloudFront distributions. (These are annual rather than monthly amounts.) CloudFront is Amazon's content delivery network that you can use to serve cached copies of your content to remote users at low latency rates.
- One million API calls/month on Amazon API Gateway. API Gateway is a tool for managing your organization's application programming interfaces (APIs). APIs allow users and applications to connect with mobile and web-based resources.

### Permanently Free Services

Some AWS services can be useful even when consumed at levels so low that they don't need to be billed. In such cases, AWS makes them available at those low levels over any time span. These use cases include the following:

- 10 custom monitoring metrics and 10 alarms on Amazon CloudWatch active at any given time. You use CloudWatch either directly or in combination with other tools to track resource performance and costs.

enough to run parallel resources if they're going to be sitting right next to each other in the same data center. That wouldn't do you a lot of good in the event of a building-wide blackout or a malicious attack, leaving your backup just as dead as the instance it was supposed to replace. So, you'll also need to distribute your resources across remote locations.

In this context, it really makes no difference whether your workload is running in your on-premises data center or in the Amazon cloud. In either case, you'll need resource redundancy that's also geographically parallel. What *is* different is how much *easier*—and sometimes cheaper—it can be to build resilience into your *cloud* infrastructure.

Since the AWS cloud is already available within dozens and dozens of Availability Zones spread across all continents (besides, for now at least, Antarctica), deploying or, at least, preparing quick-launch templates for remote backup instances is easy. And since AWS workloads can be requested and launched on-demand, the job of efficiently provisioning parallel resources is built into the platform's very DNA.

The configuration and automation stage can be a bit tricky, but that's for your developers and administrators to figure out, isn't it? They're the ones who took and passed the AWS Solutions Architect Associate (or Professional) certification, right?

You should at least be aware that AWS slays the application failure dragon using autoscaling and load balancing:

- Autoscaling can be configured to replace or replicate a resource to ensure that a predefined service level is maintained regardless of changes in user demand or the availability of existing resources.
- Load balancing orchestrates the use of multiple parallel resources to direct user requests to the server resource that's best able to provide a successful experience. A common use case for load balancing is to coordinate the use of primary and (remote) backup resources to cover for a failure.

# AWS Global Infrastructure: Edge Locations

The final major piece of the AWS infrastructure puzzle is its network of edge locations. An *edge location* is a site where AWS deploys physical server infrastructure to provide low-latency user access to Amazon-based data.

That definition is correct, but it does sound suspiciously like the way you'd define any other AWS data center, doesn't it? The important difference is that your garden-variety data centers are designed to offer the full range of AWS services, including the complete set of EC2 instance types and the networking infrastructure customers would need to shape their compute environments. Edge locations, on the other hand, are much more focused on a smaller set of roles and will therefore stock a much narrower set of hardware.

So, what actually happens at those edge locations? You can think of them as a front-line resource for directing the kind of network traffic that can most benefit from speed.

## Edge Locations and CloudFront

Perhaps the best-known tenant of edge locations is CloudFront, Amazon's CDN service. How does that work? Let's say you're hosting large media files in S3 buckets. If users would have to retrieve their files directly from the bucket each time they were requested, delivery—especially to end users living continents away from the bucket location—would be relatively slow. But if you could store cached copies of the most popular files on servers located geographically close to your users, then they wouldn't have to wait for the original file to be retrieved but could be enjoying the cached copy in a fraction of the time.


Not all content types are good candidates for CloudFront caching. Files that are frequently updated or that are accessed by end users only once in a while would probably not justify the expense of saving to cache.

In addition to CloudFront, there are other AWS services that make use of edge locations. Here are few examples:

**Amazon Route 53** Amazon's Domain Name System (DNS) administration tool for managing domain name registration and traffic routing

**AWS Shield** A managed service for countering the threat of distributed denial-of-service (DDoS) attacks against your AWS-based infrastructure

**AWS Web Application Firewall (WAF)** A managed service for protecting web applications from web-based threats

**Lambda@Edge** A tool designed to use the serverless power of Lambda to customize CloudFront behavior

So all this talk will make more sense, we suggest you explore the configuration process for a CloudFront distribution by following Exercise 4.2.

**EXERCISE 4.2**

### Take a Quick Look at the Way CloudFront Distributions Are Configured

1. Choose the CloudFront link from the AWS services menu, choose Create Distribution, and then choose the Get Started button underneath the Web Distribution section of the "Select a delivery method for your content" page.
2. Once you're on the Create Distribution page, choose inside the box next to the Origin Domain Name item. If you have any content (like a publicly available S3 bucket), it should appear as an option. Either way, since you're not planning to actually launch anything right now, it doesn't matter what—if any—value you enter.
3. Scroll down the page and note, for instance, the Distribution Settings section. Choose the Price Class drop-down arrow to see the available options for the scope of your distribution (the number of edge locations where your content will be saved).

4. Note the Distribution State settings at the bottom of the page.
5. When you're done exploring, choose the Cancel link at the bottom to ensure you don't accidentally launch a distribution for no purpose.

## Regional Edge Cache Locations

In addition to the fleet of regular edge locations, Amazon has further enhanced CloudFront functionality by adding what it calls a *regional edge cache*. The idea is that CloudFront-served objects are maintained in edge location caches only as long as there's a steady flow of requests. Once the rate of new requests drops off, an object will be deleted from the cache, and future requests will need to travel all the way back to the origin server (like an S3 bucket).

Regional edge cache locations—of which there are currently nine worldwide—can offer a compromise solution. Objects rejected by edge locations can be moved to the regional edge caches. There aren't as many such locations worldwide, so the response times for many user requests won't be as fast, but that'll still probably be better than having to go all the way back to the origin. By design, regional edge cache locations are more capable of handling less-popular content.

# The AWS Shared Responsibility Model

The AWS cloud—like any large and complex environment—is built on top of a stack of rules and assumptions. The success of your AWS projects will largely depend on how well you understand those rules and assumptions and on how fully you adopt the practices that they represent. The AWS Shared Responsibility Model is a helpful articulation of those rules and assumptions, and it's worth spending some time thinking about it.

Amazon distinguishes between the security and reliability *of* the cloud, which is its responsibility, and the security and reliability of what's *in* the cloud, which is up to you, the customer.

The cloud itself consists of the physical buildings, servers, and networking hardware used by AWS data centers. AWS is responsible for making sure that its locations are secure, reliably powered, and properly maintained. AWS is also on the hook for patching, encrypting (where relevant), and maintaining the operating systems and virtualization software running its physical servers and for the software running its managed services.

But that's where things can get a bit complicated. What exactly is "managed" and what's "unmanaged"? Figure 4.3 (which you saw previously in Chapter 1, "The Cloud") compares the "of the cloud/in the cloud" mix as it applies across the three key cloud models: Infrastructure as a Service, Platform as a Service, and Software as a Service.

**FIGURE 4.3** A general comparison between local and managed deployments

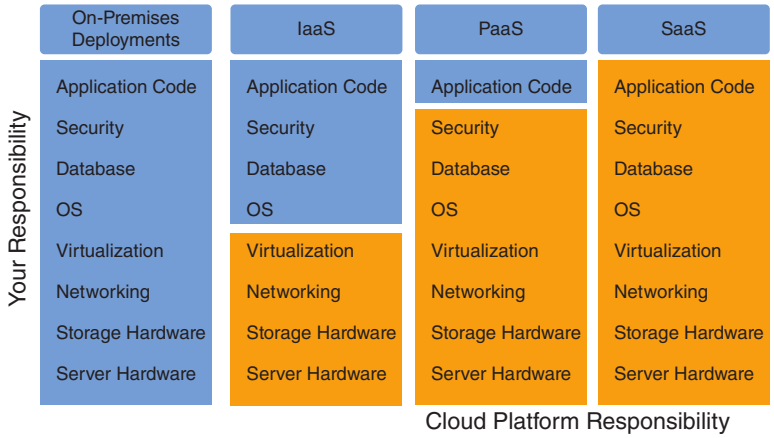
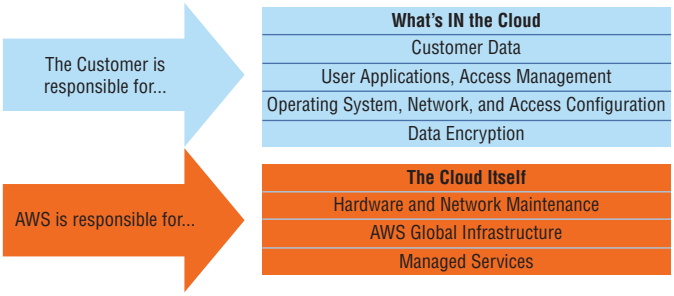



Figure 4.4 illustrates the way responsibility for the integrity and security of AWS infrastructure is divided between Amazon and its customers.

**FIGURE 4.4** A representation of the AWS Shared Responsibility Model




The AWS Shared Responsibility Model

## Managed Resources

A managed cloud service will "hide" all or some of the underlying configuration and administration work needed to keep things running, leaving you free to focus on the "business" end of your project. For example, an application running on an EC2 instance might need a database in the backend. You could install and configure a MySQL database engine on the instance itself, but you'd be responsible for patches, updates, and all the regular care and feeding (not to mention letting it out for its morning walks).

Alternatively, you could point your application to a stand-alone database you launch on Amazon's Relational Database Service (RDS). AWS is responsible for patching an RDS database and ensuring its data is secure and reliable. You only need to worry about populating the database and connecting it to your application.

RDS, therefore, is a good example of a partially managed service. How would the next level of managed service work? Look no further than Elastic Beanstalk, which hides just

about all the complexity of its runtime environment, leaving nothing for you to do beyond uploading your application code. Beanstalk handles the instances, storage, databases, and networking headaches—including ongoing patches and administration—invisibly.

## Unmanaged Resources

The most obvious example of an unmanaged AWS service is EC2. When you launch an EC2 instance, you're expected to care for the operating system and everything that's running on it exactly the way you would for a physical server in your on-premises data center. Still, even EC2 can't be said to be entirely unmanaged since the integrity of the physical server that hosts it is, of course, the responsibility of AWS.

Think of it as a sliding scale rather than a simple on-off switch. Some cloud operations will demand greater involvement from you and your administration team, and some will demand less. Use this simple rule of thumb: *if you can edit it, you own it*. The key—especially during a project's planning stages—is to be aware of your responsibilities and to always make security a critical priority.

## Service Health Status

As part of its end of the bargain, AWS makes regularly updated, region-by-region reports on the status of its services publicly available. Any service outages that could affect the performance of anyone's workload will appear on Amazon's Service Health Dashboard (`https://status.aws.amazon.com`)—often within a minute or two of the outage hitting.

While configuration errors are always a possible cause of a failure in your infrastructure, you should always make the Service Health Dashboard one of your first stops whenever you dive into a troubleshooting session.

## AWS Acceptable Use Policy

Because they're so easy to scale up, cloud computing services are powerful tools for accomplishing things no one had even dreamed of just a decade ago. But for that same reason, they're also potential weapons that can be used to commit devastating crimes.

The AWS Acceptable Use Policy (`https://aws.amazon.com/aup`) makes it abundantly clear that it does not permit the use of its infrastructure in any illegal, harmful, or offensive way. Amazon reserves the right to suspend or even terminate your use of its services should you engage in illegal, insecure, or abusive activities (including the sending of spam and related mass mailings). Even running penetration testing operations against your own AWS infrastructure can cause you trouble if you don't get explicit permission from Amazon in advance.

You should take the time to read the document and keep its terms in mind as you deploy on AWS.

# Summary

An AWS Region connects at least two Availability Zones located within a single geographic area into a low-latency network. Because of the default isolation of their underlying hardware, building secure, access-controlled regional environments is eminently possible.

An Availability Zone is a group of one or more independent (and fault-protected) data centers located within a single geographic region.

It's important to be aware of the region that's currently selected by your interface (either the AWS Management Console or a command-line terminal), as any operations you execute will launch specifically within the context of that region.

The design structure of Amazon's global system of regions allows you to build your infrastructure in ways that provide the best possible user experience while meeting your security and regulatory needs.

AWS offers some global resources whose use isn't restricted to any one region. Those include IAM, CloudFront, and S3.

You can connect to AWS service instances using their endpoint addresses, which will (generally) incorporate the host region's designation.

EC2 virtual machine instances are launched with an IP address issued from a network subnet that's associated with a single Availability Zone.

The principle of high availability can be used to make your infrastructure more resilient and reliable by launching parallel redundant instances in multiple Availability Zones.

AWS edge locations are globally distributed data servers that can store cached copies of AWS-based data from which—on behalf of the CloudFront service—they can be efficiently served to end users.

The elements of the AWS platform that you're expected to secure and maintain and those whose administration is managed by Amazon are defined by the AWS Shared Responsibility Model.

# Exam Essentials

**Understand the importance of resource isolation for cloud deployments.** Properly placing your cloud resources within the right region and Availability Zone—along with carefully setting appropriate access controls—can improve both application security and performance.

**Understand the role of autoscaling in a highly available deployment.** The scalability of AWS resources means you can automate the process of increasing or decreasing the scale of a deployment based on need. This can automate application recovery after a crash.

**Understand the role of load balancing in a highly available deployment.** The ability to automatically redirect incoming requests away from a nonfunctioning instance and to a backup replacement is managed by a load balancer.

**Understand the principles of the AWS Shared Responsibility Model.** AWS handles security and administration for its underlying physical infrastructure and for the full stack of all its managed services, while customers are responsible for everything else.

**Understand the principles of the AWS Acceptable Use Policy.** Using AWS resources to commit crimes or launch attacks against any individual or organization will result in account suspension or termination.

# CloudFront

Amazon CloudFront is a content delivery network (CDN) that helps deliver static and dynamic web content to users faster than just serving it out of an AWS Region. For example, if you're hosting a website from a single AWS Region, as a general rule, the farther a user is away from that region, the more network latency they'll encounter when accessing it. CloudFront solves this problem by caching your content in a number of data centers called *edge locations*. There are more than 150 edge locations spread out around the world on six continents.

CloudFront works by sending users to the edge location that will give them the best performance. Typically, this is the edge location that's physically closest to them. CloudFront also increases the availability of your content because copies of it are stored in multiple edge locations.

The more edge locations you use, the more redundancy you have and the better performance you can expect. As you might expect, the price of CloudFront increases as you utilize more edge locations. You can't select individual edge locations. Rather, you must choose from the following three options:

- United States, Canada, and Europe
- United States, Canada, Europe, Asia, and Africa
- All edge locations

To make your content available via CloudFront, you must create a distribution. A distribution defines the type of content you want CloudFront to cache, as well as the content's origin—where CloudFront should retrieve the content from. There are two types of distributions: Web and Real-Time Messaging Protocol (RTMP).

**Web** A Web distribution is the most common type. It's used for static and dynamic content such as web pages, graphic files, and live or on-demand streaming video. Users can access Web distributions via HTTP or HTTPS. When creating a Web distribution, you must specify an origin to act as the authoritative source for your content. An origin can be a web server or a public S3 bucket. You can't use nonpublic S3 buckets.

**RTMP** The Real-Time Messaging Protocol (RTMP) delivers streaming video or audio content to end users. To set up an RTMP distribution, you must provide both a media player and media files to stream, and these must be stored in S3 buckets.

# Summary

Virtual Private Cloud (VPC) provides the virtual network infrastructure for many AWS resources, most notably EC2. VPCs can connect to other networks, including the following:

- The internet via an internet gateway
- External, private networks via Direct Connect or a virtual private network (VPN)
- Other VPCs using VPC peering

The Route 53 service provides two distinct but related Domain Name System (DNS) services. Route 53 functions as a registrar for many top-level internet domain names (TLDs). You can register a new domain with Route 53 or transfer an existing one that you control. Route 53 also provides DNS hosting services. To use Route 53 with a public domain, you must create a public hosted zone. To use Route 53 for name resolution within a VPC, you must create a private hosted zone.

CloudFront is Amazon's content delivery network (CDN). It improves delivery of data to end users by storing content in edge locations around the world. When a user connects to a CloudFront distribution to retrieve content, CloudFront serves the content from the edge location that will give them the best performance.

# Exam Essentials

**Know the components of a VPC.** The key components of a VPC include at least one subnet, security groups, network access control lists (NACLs), and internet gateways.

**Understand the different options for connecting to resources in a VPC.** You can connect to resources in a VPC over the internet, a Direct Connect link, a VPC peering connection, or a virtual private network (VPN) connection.

**Understand the difference between a Route 53 public hosted zone and a private hosted zone.** A public hosted zone allows anyone on the internet to resolve records for the associated domain name. A private hosted zone allows resolution only from resources within the associated VPCs.

**Be able to select the best Route 53 routing policy for a given scenario.** All routing policies except the Simple routing policy can use health checks to route around failures. If you want to direct traffic to any available resource, Failover, Weighted, and Multivalue Answer routing policies will suffice. If performance is a concern, choose a Latency routing policy. If you need to direct users based on their specific location, use a Geolocation routing policy.

**Know how CloudFront improves the speed of content delivery.** CloudFront caches objects in edge locations around the world and automatically directs users to the edge location that will give them the best performance at any given time.

**Be able to identify scenarios where CloudFront would be appropriate.** CloudFront is designed to give users the fastest possible access to content regardless of their physical location. By caching content in edge locations that are distributed around the world, CloudFront helps ensure that your content is always close to your users.

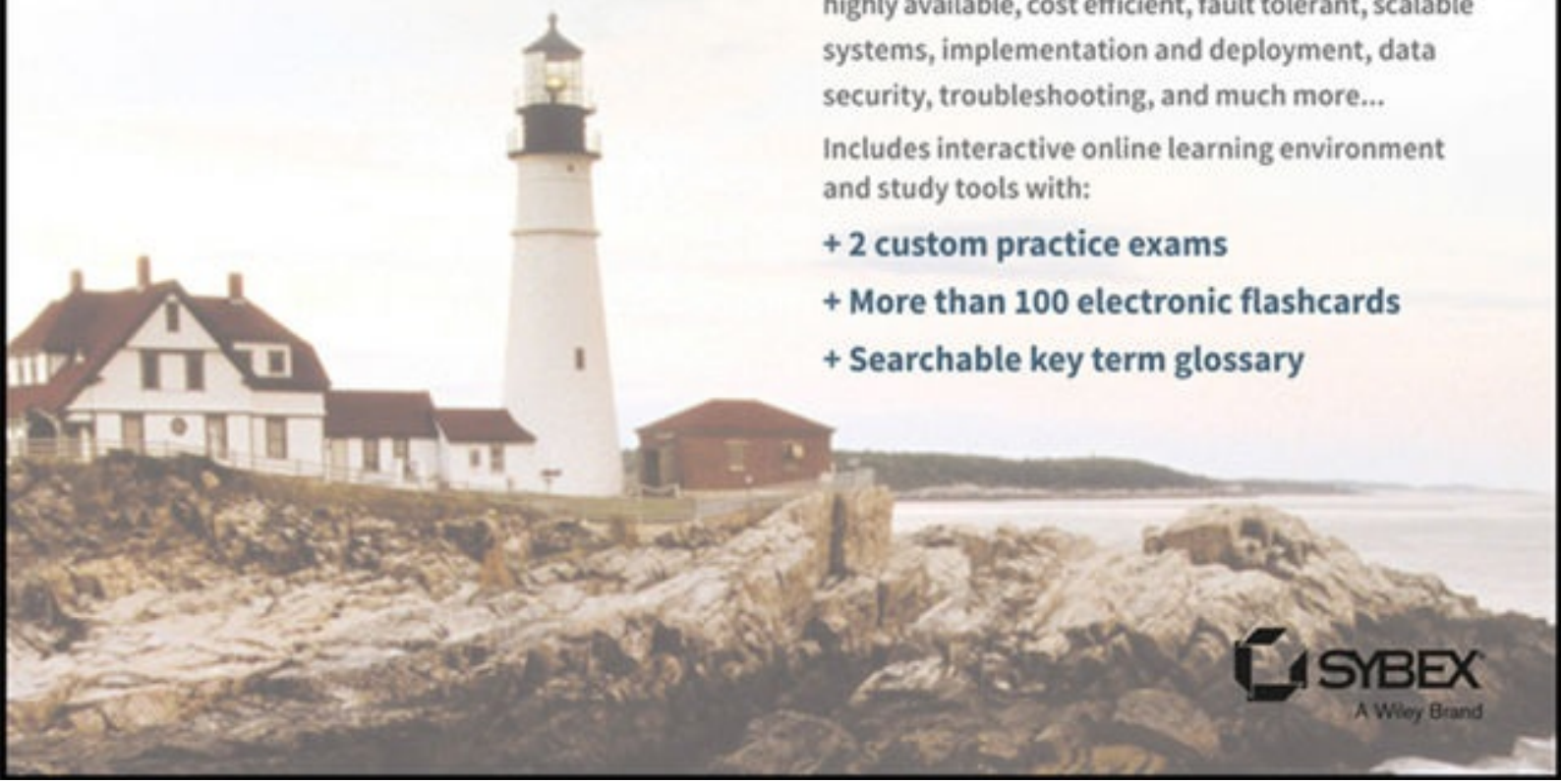
amazon web services

Joe Baron, Hisham Baz, Tim Bixler, Biff Gaut,
Kevin E. Kelly, Sean Senior, John Stamper

# AWS Certified Solutions Architect

## OFFICIAL STUDY GUIDE

**ASSOCIATE EXAM**



Covers exam objectives, including designing highly available, cost efficient, fault tolerant, scalable systems, implementation and deployment, data security, troubleshooting, and much more...

Includes interactive online learning environment and study tools with:

+ 2 custom practice exams
+ More than 100 electronic flashcards
+ Searchable key term glossary

SYBEX
A Wiley Brand

network gateways. In addition, organizations can extend their corporate data center networks to AWS by using hardware or software *virtual private network (VPN)* connections or dedicated circuits by using AWS Direct Connect.

### AWS Direct Connect

*AWS Direct Connect* allows organizations to establish a dedicated network connection from their data center to AWS. Using AWS Direct Connect, organizations can establish private connectivity between AWS and their data center, office, or colocation environment, which in many cases can reduce network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based VPN connections.

### Amazon Route 53

*Amazon Route 53* is a highly available and scalable Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications by translating human readable names, such as `www.example.com`, into the numeric IP addresses, such as 192.0.2.1, that computers use to connect to each other. Amazon Route 53 also serves as domain registrar, allowing you to purchase and manage domains directly from AWS.

## Storage and Content Delivery

AWS provides a variety of services to meet your storage needs, such as Amazon Simple Storage Service, Amazon CloudFront, and Amazon Elastic Block Store. This section provides an overview of the storage and content delivery services.

### Amazon Simple Storage Service (Amazon S3)

*Amazon Simple Storage Service (Amazon S3)* provides developers and IT teams with highly durable and scalable object storage that handles virtually unlimited amounts of data and large numbers of concurrent users. Organizations can store any number of objects of any type, such as HTML pages, source code files, image files, and encrypted data, and access them using HTTP-based protocols. Amazon S3 provides cost-effective object storage for a wide variety of use cases, including backup and recovery, nearline archive, big data analytics, disaster recovery, cloud applications, and content distribution.

### Amazon Glacier

*Amazon Glacier* is a secure, durable, and extremely low-cost storage service for data archiving and long-term backup. Organizations can reliably store large or small amounts of data for a very low cost per gigabyte per month. To keep costs low for customers, Amazon Glacier is optimized for infrequently accessed data where a retrieval time of several hours is suitable. Amazon S3 integrates closely with Amazon Glacier to allow organizations to choose the right storage tier for their workloads.

### Amazon Elastic Block Store (Amazon EBS)

*Amazon Elastic Block Store (Amazon EBS)* provides persistent block-level storage volumes for use with Amazon EC2 instances. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect organizations from component failure, offering high

availability and durability. By delivering consistent and low-latency performance, Amazon EBS provides the disk storage needed to run a wide variety of workloads.

### AWS Storage Gateway

*AWS Storage Gateway* is a service connecting an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and the AWS storage infrastructure. The service supports industry-standard storage protocols that work with existing applications. It provides low-latency performance by maintaining a cache of frequently accessed data on-premises while securely storing all of your data encrypted in Amazon S3 or Amazon Glacier.

### Amazon CloudFront

*Amazon CloudFront* is a content delivery web service. It integrates with other AWS Cloud services to give developers and businesses an easy way to distribute content to users across the world with low latency, high data transfer speeds, and no minimum usage commitments. Amazon CloudFront can be used to deliver your entire website, including dynamic, static, streaming, and interactive content, using a global network of edge locations. Requests for content are automatically routed to the nearest edge location, so content is delivered with the best possible performance to end users around the globe.

## Database Services

AWS provides fully managed relational and NoSQL database services, and in-memory caching as a service and a petabyte-scale data warehouse solution. This section provides an overview of the products that the database services comprise.

### Amazon Relational Database Service (Amazon RDS)

*Amazon Relational Database Service (Amazon RDS)* provides a fully managed relational database with support for many popular open source and commercial database engines. It's a cost-efficient service that allows organizations to launch secure, highly available, fault-tolerant, production-ready databases in minutes. Because Amazon RDS manages time-consuming administration tasks, including backups, software patching, monitoring, scaling, and replication, organizational resources can focus on revenue-generating applications and business instead of mundane operational tasks.

### Amazon DynamoDB

*Amazon DynamoDB* is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed database and supports both document and key/value data models. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, Internet of Things, and many other applications.

### Amazon Redshift

*Amazon Redshift* is a fast, fully managed, petabyte-scale data warehouse service that makes it simple and cost effective to analyze structured data. Amazon Redshift provides a standard SQL interface that lets organizations use existing business intelligence tools. By leveraging

If you are using Amazon S3 in a GET-intensive mode, such as a static website hosting, for best performance you should consider using an Amazon CloudFront distribution as a caching layer in front of your Amazon S3 bucket.

# Chapter 9
# Domain Name System (DNS) and Amazon Route 53

## THE AWS CERTIFIED SOLUTIONS ARCHITECT EXAM TOPICS COVERED IN THIS CHAPTER MAY INCLUDE, BUT ARE NOT LIMITED TO, THE FOLLOWING:

**Domain 1.0: Designing highly available, cost-efficient, fault-tolerant, scalable systems**

✓ **1.1 Identify and recognize cloud architecture considerations, such as fundamental components and effective designs.**

**Content may include the following:**

- How to design cloud services
- Planning and design
- Monitoring and logging
- Familiarity with:
  - Best practices for AWS architecture
  - Developing to client specifications, including pricing/cost (for example, on-demand vs. reserved vs. spot; RTO and RPO DR design)
  - Architectural trade-off decisions (for example, high availability vs. cost, Amazon Relational Database Service [RDS] vs. installing your own database on Amazon Elastic Compute Cloud—EC2)
  - Elasticity and scalability (for example, auto-scaling, SQS, ELB, CloudFront)

**Domain 3.0: Data Security**

✓ **3.1 Recognize and implement secure procedures for optimum cloud deployment and maintenance.**

✓ **3.2 Recognize critical disaster-recovery techniques and their implementation.**

- Amazon Route 53

# Amazon Route 53 Overview

Now that you have a foundational understanding of DNS and the different DNS record types, you can explore Amazon Route 53. *Amazon Route 53* is a highly available and scalable cloud DNS web service that is designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications.

Amazon Route 53 performs three main functions:

- ***Domain registration***—Amazon Route 53 lets you register domain names, such as `example.com`.
- ***DNS service***—Amazon Route 53 translates friendly domain names like `www.example.com` into IP addresses like `192.0.2.1`. Amazon Route 53 responds to DNS queries using a global network of authoritative DNS servers, which reduces latency. To comply with DNS standards, responses sent over User Datagram Protocol (UDP) are limited to 512 bytes in size. Responses exceeding 512 bytes are truncated, and the resolver must re-issue the request over TCP.
- ***Health checking***—Amazon Route 53 sends automated requests over the Internet to your application to verify that it's reachable, available, and functional.

You can use any combination of these functions. For example, you can use Amazon Route 53 as both your registrar and your DNS service, or you can use Amazon Route 53 as the DNS service for a domain that you registered with another domain registrar.

## Domain Registration

If you want to create a website, you first need to register the domain name. If you already registered a domain name with another registrar, you have the option to transfer the domain registration to Amazon Route 53. It isn't required to use Amazon Route 53 as your DNS service or to configure health checking for your resources.

Amazon Route 53 supports domain registration for a wide variety of generic TLDs (for example, `.com` and `.org`) and geographic TLDs (for example, `.be` and `.us`). For a complete list of supported TLDs, refer to the Amazon Route 53 Developer Guide at https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/.

## Domain Name System (DNS) Service

As stated previously, Amazon Route 53 is an authoritative DNS service that routes Internet traffic to your website by translating friendly domain names into IP addresses. When someone enters your domain name in a browser or sends you an email, a DNS request is forwarded to the nearest Amazon Route 53 DNS server in a global network of authoritative DNS servers. Amazon Route 53 responds with the IP address that you specified.

If you register a new domain name with Amazon Route 53, Amazon Route 53 will be automatically configured as the DNS service for the domain, and a *hosted zone* will be created for your domain. You add resource record sets to the hosted zone, which define how you want Amazon Route 53 to respond to DNS queries for your domain (for example, with the IP address for a web server, the IP address for the nearest Amazon CloudFront edge location, or

the IP address for an Elastic Load Balancing load balancer).

If you registered your domain with another domain registrar, that registrar is probably providing the DNS service for your domain. You can transfer DNS service to Amazon Route 53, with or without transferring registration for the domain.

If you're using Amazon CloudFront, Amazon Simple Storage Service (Amazon S3), or Elastic Load Balancing, you can configure Amazon Route 53 to route Internet traffic to those resources.

## Hosted Zones

A *hosted zone* is a collection of resource record sets hosted by Amazon Route 53. Like a traditional DNS zone file, a hosted zone represents resource record sets that are managed together under a single domain name. Each hosted zone has its own metadata and configuration information.

There are two types of hosted zones: private and public. A *private hosted zone* is a container that holds information about how you want to route traffic for a domain and its subdomains within one or more Amazon Virtual Private Clouds (Amazon VPCs). A *public hosted zone* is a container that holds information about how you want to route traffic on the Internet for a domain (for example, `example.com`) and its subdomains (for example, `apex.example.com` and `acme.example.com`).

The resource record sets contained in a hosted zone must share the same suffix. For example, the `example.com` hosted zone can contain resource record sets for the `www.example.com` and `www.aws.example.com` subdomains, but it cannot contain resource record sets for a `www.example.ca` subdomain.

 You can use Amazon S3 to host your static website at the hosted zone (for example, `domain.com`) and redirect all requests to a subdomain (for example, `www.domain.com`). Then, in Amazon Route 53, you can create an alias resource record that sends requests for the root domain to the Amazon S3 bucket.

 Use an alias record, not a CNAME, for your hosted zone. CNAMEs are not allowed for hosted zones in Amazon Route 53.

 Do not use A records for subdomains (for example, `www.domain.com`), as they refer to hardcoded IP addresses. Instead, use Amazon Route 53 alias records or traditional CNAME records to always point to the right resource, wherever your site is hosted, even when the physical server has changed its IP address.

- The application's failover environment (for example, `fail.domain.com`) has an Amazon Route 53 alias record that points to an Amazon CloudFront distribution of an Amazon S3 bucket hosting a static version of the application.
- The application's subdomain (for example, `www.domain.com`) has an Amazon Route 53 alias record that points to `prod.domain.com` (as primary target) and `fail.domain .com` (as secondary target) using a failover routing policy. This ensures `www.domain.com` routes to the production load balancers if at least one of them is healthy or the "fail whale" if all of them appear to be unhealthy.
- The application's hosted zone (for example, `domain.com`) has an Amazon Route 53 alias record that redirects requests to `www.domain.com` using an Amazon S3 bucket of the same name.
- Application content (both static and dynamic) can be served using Amazon CloudFront. This ensures that the content is delivered to clients from Amazon CloudFront edge locations spread all over the world to provide minimal latency. Serving dynamic content from a Content Delivery Network (CDN), where it is cached for short periods of time (that is, several seconds), takes the load off of the application and further improves its latency and responsiveness.
- The application is deployed in multiple AWS regions, protecting it from a regional outage.

# Storage and Content Delivery

This section covers two additional storage and content delivery services that are important for a Solutions Architect to understand: Amazon CloudFront and AWS Storage Gateway.

## Amazon CloudFront

*Amazon CloudFront* is a global Content Delivery Network (CDN) service. It integrates with other AWS products to give developers and businesses an easy way to distribute content to end users with low latency, high data transfer speeds, and no minimum usage commitments.

### Overview

A *Content Delivery Network (CDN)* is a globally distributed network of caching servers that speed up the downloading of web pages and other content. CDNs use Domain Name System (DNS) *geo-location* to determine the geographic location of each request for a web page or other content, then they serve that content from edge caching servers closest to that location instead of the original web server. A CDN allows you to increase the scalability of a website or mobile application easily in response to peak traffic spikes. In most cases, using a CDN is completely transparent—end users simply experience better website performance, while the load on your original website is reduced.

Amazon CloudFront is AWS CDN. It can be used to deliver your web content using Amazon's global network of *edge locations*. When a user requests content that you're serving with Amazon CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so content is delivered with the best possible performance. If the content is already in the edge location with the lowest latency, Amazon CloudFront delivers it immediately. If the content is not currently in that edge location, Amazon CloudFront retrieves it from the *origin server*, such as an Amazon Simple Storage Service (Amazon S3) bucket or a web server, which stores the original, definitive versions of your files.

Amazon CloudFront is optimized to work with other AWS cloud services as the origin server, including Amazon S3 buckets, Amazon S3 static websites, Amazon Elastic Compute Cloud (Amazon EC2), and Elastic Load Balancing. Amazon CloudFront also works seamlessly with any non-AWS origin server, such as an existing on-premises web server. Amazon CloudFront also integrates with Amazon Route 53.

Amazon CloudFront supports all content that can be served over HTTP or HTTPS. This includes any popular static files that are a part of your web application, such as HTML files, images, JavaScript, and CSS files, and also audio, video, media files, or software downloads. Amazon CloudFront also supports serving dynamic web pages, so it can actually be used to deliver your entire website. Finally, Amazon CloudFront supports media *streaming*, using both HTTP and RTMP.

### Amazon CloudFront Basics

There are three core concepts that you need to understand in order to start using CloudFront: distributions, origins, and cache control. With these concepts, you can easily use CloudFront to speed up delivery of static content from your websites.

**Distributions** To use Amazon CloudFront, you start by creating a *distribution*, which is identified by a DNS domain name such as d111111abcdef8.cloudfront.net. To serve files from Amazon CloudFront, you simply use the distribution domain name in place of your website's domain name; the rest of the file paths stay unchanged. You can use the Amazon CloudFront distribution domain name as-is, or you can create a user-friendly DNS name in your own domain by creating a CNAME record in Amazon Route 53 or another DNS service. The CNAME is automatically redirected to your Amazon CloudFront distribution domain name.

**Origins** When you create a distribution, you must specify the DNS domain name of the *origin*—the Amazon S3 bucket or HTTP server—from which you want Amazon CloudFront to get the definitive version of your objects (web files). For example:

- **Amazon S3 bucket:** `myawsbucket.s3.amazonaws.com`
- **Amazon EC2 instance:** `ec2-203-0-113-25.compute-1.amazonaws.com`
- **Elastic Load Balancing load balancer:** `my-load-balancer-1234567890.us-west-2.elb.amazonaws.com`
- **Website URL:** `mywebserver.mycompanydomain.com`

**Cache Control** Once requested and served from an edge location, objects stay in the cache until they expire or are evicted to make room for more frequently requested content. By default, objects expire from the cache after 24 hours. Once an object expires, the next request results in Amazon CloudFront forwarding the request to the origin to verify that the object is unchanged or to fetch a new version if it has changed.

Optionally, you can control how long objects stay in an Amazon CloudFront cache before expiring. To do this, you can choose to use Cache-Control headers set by your origin server or you can set the minimum, maximum, and default *Time to Live (TTL)* for objects in your Amazon CloudFront distribution.

You can also remove copies of an object from all Amazon CloudFront edge locations at any time by calling the *invalidation* Application Program Interface (API). This feature removes the object from every Amazon CloudFront edge location regardless of the expiration period you set for that object on your origin server. The invalidation feature is designed to be used in unexpected circumstances, such as to correct an error or to make an unanticipated update to a website, not as part of your everyday workflow.

Instead of invalidating objects manually or programmatically, it is a best practice to use a version identifier as part of the object (file) path name. For example:

- **Old file:** `assets/v1/css/narrow.css`
- **New file:** `assets/v2/css/narrow.css`

When using versioning, users always see the latest content through Amazon CloudFront when you update your site without using invalidation. Old versions will expire from the cache automatically.

### Amazon CloudFront Advanced Features

CloudFront can do much more than simply serve static web files. To start using CloudFront's advanced features, you will need to understand how to use cache behaviors, and how to

restrict access to sensitive content.

**Dynamic Content, Multiple Origins, and Cache Behaviors** Serving static assets, such as described previously, is a common way to use a CDN. An Amazon CloudFront distribution, however, can easily be set up to serve dynamic content in addition to static content and to use more than one origin server. You control which requests are served by which origin and how requests are cached using a feature called *cache behaviors*.

A cache behavior lets you configure a variety of Amazon CloudFront functionalities for a given URL path pattern for files on your website. For example see Figure 11.1. One cache behavior applies to all PHP files in a web server (dynamic content), using the path pattern `*.php`, while another behavior applies to all JPEG images in another origin server (static content), using the path pattern `*.jpg`.

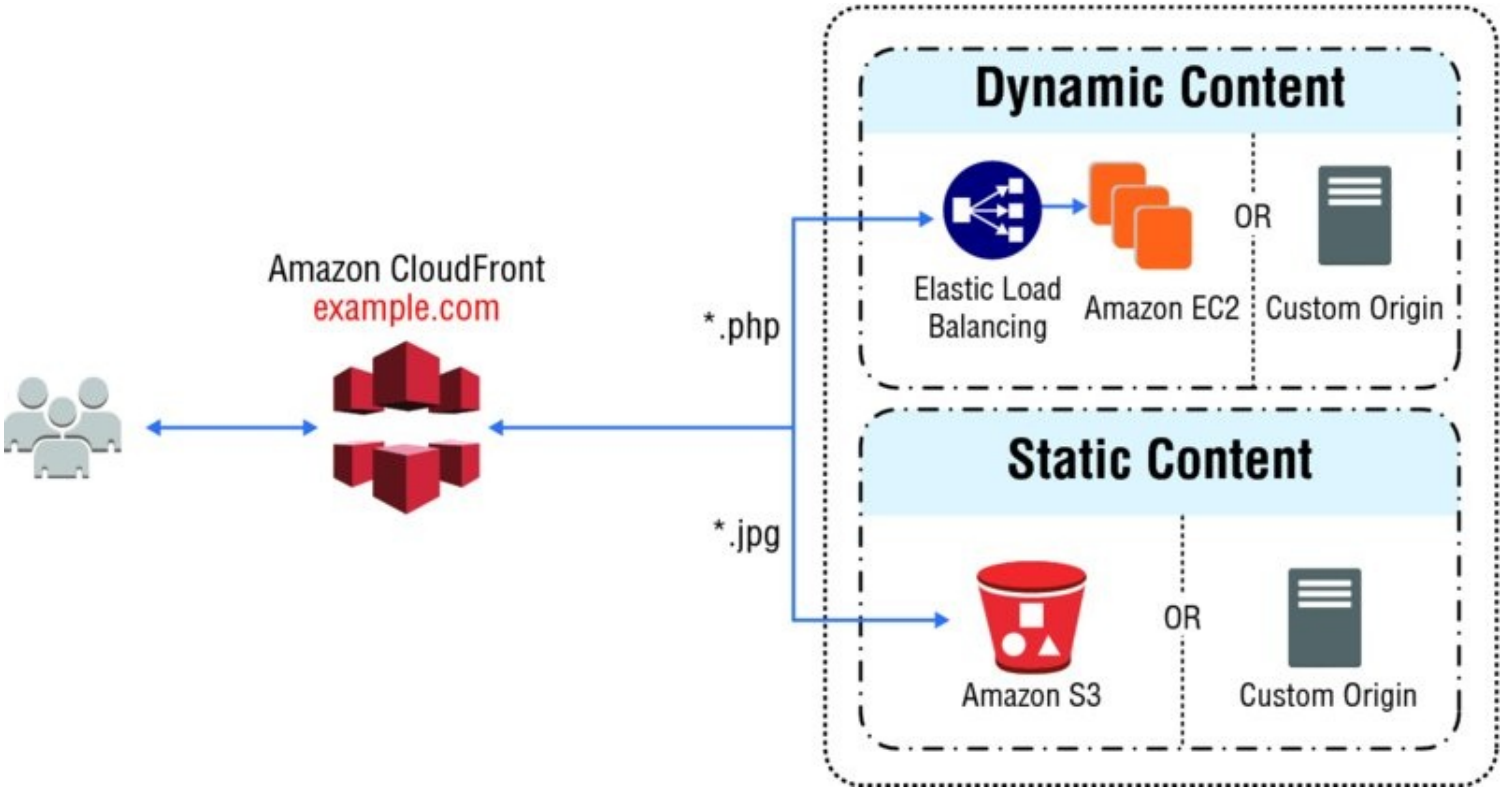



**FIGURE 11.1** Delivering static and dynamic content

The functionality you can configure for each cache behavior includes the following:

- The path pattern
- Which origin to forward your requests to
- Whether to forward query strings to your origin
- Whether accessing the specified files requires signed URLs
- Whether to require HTTPS access
- The amount of time that those files stay in the Amazon CloudFront cache (regardless of the value of any Cache-Control headers that your origin adds to the files)

Cache behaviors are applied in order; if a request does not match the first path pattern, it drops down to the next path pattern. Normally the last path pattern specified is `*` to match all files.

**Whole Website** Using cache behaviors and multiple origins, you can easily use Amazon CloudFront to serve your whole website and to support different behaviors for different client devices.

**Private Content** In many cases, you may want to restrict access to content in Amazon CloudFront to only selected requestors, such as paid subscribers or to applications or users in your company network. Amazon CloudFront provides several mechanisms to allow you to serve private content. These include:

**Signed URLs** Use URLs that are valid only between certain times and optionally from certain IP addresses.

**Signed Cookies** Require authentication via public and private key pairs.

**Origin Access Identities (OAI)** Restrict access to an Amazon S3 bucket only to a special Amazon CloudFront user associated with your distribution. This is the easiest way to ensure that content in a bucket is only accessed by Amazon CloudFront.

### Use Cases

There are several use cases where Amazon CloudFront is an excellent choice, including, but not limited to:

**Serving the Static Assets of Popular Websites** Static assets such as images, CSS, and JavaScript traditionally make up the bulk of requests to typical websites. Using Amazon CloudFront will speed up the user experience and reduce load on the website itself.

**Serving a Whole Website or Web Application** Amazon CloudFront can serve a whole website containing both dynamic and static content by using multiple origins, cache behaviors, and short TTLs for dynamic content.

**Serving Content to Users Who Are Widely Distributed Geographically** Amazon CloudFront will improve site performance, especially for distant users, and reduce the load on your origin server.

**Distributing Software or Other Large Files** Amazon CloudFront will help speed up the download of these files to end users.

**Serving Streaming Media** Amazon CloudFront helps serve streaming media, such as audio and video.

There are also use cases where CloudFront is not appropriate, including:

**All or Most Requests Come From a Single Location** If all or most of your requests come from a single geographic location, such as a large corporate campus, you will not take advantage of multiple edge locations.

**All or Most Requests Come Through a Corporate VPN** Similarly, if your users connect via a corporate Virtual Private Network (VPN), even if they are distributed, user requests appear to CloudFront to originate from one or a few locations. These use cases will generally not see benefit from using Amazon CloudFront.

## AWS Storage Gateway

AWS Storage Gateway is a service connecting an on-premises software appliance with cloud-

## Exam Essentials

**Know the basic use cases for amazon CloudFront.** Know when to use Amazon CloudFront (for popular static and dynamic content with geographically distributed users) and when not to (all users at a single location or connecting through a corporate VPN).

**Know how amazon CloudFront works.** Amazon CloudFront optimizes downloads by using geolocation to identify the geographical location of users, then serving and caching content at the edge location closest to each user to maximize performance.

**Know how to create an amazon CloudFront distribution and what types of origins are supported.** To create a distribution, you specify an origin and the type of distribution, and Amazon CloudFront creates a new domain name for the distribution. Origins supported include Amazon S3 buckets or static Amazon S3 websites and HTTP servers located in Amazon EC2 or in your own data center.

**Know how to use amazon CloudFront for dynamic content and multiple origins.** Understand how to specify multiple origins for different types of content and how to use cache behaviors and path strings to control what content is served by which origin.

**Know what mechanisms are available to serve private content through amazon CloudFront**. Amazon CloudFront can serve private content using Amazon S3 Origin Access Identifiers, signed URLs, and signed cookies.

**Know the three configurations of AWS storage gateway and their use cases.** Gateway-Cached volumes expand your on-premises storage into Amazon S3 and cache frequently used files locally. Gateway-Stored values keep all your data available locally at all times and also replicate it asynchronously to Amazon S3. Gateway-VTL enables you to keep your current backup tape software and processes while eliminating physical tapes by storing your data in the cloud.

**Understand the value of AWS Directory Service.** AWS Directory Service is designed to reduce identity management tasks, thereby allowing you to focus more of your time and resources on your business.

**Know the AWS Directory Service Directory types.** AWS Directory Service offers three directory types:

- AWS Directory Service for Microsoft Active Directory (Enterprise Edition), also referred to as Microsoft AD
- Simple AD
- AD Connector

**Know when you should use AWS Directory Service for Microsoft Active Directory.** You should use Microsoft Active Directory if you have more than 5,000 users or need a trust relationship set up between an AWS hosted directory and your on-premises directories.

**Understand key management.** Key management is the management of cryptographic keys within a cryptosystem. This includes dealing with the generation, exchange, storage, use,

# AWS®

# Certified Developer

## Official Study Guide

### Associate (DVA-C01) Exam

one of your buckets and allow the user to add, update, and delete objects. You can grant them access with a user policy.

Now we will discuss the differences between IAM policies and Amazon S3 bucket policies. Both are used for access control, and they are both written in JSON using the AWS access policy language. However, unlike Amazon S3 bucket policies, IAM policies specify what actions are allowed or denied on what AWS resources (such as, allow `ec2:TerminateInstance` on the Amazon EC2 instance with `instance_id=i8b3620ec`). You attach IAM policies to IAM users, groups, or roles, which are then subject to the permissions that you have defined. Instead of attaching policies to the users, groups, or roles, bucket policies are attached to a specific resource, such as an Amazon S3 bucket.

### Managing Access with Access Control Lists

*Access with access control lists* (ACLs) are resource-based access policies that you can use to manage access to your buckets and objects, including granting basic read/write permissions to other accounts.

There are limits to managing permissions using ACLs. For example, you can grant permissions only to other accounts; you cannot grant permissions to users in your account. You cannot grant conditional permissions, nor can you explicitly deny permissions using ACLs.

ACLs are suitable only for specific scenarios (for example, if a bucket owner allows other accounts to upload objects), and permissions to these objects can be managed only using an object ACL by the account that owns the object.


You can only grant access to other accounts using ACLs—not users in your own account.

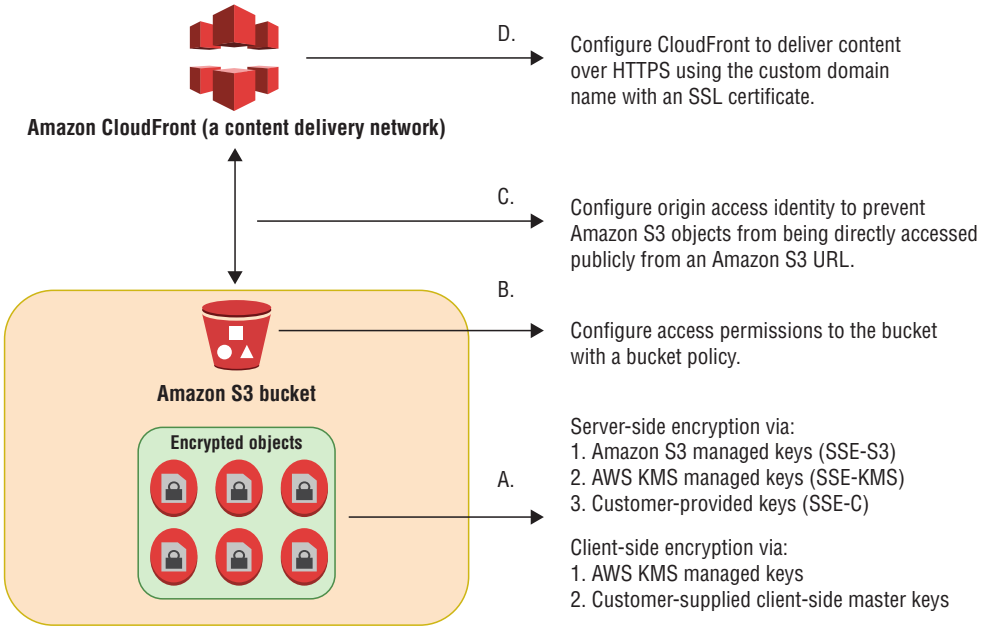
### Defense in Depth—Amazon S3 Security

Amazon S3 provides comprehensive security and compliance capabilities that meet the most stringent regulatory requirements, and it gives you flexibility in the way that you manage data for cost optimization, access control, and compliance. With this flexibility, however, comes the responsibility of ensuring that your content is secure.

You can use an approach known as *defense in depth* in Amazon S3 to secure your data. This approach uses multiple layers of security to ensure redundancy if one of the multiple layers of security fails.

Figure 3.14 represents defense in depth visually. It contains several Amazon S3 objects (A) in a single Amazon S3 bucket (B). You can encrypt these objects on the server side or the client side, and you can also configure the bucket policy such that objects are accessible only through Amazon CloudFront, which you can accomplish through an origin access identity (C). You can then configure Amazon CloudFront to deliver content only over HTTPS in addition to using your own domain name (D).

**FIGURE 3.14** Defense in depth on Amazon S3




To meet defense in depth requirements on Amazon S3:

- Data must be encrypted at rest and during transit.
- Data must be accessible only by a limited set of public IP addresses.
- Data must not be publicly accessible directly from an Amazon S3 URL.
- A domain name is required to consume the content.

You can apply policies to Amazon S3 buckets so that only users with appropriate permissions are allowed to access the buckets. Anonymous users (with public-read/public-read-write permissions) and authenticated users without the appropriate permissions are prevented from accessing the buckets.

You can also secure access to objects in Amazon S3 buckets. The objects in Amazon S3 buckets can be encrypted at rest and during transit to provide end-to-end security from the source (in this case, Amazon S3) to your users.

## Query String Authentication

You can provide authentication information using *query string parameters*. Using query parameters to authenticate requests is useful when expressing a request entirely in a URL. This method is also referred to as *presigning* a URL.

With presigned URLs, you can grant temporary access to your Amazon S3 resources. For example, you can embed a presigned URL on your website, or alternatively use it in a command line client (such as Curl), to download objects.

The following is an example presigned URL:

```
https://s3.amazonaws.com/examplebucket/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request
&X-Amz-Date=20130721T201207Z
&X-Amz-Expires=86400
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

In the example URL, note the following:

- The line feeds are added for readability.
- The `X-Amz-Credential` value in the URL shows the `/` character only for readability. In practice, it should be encoded as `%2F`.

## Hosting a Static Website

If your website contains static content and optionally client-side scripts, then you can host your *static website* directly in Amazon S3 without the use of web-hosting servers.

To host a static website, you configure an Amazon S3 bucket for website hosting and upload your website content to the bucket. The website is then available at the AWS Region–specific website endpoint of the bucket in one of the following formats:

```
<bucket-name>.s3-website-<AWS-region>.amazonaws.com
<bucket-name>.s3-website.<AWS-region>.amazonaws.com
```

Instead of accessing the website by using an Amazon S3 website endpoint, use your own domain (for instance, `example.com`) to serve your content. The following steps allow you to configure your own domain:

1. Register your domain with the registrar of your choice. You can use Amazon Route 53 to register your domain name or any other third-party domain registrar.
2. Create your bucket in Amazon S3 and upload your static website content.
3. Point your domain to your Amazon S3 bucket using either of the following as your DNS provider:
   - Amazon Route 53
   - Your third-party domain name registrar

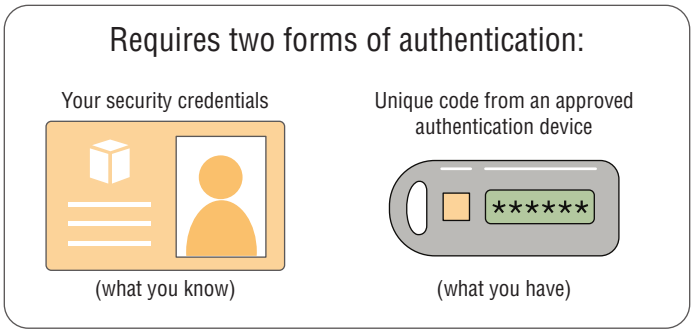
Amazon S3 does not support server-side scripting or dynamic content. We discuss other AWS options for that throughout this study guide.


Static websites can be hosted in Amazon S3.

## MFA Delete

*MFA* is another way to control deletes on your objects in Amazon S3. It does so by adding another layer of protection against unintentional or malicious deletes, requiring an authorized request against Amazon S3 to delete the object. MFA also requires a unique code from a token or an authentication device (virtual or hardware). These devices provide a unique code that will then allow you to delete the object. Figure 3.15 shows what would be required for a user to execute a delete operation on an object when MFA is enabled.

**FIGURE 3.15** MFA Delete




## Cross-Region Replication

*Cross-region replication* (CRR) is a bucket-level configuration that enables automatic, asynchronous copying of objects across buckets in different AWS Regions. We refer to these buckets as the *source* bucket and *destination* bucket. These buckets can be owned by different accounts.

To activate this feature, add a replication configuration to your source bucket to direct Amazon S3 to replicate objects according to the configuration. In the replication configuration, provide information including the following:

- The destination bucket
- The objects that need to be replicated
- Optionally, the destination storage class (otherwise the source storage class will be used)

The replicas that are created in the destination bucket will have these same characteristics as the source objects:

- Key names
- Metadata
- Storage class (unless otherwise specified)
- Object ACL

All data is encrypted in transit across AWS Regions using SSL.

You can replicate objects from a source bucket to only one destination bucket. After Amazon S3 replicates an object, the object cannot be replicated again. For example, even after you change the destination bucket in an existing replication configuration, Amazon S3 will not replicate it again.

After Amazon S3 replicates an object using CRR, the object cannot be replicated again (such as to another destination bucket).

Requirements for CRR include the following:

- Versioning is enabled for both the source and destination buckets.
- Source and destination buckets must be in different AWS Regions.
- Amazon S3 must be granted appropriate permissions to replicate files.

## VPC Endpoints

A *virtual private cloud (VPC) endpoint* enables you to connect your VPC privately to Amazon S3 without requiring an internet gateway, *network address translation (NAT)* device, *virtual private network (VPN)* connection, or *AWS Direct Connect* connection. Instances in your VPC do not require public IP addresses to communicate with the resources in the service. Traffic between your VPC and Amazon S3 does not leave the Amazon network.

Amazon S3 uses a gateway type of VPC endpoint. The gateway is a target for a specified route in your route table, used for traffic destined for a supported AWS service. These endpoints are easy to configure, are highly reliable, and provide a secure connection to Amazon S3 that does not require a gateway or NAT instance.

Amazon EC2 instances running in private subnets of a VPC can have controlled access to Amazon S3 buckets, objects, and API functions that are in the same region as the VPC. You can use an Amazon S3 bucket policy to indicate which VPCs and which VPC endpoints have access to your Amazon S3 buckets.

## Using the AWS SDKs, AWS CLI, and AWS Explorers

You can use the AWS SDKs when developing applications with Amazon S3. The AWS SDKs simplify your programming tasks by wrapping the underlying REST API. The AWS Mobile SDKs and the AWS Amplify JavaScript library are also available for building connected mobile and web applications using AWS. In addition to AWS SDKs, AWS explorers are available for Visual Studio and Eclipse for *Java Integrated Development Environment* (IDE). In this case, the SDKs and AWS explorers are available bundled together as AWS Toolkits. You can also use the AWS CLI to manage Amazon S3 buckets and objects.

AWS has deprecated SOAP support over HTTP, but it is still available over HTTPS. New Amazon S3 features will not be supported over SOAP. We recommend that you use

either the REST API or the AWS SDKs for any new development and migrate any existing SOAP calls when you are able.

## Making Requests

Every interaction with Amazon S3 is either authenticated or anonymous. *Authentication* is the process of verifying the identity of the requester trying to access an AWS product (you are who you say you are, and you are allowed to do what you are asking to do). Authenticated requests must include a signature value that authenticates the request sender, generated in part from the requester's AWS access keys. If you are using the AWS SDK, the libraries compute the signature from the keys that you provide. If you make direct REST API calls in your application, however, you must write code to compute the signature and add it to the request.

## Stateless and Serverless Applications

Amazon S3 provides developers with secure, durable, and highly scalable object storage that can be used to decouple storage for use in *serverless applications*. Developers can also use Amazon S3 for storing and sharing state in *stateless applications*.

Developers on AWS are regularly moving shared file storage to Amazon S3 for stateless applications. This is a common method for decoupling your compute and storage and increasing the ability to scale your application by decoupling that storage. We will discuss stateless and serverless applications throughout this study guide.

## Data Lake

Traditional data storage can no longer provide the agility and flexibility required to handle the volume, velocity, and variety of data used by today's applications. Because of this, many organizations are shifting to a *data lake* architecture.

A *data lake* is an architectural approach that allows you to store massive amounts of data in a central location for consumption by multiple applications. Because data can be stored as is, there is no need to convert it to a predefined schema, and you no longer need to know what questions to ask of your data beforehand.

Amazon S3 is a common component of a data lake solution on the cloud, and it can complement your other storage solutions. If you move to a data lake, you are essentially separating compute and storage, meaning that you are going to build and scale your storage and compute separately. You can take storage that you currently have on premises or in your data center and instead use Amazon S3, which then allows you to scale and build your compute in any desired configuration, regardless of your storage.

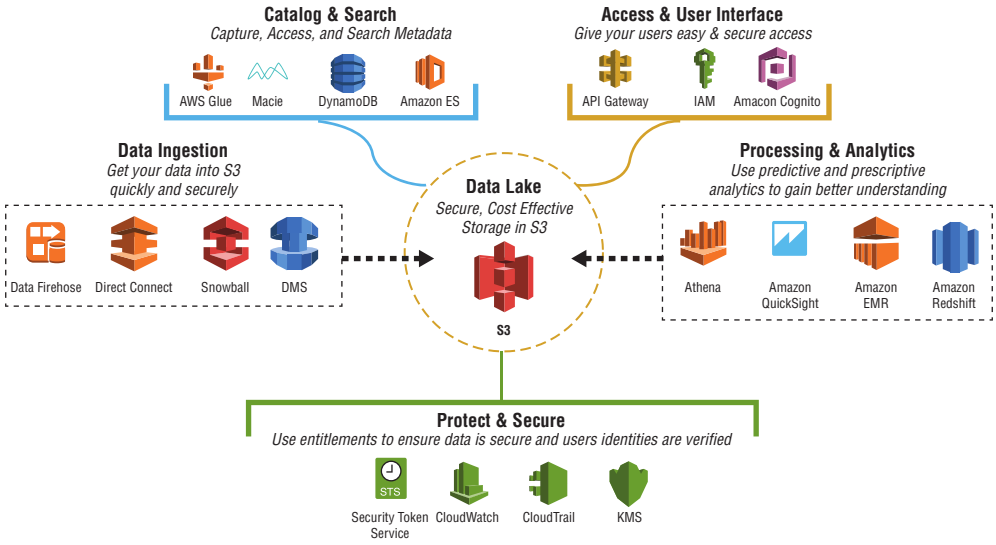
That design pattern is different from most applications available today, where the storage is tied to the compute. When you separate those two features and instead use a data lake, you achieve an agility that allows you to invent new types of applications while you are managing your storage as an independent entity.

In addition, Amazon S3 lets you grow and scale in a virtually unlimited fashion. You do not have to take specific actions to expand your storage capacity—it grows automatically with your data.

In the data lake diagram shown in Figure 3.16, you will see how to use Amazon S3 as a highly available and durable central storage repository. From there, a virtually unlimited

number of services and applications, both on premises and in the cloud, can take advantage of using Amazon S3 as a data lake.

**FIGURE 3.16** Data lakes




Customers often set up a data lake as part of their migration to the cloud so that they can access their data from new applications on the cloud, migrated applications to the cloud, and applications that have not yet been migrated to the cloud.

## Performance

There are a number of actions that Amazon S3 takes by default to help you achieve high levels of performance. Amazon S3 automatically scales to thousands of requests per second per prefix based on your steady state traffic. Amazon S3 will automatically partition your prefixes within hours, adjusting to increases in request rates.

### Consideration for Workloads

To optimize the use of Amazon S3 mixed or `GET`-intensive workloads, you must become familiar with best practices for performance optimization.

**Mixed request types** If your requests are typically a mix of `GET`, `PUT`, `DELETE`, and `GET` Bucket (list objects), choosing appropriate key names for your objects ensures better performance by providing low-latency access to the Amazon S3 index.

**`GET`-intensive workloads** If the bulk of your workload consists of `GET` requests, you may want to use Amazon CloudFront, a content delivery service (discussed later in this chapter).

### Tips for Object Key Naming

The way that you name your keys in Amazon S3 can affect the data access patterns, which may directly impact the performance of your application.

It is a best practice at AWS to design for performance from the start. Even though you may be developing a new application, that application is likely to grow over time. If you anticipate your application growing to more than approximately 1,000 requests per second (including both `PUT`s and `GET`s on your object), you will want to consider using a three- or four-character hash in your key names.

If you anticipate your application receiving fewer than 1,000 requests per second and you don't see a lot of traffic in your storage, then you do not need to implement this best practice. Your application will still benefit from Amazon S3's default performance.


In the past, customers would also add entropy in their key names. Because of recent Amazon S3 performance enhancements, most customers no longer need to worry about introducing entropy in key names.

**Example 1:** Random Hash

```
examplebucket/232a-2017-26-05-15-00-00/cust1234234/photo1.jpg
examplebucket/7b54-2017-26-05-15-00-00/cust3857422/photo2.jpg
examplebucket/921c-2017-26-05-15-00-00/cust1248473/photo2.jpg
```


A random hash should come before patterns, such as dates and sequential IDs.

Using a *naming hash* can improve the performance of heavy-traffic applications. Object keys are stored in an index in all regions. If you're constantly writing the same key prefix over and over again (for example, a key with the current year), all of your objects will be close to each other within the same partition in the index. When your application experiences an increase in traffic, it will be trying to read from the same section of the index, resulting in decreased performance as Amazon S3 tries to spread out your data to achieve higher levels of throughput.


Always first ensure that your application can accommodate a naming hash.

By putting the hash at the beginning of your key name, you are adding randomness. You could hash the key name and place it at the beginning of your object right after the bucket name. This will ensure that your data will be spread across different partitions and allow you to grow to a higher level of throughput without experiencing a re-indexing slowdown if you go above peak traffic volumes.

**Example 2:** Naming Hash

```
examplebucket/animations/232a-2017-26-05-15-00/cust1234234/animation1.obj
examplebucket/videos/ba65-2017-26-05-15-00/cust8474937/video2.mpg
examplebucket/photos/8761-2017-26-05-15-00/cust1248473/photo3.jpg
```

In this second example, imagine that you are storing a lot of animations, videos, and photos in Amazon S3. If you know that you are going to have a lot of traffic to those individual prefixes, you can add your hash after the prefix. That allows you to write prefixes into your lifecycle policies or perform *list API* calls against a particular prefix. You are still getting the performance benefit by adding the hash to your key name, but now you can also use the prefix as necessary.

This example allows you to balance the need to list your objects and organize them against the need to spread your data across different partitions for performance.

### Amazon S3 Transfer Acceleration

*Amazon S3 Transfer Acceleration* is a feature that optimizes throughput when transferring larger objects across larger geographic distances. Amazon S3 Transfer Acceleration uses Amazon CloudFront edge locations to assist you in uploading your objects more quickly in cases where you are closer to an edge location than to the region to which you are transferring your files.

Instead of using the public internet to upload objects from Southeast Asia, across the globe to Northern Virginia, take advantage of the global *Amazon content delivery network* (CDN). AWS has edge locations around the world, and you upload your data to the edge location closest to your location. This way, you are traveling across the AWS network backbone to your destination region, instead of across the public internet. This option might give you a significant performance improvement and better network consistency than the public internet.

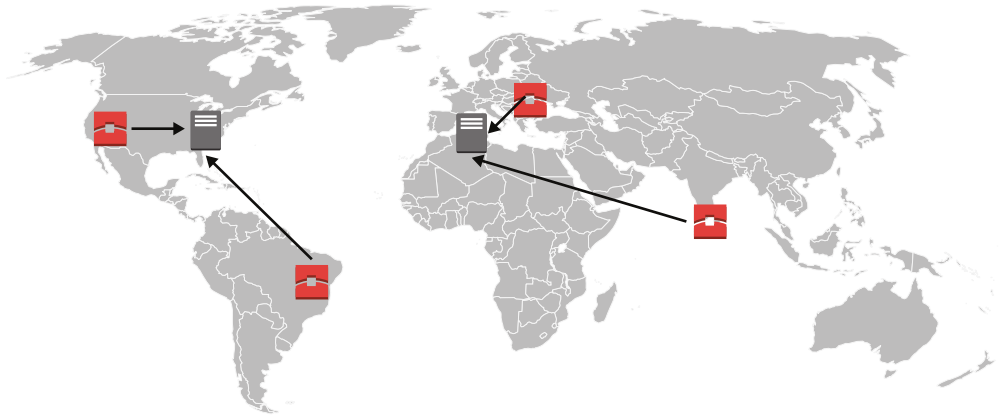
To implement Amazon S3 Transfer Acceleration, you do not need to make any changes to your application. It is enabled by performing the following steps:

1. Enable Transfer Acceleration on a bucket that conforms to DNS naming requirements and does not contain periods (.).
2. Transfer data to and from the acceleration-enabled bucket by using one of the s3-accelerate endpoint domain names.

There is a small fee for using Transfer Acceleration. If your speed using Transfer Acceleration is no faster than it would have been going over the public internet, however, there is no additional charge.

The further you are from a particular region, the more benefit you will derive from transferring your files more quickly by uploading to a closer edge location. Figure 3.17 shows how accessing an edge location can reduce the latency for your users, as opposed to accessing content from a region that is farther away.

**FIGURE 3.17** Using an AWS edge location



### Multipart Uploads

When uploading a large object to Amazon S3 in a single-threaded manner, it can take a significant amount of time to complete. The multipart upload API enables you to upload large objects in parts to speed up your upload by doing so in parallel.

To use multipart upload, you first break the object into smaller parts, parallelize the upload, and then submit a manifest file telling Amazon S3 that all parts of the object have been uploaded. Amazon S3 will then assemble all of those individual pieces to a single Amazon S3 object.

Multipart upload can be used for objects ranging from 5 MB to 5 TB in size.

### Range GETs

*Range GETs* are similar to multipart uploads, but in reverse. If you are downloading a large object and tracking the offsets, use range `GET`s to download the object as multiple parts instead of a single part. You can then download those parts in parallel and potentially see an improvement in performance.

### Amazon CloudFront

Using a CDN like Amazon CloudFront, you may achieve lower latency and higher-throughput performance. You also will not experience as many requests to Amazon S3 because your content will be cached at the edge location. Your users will also experience the performance improvement of having cached storage through Amazon CloudFront versus going back to Amazon S3 for each new `GET` on an object.

### TCP Window Scaling

Transmission Control Protocol (TCP) window scaling allows you to improve network throughput performance between your operating system, application layer, and Amazon S3 by supporting window sizes larger than 64 KB. Although it can improve performance,

it can be challenging to set up correctly, so refer to the AWS Documentation repository for details.

#### TCP Selective Acknowledgment

*TCP selective acknowledgment* is designed to improve recovery time after a large number of packet losses. It is supported by most newer operating systems, but it might have to be enabled. Refer to the Amazon S3 Developer Guide for more information.

### Pricing

With Amazon S3, you pay only for what you use. There is no minimum fee, and there is no charge for data transfer into Amazon S3.

You pay for the following:

- The storage that you use
- The API calls that you make (`PUT`, `COPY`, `POST`, `LIST`, `GET`)
- Data transfer out of Amazon S3

Data transfer out pricing is tiered, so the more you use, the lower your cost per gigabyte. Refer to the AWS website for the latest pricing.


Amazon S3 pricing differs from the pricing of Amazon EBS volumes in that if you create an Amazon EBS volume and store nothing on it, you are still paying for the storage space of the volume that you have allocated. With Amazon S3, you pay for the storage space that is being used—not allocated.

## Object Lifecycle Management

To manage your objects so that they are stored cost effectively throughout their lifecycle, use a *lifecycle configuration*. A lifecycle configuration is a set of rules that defines actions that Amazon S3 applies to a group of objects.

There are two types of actions:

**Transition actions** *Transition actions* define when objects transition to another storage class. For example, you might choose to transition objects to the STANDARD_IA storage class 30 days after you created them or archive objects to the GLACIER storage class one year after creating them.

**Expiration actions** *Expiration actions* define when objects expire. Amazon S3 deletes expired objects on your behalf.

## When Should You Use Lifecycle Configuration?

You should use lifecycle configuration rules for objects that have a well-defined lifecycle. The following are some examples:

- If you upload periodic logs to a bucket, your application might need them for a week or a month. After that, you may delete them.
- Some documents are frequently accessed for a limited period of time. After that, they are infrequently accessed. At some point, you might not need real-time access to them, but your organization or regulations might require you to archive them for a specific period. After that, you may delete them.
- You can upload some data to Amazon S3 primarily for archival purposes. For example, archiving digital media, financial, and healthcare records; raw genomics sequence data, long-term database backups; and data that must be retained for regulatory compliance.

With lifecycle configuration rules, you can tell Amazon S3 to transition objects to less expensive storage classes or archive or delete them.

## Configuring a Lifecycle

A *lifecycle configuration* (an XML file) comprises a set of rules with predefined actions that you need Amazon S3 to perform on objects during their lifetime. Amazon S3 provides a set of API operations for managing lifecycle configuration on a bucket, and it is stored by Amazon S3 as a *lifecycle subresource* that is attached to your bucket.

You can also configure the lifecycle by using the Amazon S3 console, the AWS SDKs, or the REST API.

The following lifecycle configuration specifies a rule that applies to objects with key name prefix `logs/`. The rule specifies the following actions:

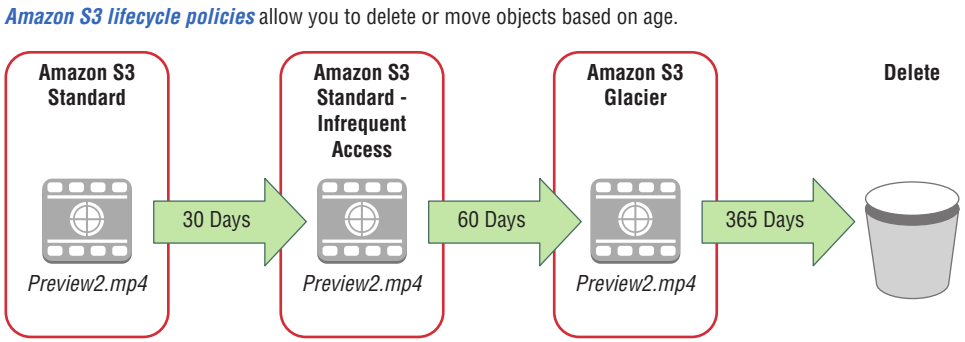
- Two transition actions
  - Transition objects to the STANDARD_IA storage class 30 days after creation
  - Transition objects to the GLACIER storage class 90 days after creation
- One expiration action that directs Amazon S3 to delete objects a year after creation

```
<LifecycleConfiguration>
  <Rule>
    <ID>example-id</ID>
    <Filter>
       <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
```

```
    <Transition>
      <Days>90</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Figure 3.18 shows a set of Amazon S3 lifecycle policies in place. These policies move files automatically from one storage class to another as they age out at certain points in time.

**FIGURE 3.18** Amazon S3 lifecycle policies




# AWS File Storage Services

AWS offers Amazon Elastic File System (Amazon EFS) for file storage to enable you to share access to files that reside on the cloud.

## Amazon Elastic File System

*Amazon Elastic File System* (Amazon EFS) provides scalable file storage and a standard file system interface for use with Amazon EC2. You can create an Amazon EFS file system, configure your instances to mount the file system, and then use an Amazon EFS file system as a common data source for workloads and application running on multiple instances.

Amazon EFS can be mounted to multiple Amazon EC2 instances simultaneously, where it can continue to expand up to petabytes while providing low latency and high throughput.

Consider using Amazon EFS instead of Amazon S3 or Amazon EBS if you have an application (Amazon EC2 or on premises) or a use case that requires a file system and any of the following:

- Multi-attach
- GB/s throughput
- Multi-AZ availability/durability
- Automatic scaling (growing/shrinking of storage)

Customers use Amazon EFS for the following use cases today:

- Web serving
- Database backups
- Container storage
- Home directories
- Content management
- Analytics
- Media and entertainment workflows
- Workflow management
- Shared state management


Amazon EFS is not supported on Windows instances.

## Creating your Amazon EFS File System

### File System

The *Amazon EFS file system* is the primary resource in Amazon EFS, and it is where you store your files and directories. You can create up to 125 file systems per account.

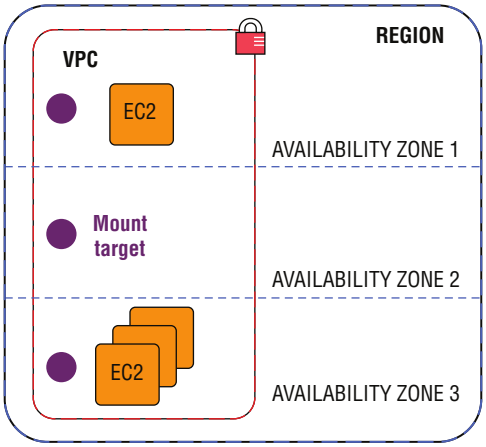
### Mount Target

To access your file system from within a VPC, create mount targets in the VPC. A *mount target* is a Network File System (NFS) endpoint within your VPC that includes an IP address and a DNS name, both of which you use in your mount command. A mount target is highly available, and it is illustrated in Figure 3.19.

## Accessing an Amazon EFS File System

There are several different ways that you can access an Amazon EFS file system, including using Amazon EC2 and AWS Direct Connect.

**FIGURE 3.19** Mount target




### Using Amazon Elastic Compute Cloud

To access a file system from an *Amazon Elastic Compute Cloud* (Amazon EC2) instance, you must mount the file system by using the standard Linux `mount` command, as shown in Figure 3.20. The file system will then appear as a local set of directories and files. An NFS v4.1 client is standard on Amazon Linux AMI distributions.

**FIGURE 3.20** Mounting the file system

```
mount -t nfs4 -o nfsvers=4.1
    [file system DNS name]:/
    /[user's target directory]
```

In your command, specify the file system type (nfs4), the version (4.1), the file system DNS name or IP address, and the user's target directory.

A file system belongs to a region, and your Amazon EFS file system spans all Availability Zones in that region. Once you have mounted your file system, data can be accessed from any Availability Zone in the region within your VPC while maintaining full consistency. Figure 3.21 shows how you communicate with Amazon EC2 instances within a VPC.

### Using AWS Direct Connect

You can also mount your on-premises servers to Amazon EFS in your Amazon VPC using AWS Direct Connect. With *AWS Direct Connect*, you can mount your on-premises servers to Amazon EFS using the same mount command used to mount in Amazon EC2. Figure 3.22 shows how to use AWS Direct Connect with Amazon EFS.

**FIGURE 3.21** Using Amazon EFS

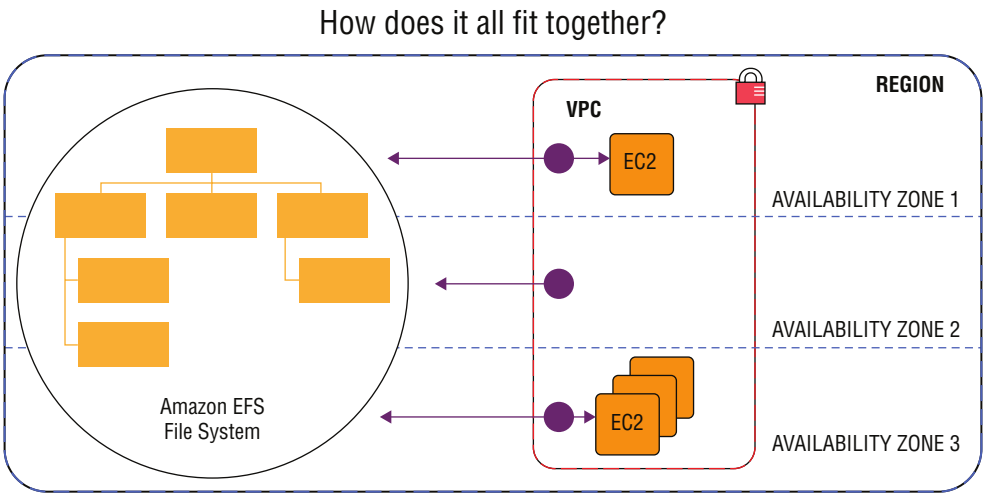
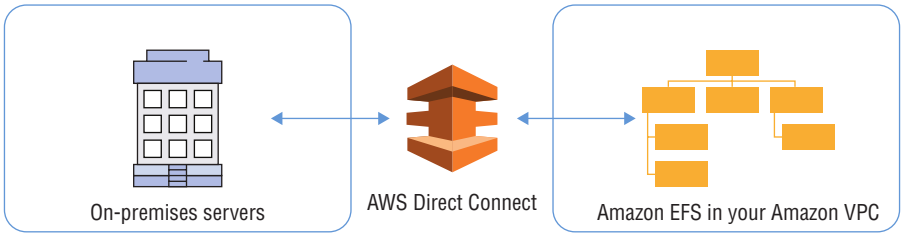



**FIGURE 3.22** Using AWS Direct Connect with Amazon EFS




Customers can use Amazon EFS combined with AWS Direct Connect for migration, bursting, or backup and disaster recovery.

## Syncing Files Using AWS DataSync

Now that you have a functioning Amazon EFS file system, you can use *AWS DataSync* to synchronize files from an existing file system to Amazon EFS. AWS DataSync can synchronize your file data and also file system metadata such as ownership, time stamps, and access permissions.

To do this, download and deploy a sync agent from the Amazon EFS console as either a *virtual machine* (VM) image or an AMI.

Next, create a sync tack and configure your source and destination file systems. Then start your task to begin syncing the files and monitor the progress of the file sync using Amazon CloudWatch.

## Performance

Amazon EFS is designed for a wide spectrum of performance needs, including the following:

- High throughput and parallel I/O
- Low latency and serial I/O

To support those two sets of workloads, Amazon EFS offers two different performance modes, as described here:

**General purpose (default)** General-purpose mode is the default mode, and it is used for latency-sensitive applications and general-purpose workloads, offering the lowest latencies for file operations. While there is a trade-off of limiting operations to 7,000 per second, general-purpose mode is the best choice for most workloads.

**Max I/O** If you are running large-scale and data-heavy applications, then choose the max I/O performance option, which provides you with a virtually unlimited ability to scale out throughput and IOPS, but with a trade-off of slightly higher latencies. Use max I/O when you have 10 or more instances accessing your file system concurrently, as shown in Table 3.6.

**TABLE 3.6** I/O Performance Options

| Mode | What's It For? | Advantages | Trade-Offs | When to Use |
|---|---|---|---|---|
| General purpose (default) | Latency-sensitive applications and general-purpose workloads | Lowest latencies for file operations | Limit of 7,000 ops/sec | Best choice for most workloads |
| Max I/O | Large-scale and data-heavy applications | Virtually unlimited ability to scale out throughput/ IOPS | Slightly higher latencies | Consider if 10 (or more) instances are accessing your file system concurrently |

If you are not sure which mode is best for your usage pattern, use the `PercentIOLimit` Amazon CloudWatch metric to determine whether you are constrained by general-purpose mode. If you are regularly hitting the 7,000 IOPS limit in general-purpose mode, then you will likely benefit from max I/O performance mode.

As discussed with the CAP theorem earlier in this study guide, there are differences in both performance and trade-off decisions when you're designing systems that use Amazon EFS and Amazon EBS. The distributed architecture of Amazon EFS results in a small increase in latency for each operation, as the data that you are storing gets pushed across multiple servers in multiple Availability Zones. Amazon EBS can provide lower latency

than Amazon EFS, but at the cost of some durability. With Amazon EBS, you provision the size of the device, and if you reach its maximum limit, you must increase its size or add more volumes, whereas Amazon EFS scales automatically. Table 3.7 shows the various performance and other characteristics for Amazon EFS as related to Amazon EBS Provisioned IOPS.

**TABLE 3.7** Amazon EBS Performance Relative to Amazon EFS

| | | Amazon EFS | Amazon EBS Provisioned IOPS |
|---|---|---|---|
| **Performance** | Per-operation latency | Low, consistent | Lowest, consistent |
| | Throughput scale | Multiple GBs per second | Single GB per second |
| **Characteristics** | Data availability/ durability | Stored redundantly across multiple Availability Zones | Stored redundantly in a single Availability Zone |
| | Access | 1 to 1000s of EC2 instances, from multiple Availability Zones, concurrently | Single Amazon EC2 instance in a single Availability Zone |
| | Use cases | Big Data and analytics, media processing workflows, content management, web serving, home directories | Boot volumes, transactional and NoSQL databases, data warehousing, ETL |

## Security

You can implement security in multiple layers with Amazon EFS by controlling the following:

- Network traffic to and from file systems (mount targets) using the following:
  - VPC security groups
  - Network ACLs
- File and directory access by using POSIX permissions
- Administrative access (API access) to file systems by using IAM. Amazon EFS supports:
  - Action-level permissions
  - Resource-level permissions

Familiarize yourself with the Amazon EFS product, details, and FAQ pages. Some exam questions may be answered by components from those pages.

# Storage Comparisons

This section provides valuable charts that can serve as a quick reference if you are tasked with choosing a storage system for a particular project or application.

## Use Case Comparison

Table 3.8 will help you understand the main properties and use cases for each of the cloud storage products on AWS.

**TABLE 3.8** AWS Cloud Storage Products

| If You Need: | Consider Using: |
|---|---|
| Persistent local storage for Amazon EC2, relational and NoSQL databases, data warehousing, enterprise applications, big data processing, or backup and recovery | Amazon EBS |
| A file system interface and file system access semantics to make data available to one or more Amazon EC2 instances for content serving, enterprise applications, media processing workflows, big data storage, or backup and recovery | Amazon EFS |
| A scalable, durable platform to make data accessible from any internet location for user-generated content, active archive, serverless computing, Big Data storage, or backup and recovery | Amazon S3 |
| Highly affordable, long-term storage that can replace tape for archive and regulatory compliance | Amazon S3 Glacier |
| A hybrid storage cloud augmenting your on-premises environment with AWS cloud storage for bursting, tiering, or migration | AWS Storage Gateway |
| A portfolio of services to help simplify and accelerate moving data of all types and sizes into and out of the AWS Cloud | AWS Cloud Data Migration Services |

## Storage Temperature Comparison

Table 3.9 shows a comparison of instance store, Amazon EBS, Amazon S3, and Amazon S3 Glacier.

Understanding Table 3.9 will help you make decisions about latency, size, durability, and cost during the exam.

**TABLE 3.9** Storage Comparison

| | Instance Store | Amazon EBS | Amazon S3 | Amazon S3 Glacier |
|---|---|---|---|---|
| **Average latency** | ms | | ms, sec, min (~ size) | hrs |
| **Data volume** | 4 GB to 48 TB | 1 GiB to 1 TiB | No limit | |
| **Item size** | Block storage | | 5 TB max | 40 TB max |
| **Request rate** | Very high | | Low to very high (no limit) | Very low (no limit) |
| **Cost/GB per month** | Amazon EC2 instance cost | ¢¢ | ¢ | |
| **Durability** | Low | High | Very high | Very high |
| **Temperature** | Hot <————————————————————> Cold | | | |

## Comparison of Amazon EBS and Instance Store

Before considering Amazon EC2 instance store as a storage option, make sure that your data does *not* meet any of these criteria:

- Must persist through instance stops, terminations, or hardware failures
- Needs to be encrypted at the full volume level
- Needs to be backed up with Amazon EBS snapshots
- Needs to be removed from instances and reattached to another

If your data meets any of the previous four criteria, use an Amazon EBS volume. Otherwise, compare instance store and Amazon EBS for storage.

Because instance store is directly attached to the host computer, it will have lower latency than an Amazon EBS volume attached to the Amazon EC2 instance. Instance store is provided at no additional cost beyond the price of the Amazon EC2 instance you choose (if the instance has instance store[s] available), whereas Amazon EBS volumes incur an additional cost.

## Comparison of Amazon S3, Amazon EBS, and Amazon EFS

Table 3.10 is a useful in helping you to compare performance and storage characteristics for Amazon's highest-performing file, object, and block cloud storage offerings. This comparison will also be helpful when choosing the right data store for the applications that you are developing. It is also important for the exam.

**TABLE 3.10** Storage Service Comparison (EFS, S3, and EBS)

| | | File<br>Amazon EFS | Object<br>Amazon S3 | Block<br>Amazon EBS |
|---|---|---|---|---|
| **Performance** | Per-operation latency | Low, consistent | Low, for mixed request types, and integration with CloudFront | Low, consistent |
| | Throughput scale | Multiple GB per second | | Single GB per second |
| **Characteristics** | Data Availability/ Durability | Stored redundantly across multiple Availability Zones | | Stored redundantly in a single Availability Zone |
| | Access | One to thousands of Amazon EC2 instances or on-premises servers, from multiple Availability Zones, concurrently | One to millions of connections over the web | Single Amazon EC2 instance in a single Availability Zone |
| | Use Cases | Web serving and content management, enterprise applications, media and entertainment, home directories, database backups, developer tools, container storage, Big Data analytics | Web serving and content management, media and entertainment, backups, Big Data analytics, data lake | Boot volumes, transactional and NoSQL databases, data warehousing, ETL |

run. The remaining sections allow for fine-grained configurations to integrate packages, sources, files, and container commands.

Launch environments from integrated development environment (IDE) tools to avoid poorly formatted configurations and source bundles that could cause unrecoverable failures.

You apply configuration files in the `ebextensions` directory to Elastic Beanstalk stacks. The stacks are the AWS resources that you allocate for your infrastructure and application. If you have any resource, such as Amazon VPC, Amazon EC2, or Amazon S3, that was updated or configured, these files deploy with your changes. You can zip your `ebextension` files, upload, and apply them to multiple application environments. You can view your environment variables in `option_settings` for future evaluation or changes. These are accessible from the AWS Management Console, command line, and API calls.

You can view Elastic Beanstalk stacks in AWS CloudFormation, but always use the Elastic Beanstalk service and `ebextensions` to make modifications. This way, edits and modifications to the application stacks are simplified without introducing unrecoverable failures.

Elastic Beanstalk generates logs that you can view to troubleshoot your environments and resources. The logs display Amazon EC2 operational logs and logs that are specific to servers running for your applications.

## Integrating with Other AWS Services

Elastic Beanstalk automatically integrates or manages other AWS services with application code to provision efficient working environments. However, you might find it necessary to add additional services, such as Amazon S3 for content storage or Amazon DynamoDB for data records, to work with an environment. To grant access between any integrated service and Elastic Beanstalk, you must configure permissions in IAM.

### Amazon S3

You can use Amazon S3 to store static content you want to integrate with your application and point directly to objects you store in Amazon S3 from your application or from other resources. In addition to setting permissions in IAM policies, take advantage of presigned URLs for controlled Amazon S3 `GET` and `PUT` operations.

### Amazon CloudFront

You can integrate your Elastic Beanstalk environment with Amazon CloudFront, which provides content delivery and distribution through the use of edge locations throughout the world. This can decrease the time in which your content is delivered to you, as the content

is cached and routed through the closest edge location serving you. After you deploy your application on Elastic Beanstalk, use the Amazon CloudFront content delivery network (CDN) to cache static content from your application. To identify the source of your content in Amazon CloudFront, you can use URL path patterns to cache your content and then retrieve it from the cache. This approach serves your content more rapidly and offloads requests directly sourced from your application.

## AWS Config

With *AWS Config*, you can visualize configuration history and how configurations evolve over time. Tracking changes helps you to fulfill compliance obligations and meet auditing requirements. You can integrate AWS Config directly with your application and its versions or your Elastic Beanstalk environment. *You can customize AWS Config to record changes per resource, per region, or globally.* In the AWS Config console, you can select Elastic Beanstalk resource types to record specific applications and environment resources. You can view the recorded information in the AWS Config dashboard under Resource Inventory.

## Amazon RDS

Various options are available for creating databases for your environment, such as Amazon Relational Database Service (Amazon RDS) for SQL databases and Amazon DynamoDB for NoSQL databases. Elastic Beanstalk can create a database and store and retrieve data for any of your environments. Each service has its own features to handle scaling, capacity, performance, and availability.

To store, read, or write to your data records, you can set up an Amazon RDS database instance or an Amazon DynamoDB table by using the same configuration files for your other service option settings. You must create connections to the database, which require you to set up password management in Elastic Beanstalk. Your configurations are saved in the `ebextensions` directory. You can also create direct connections, within your application code or application configuration files, to both internal and external databases. When using Amazon RDS, avoid accidentally deleting and re-creating databases without a properly installed backup. To reduce the risk of losing data, take a manual snapshot of the master Amazon RDS database immediately before deleting.


If you create periodic tasks with a worker environment, Elastic Beanstalk automatically creates an Amazon DynamoDB table to perform leader election and stores task information.

## Amazon ElastiCache

For caching capabilities, you can integrate Amazon ElastiCache service clusters with the Elastic Beanstalk environment. If you use a nonlegacy container, you can set your configuration files to use the supported container and then offload requests to the cache cluster.

Route 53 to configure DNS health checks to route traffic to healthy endpoints or to monitor independently the health of your application and its endpoints. Amazon Route 53 Traffic Flow makes it easy for you to manage traffic globally through a variety of routing types, including latency-based routing, geolocation, geoproximity, and weighted round-robin, all of which can be combined with DNS failover to enable a variety of low-latency, fault-tolerant architectures.

Using Amazon Route 53 Traffic Flow's simple visual editor, you can easily manage how your end users are routed to your application's endpoints—whether in a single AWS Region or distributed around the globe. Amazon Route 53 also offers domain name registration. You can purchase and manage domain names such as `example.com`, and Amazon Route 53 will automatically configure DNS settings for your domains.

## Speeding Up Content Delivery with Amazon CloudFront

*Latency* is an increasingly important aspect when you deliver web applications to the end user, as you always want your end user to have an efficient, low-latency experience on your website. Increased latency can result in both decreased customer satisfaction and decreased sales. One way to decrease latency is to use *Amazon CloudFront* to move your content closer to your end users. Amazon CloudFront has two delivery methods to deliver content. The first is a web distribution, and this is for storing of `.html`, `.css`, and graphic files. Amazon CloudFront also provides the ability to have an RTMP distribution, which speeds up distribution of your streaming media files using Adoble Flash Media Server's RTMP protocol. An RTMP distribution allows an end user to begin playing a media file before the file has finished downloading from a CloudFront edge location.

To use Amazon CloudFront with your Amazon S3 static website, perform these tasks:

1. Choose a delivery method.

   In the example, Amazon S3 is used to store a static web page; thus, you will be using the Web delivery method. However, as mentioned previously, you could also use RTMP for streaming media files.

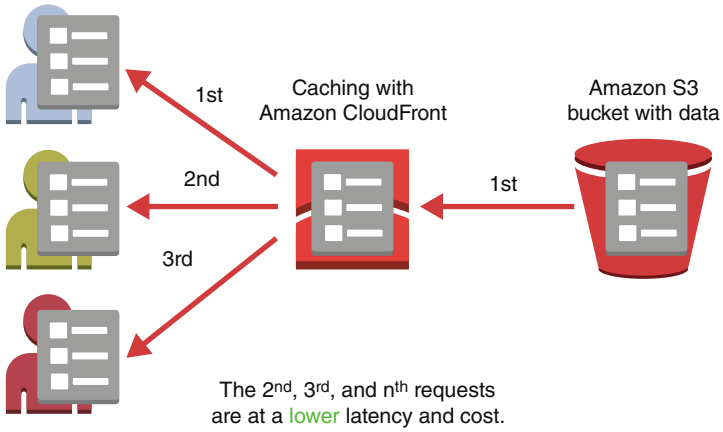
2. Specify the cache behavior. A cache behavior lets you configure a variety of CloudFront functionality for a given URL path pattern for files on your website.

3. Choose the distribution settings and network that you want to use. For example, you can use all edge locations or only U.S., Canada, and Europe locations.

Amazon CloudFront enables you to cache your data to minimize redundant data-retrieval operations. Amazon CloudFront reduces the number of requests to which your origin server must respond directly. This reduces the load on your origin server and reduces latency because more objects are served from Amazon CloudFront edge locations, which are closer to your users.

The Amazon S3 bucket pushes the first request to Amazon CloudFront's cache. The second, third, and $n^{th}$ requests pull from the Amazon CloudFront's cache at a lower latency and cost, as shown in Figure 13.1.

**FIGURE 13.1** Amazon CloudFront cache




The more requests that Amazon CloudFront is able to serve from edge caches as a proportion of all requests (that is, the greater the cache hit ratio), the fewer viewer requests that Amazon CloudFront needs to forward to your origin to get the latest version or a unique version of an object. You can view the percentage of viewer requests that are hits, misses, and errors in the Amazon CloudFront console.

A number of factors affect the cache hit ratio. You can adjust your Amazon CloudFront distribution configuration to improve the cache hit ratio.

Use Amazon CloudFront with Amazon S3 to improve your performance, decrease your application's latency and costs, and provide a better user experience. Amazon CloudFront is also a serverless service, and it fits well with serverless stack services, especially when you use it in conjunction with Amazon S3.

# Dynamic Data with Amazon API Gateway (Logic or App Tier)

This section details how to use dynamic data with the Amazon API Gateway service in a logic tier or app tier.

*Amazon API Gateway* is a fully managed, serverless AWS service, with no server that runs inside your environment to define, deploy, monitor, maintain, and secure APIs at any scale. Clients integrate with the APIs that use standard HTTPS requests. Amazon API Gateway can integrate with a service-oriented multitier architecture with Amazon services,

such as AWS Lambda and Amazon EC2. It also has specific features and qualities that make it a powerful edge for your logic tier. You can use these features and qualities to enhance and build your dynamic web application.

The Amazon API Gateway integration strategy that provides access to your code includes the following:

**Control service** Uses REST to provide access to Amazon services, such as AWS Lambda, Amazon Kinesis, Amazon S3, and Amazon DynamoDB. The access methods include the following:

- Consoles
- CLI
- SDKs
- REST API requests and responses

**Execution service** Uses standard HTTP protocols or language-specific SDKs to deploy API access to backend functionality.


Do not directly expose resources or the API—always use AWS edge services and the Amazon API Gateway service to safeguard your resources and APIs.

## Endpoints

There are three types of *endpoints* for Amazon API Gateway.

**Regional endpoints** Live inside the AWS Region, such as `us-west-2`.

**Edge optimized endpoints** Use Amazon CloudFront, a content delivery web service with the AWS global network of edge locations as connection points for clients, and integrate with your API.

**Private endpoints** Can live only inside of a *virtual private cloud* (VPC).

You use Amazon API Gateway to help drive down the total response time latency of your API. You can improve the performance of specific API requests with Amazon API Gateway to store responses in an optional in-memory cache. This not only provides performance benefits for API requests that repeat, but it also reduces backend executions, which helps to reduce overall costs.

The API endpoint can be a default host name or a custom domain name. The default host name is as follows:

```
{api-id}.execute-api.{region}.amazonaws.com
```

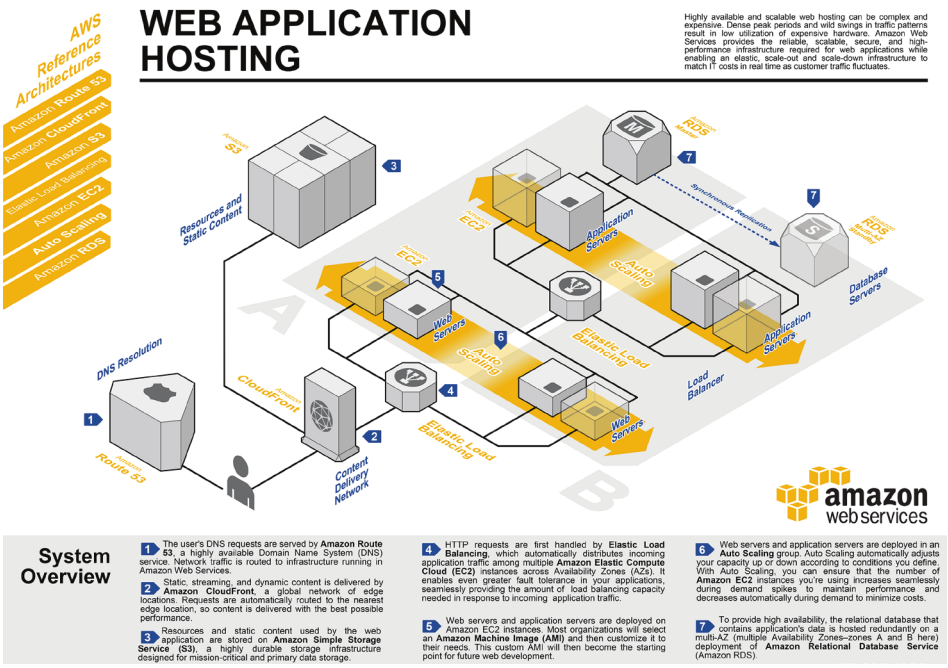
In addition to using the higher-level mobile and JavaScript SDKs, you can also use the lower-level APIs available via the following AWS SDKs to integrate all Amazon Cognito functionality in your applications:

- Java SDK
- .NET SDK
- Node.js SDK
- Python SDK
- PHP SDK
- Ruby SDK

# Standard Three-Tier vs. the Serverless Stack

This chapter has introduced serverless services and their benefits. Now that you know about some of the serverless services that are available in AWS, let's compare a traditional three-tier application against a serverless application architecture. Figure 13.5 shows a typical three-tier web application.

**FIGURE 13.5** Standard three-tier web infrastructure architecture




*Source*: `https://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_web_01.pdf`

This architecture uses the following components and services:

**Routing:** Amazon Route 53

**Content distribution network (CDN):** Amazon CloudFront

**Static data:** Amazon S3

**High availability/decoupling:** Application load balancers

**Web servers:** Amazon EC2 with Auto Scaling

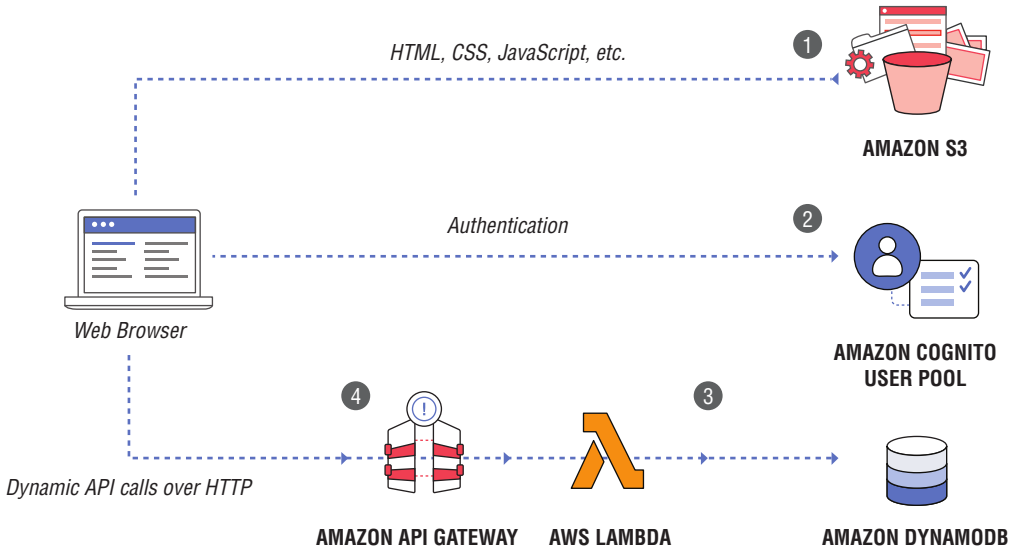
**App servers:** Amazon EC2 with Auto Scaling

**Database:** Amazon RDS in a multi-AZ configuration

Amazon Route 53 provides a DNS service that allows you to take domain names such as `examplecompany.com` and translate them to an IP address that points to running servers.

The CDN shown in Figure 13.6 is the Amazon CloudFront service, which improves your site performance with the use of its global content delivery network.

**FIGURE 13.6** Serverless web application architecture




*Source*: `https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/`

Amazon S3 stores your static files such as photos or movie files.

*Application load balancers* are responsible for distributing load across Availability Zones to your Amazon EC2 servers, which run your web application with a service such as Apache or NGINX.

Application servers are responsible for performing business logic prior to storing the data in your database servers that are run by Amazon RDS.

Amazon RDS is the managed database server, and it can run an Amazon Aurora, Microsoft SQL Server, Oracle SQL Server, MySQL, PostgreSQL, or MariaDB database server.

While this architecture is a robust and highly available service, there are several downsides, including the fact that you have to manage servers. You are responsible for patching those servers, preventing downtime associated with those patches, and proper server scaling.

In a typical serverless web application architecture, you also run a web application, but you have zero servers that run inside your AWS account, as shown in Figure 13.6.

Serverless web application architecture services include the following:

**Routing:** Amazon Route 53

**Web servers/static data:** Amazon S3

**User authentication:** Amazon Cognito user pools

**App servers:** Amazon API Gateway and AWS Lambda

**Database:** Amazon DynamoDB

Amazon Route 53 is your DNS, and you can use Amazon CloudFront for your CDN.

You can also use Amazon S3 for your web servers. In this architecture, you use Amazon S3 to host your entire static website. You use JavaScript to make API calls to the Amazon API Gateway service.

For your business or application servers, you use Amazon API Gateway in conjunction with AWS Lambda. This allows you to retrieve and save data dynamically.

You use Amazon DynamoDB as a serverless database service, and you do not provision any Amazon EC2s inside of your Amazon VPC account. Amazon DynamoDB is also a great database service for storing session state for stateful applications. You can use Amazon RDS instead if you need a relational database. However, it would not then be a fully serverless stack. There is a new service released called *Amazon Aurora Serverless*, which is a full RDS MySQL 5.6–compatible service that is completely serverless. This would allow you to run a traditional SQL database, but one that has the benefit of being serverless. Amazon Aurora Serverless is discussed in the next section.

You use Amazon Cognito user pools for user authentication, which provides a secure user directory that can scale to hundreds of millions of users. Amazon Cognito User Pools is a fully managed service with no servers for you to manage. While user authentication was not shown in Figure 13.6, you can use your web server tier to talk to a user directory, such as *Lightweight Directory Access Protocol* (LDAP), for user authentication.

As you can see, while some of the components are the same, you may use them in slightly different ways. By taking advantage of the AWS global network, you can develop fully scalable, highly available web applications—all without having to worry about maintaining or patching servers.

# Amazon Aurora Serverless

Amazon Aurora Serverless is an on-demand, auto-scaling configuration for the Aurora MySQL-compatible edition, where the database automatically starts, shuts down, and scales up or down as needed by your application. This allows you to run a traditional SQL database in the cloud without needing to manage any infrastructure or instances.

With Amazon Aurora Serverless, you also get the same high availability as traditional Amazon Aurora, which means that you get six-way replication across three Availability Zones inside of a region in order to prevent against data loss.

Amazon Aurora Serverless is great for infrequently used applications, new applications, variable workloads, unpredictable workloads, development and test databases, and multitenant applications. This is because you can scale automatically when you need to and scale down when application demand is not high. This can help cut costs and save you the heartache of managing your own database infrastructure.

Amazon Aurora Serverless is easy to set up, either through the console or directly with the CLI. To create an Amazon Aurora Serverless cluster with the CLI, you can run the following command:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster --engine aurora
--engine-version 5.6.10a \
--engine-mode serverless --scaling-configuration
MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true \
--master-username user-name --master-user-password password \
--db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
–region us-east-1
```

Amazon Aurora Serverless gives you many of the similar benefits as other serverless technologies, such as AWS Lambda, but from a database perspective. Managing databases is hard work, and with Amazon Aurora Serverless, you can utilize a database that automatically scales and you don't have to manage any of the underlying infrastructure.

# AWS Serverless Application Model

The *AWS Serverless Application Model* (AWS SAM) allows you to create and manage resources in your serverless application with *AWS CloudFormation* to define your serverless application infrastructure as a SAM template. A *SAM template* is a JSON or YAML configuration file that describes the AWS Lambda functions, API endpoints, tables, and other resources in your application. With simple commands, you upload this template to AWS CloudFormation, which creates the individual resources and groups them into an *AWS CloudFormation stack* for ease of management. When you update your AWS SAM template, you re-deploy the changes to this stack. AWS CloudFormation updates the individual resources for you.

AWS SAM is an extension of AWS CloudFormation. You can define resources by using the AWS CloudFormation in your AWS SAM template. This is a powerful feature, as you can use AWS SAM to create a template of your serverless infrastructure, which you can then build into a DevOps pipeline. For example, examine the following:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Description: 'Example of Multiple-Origin CORS using API Gateway and Lambda'
```

```
Resources:
  ExampleRoot:
    Type: 'AWS::Serverless::Function'
    Properties:
      CodeUri: '.'
      Handler: 'routes/root.handler'
      Runtime: 'nodejs8.10'
      Events:
        Get:
          Type: 'Api'
          Properties:
            Path: '/'
            Method: 'get'
  ExampleTest:
    Type: 'AWS::Serverless::Function'
    Properties:
      CodeUri: '.'
      Handler: 'routes/test.handler'
      Runtime: 'nodejs8.10'
      Events:
        Delete:
          Type: 'Api'
          Properties:
            Path: '/test'
            Method: 'delete'
        Options:
          Type: 'Api'
          Properties:
            Path: '/test'
            Method: 'options'

Outputs:
    ExampleApi:
      Description: "API Gateway endpoint URL for Prod stage for API Gateway
Multi-Origin CORS Function"
      Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}
.amazonaws.com/Prod/"
    ExampleRoot:
      Description: "API Gateway Multi-Origin CORS Lambda Function (Root) ARN"
      Value: !GetAtt ExampleRoot.Arn
    ExampleRootIamRole:
```

```
      Description: "Implicit IAM Role created for API Gateway Multi-Origin CORS
Function (Root)"
      Value: !GetAtt ExampleRootRole.Arn
    ExampleTest:
      Description: "API Gateway Multi-Origin CORS Lambda Function (Test) ARN"
      Value: !GetAtt ExampleTest.Arn
    ExampleTestIamRole:
      Description: "Implicit IAM Role created for API Gateway Multi-Origin CORS
Function (Test)"
      Value: !GetAtt ExampleTestRole.Arn
```

In the previous code example, you create two AWS Lambda functions and then associate *three* different Amazon API Gateway endpoints to trigger those functions. To deploy this AWS SAM template, download the template and all of the necessary dependencies from here:

```
https://github.com/awslabs/serverless-application-model/tree/develop/
examples/apps/api-gateway-multiple-origin-cors
```

AWS SAM is similar to AWS CloudFormation, with a few key differences, as shown in the second line:

```
Transform: 'AWS::Serverless-2016-10-31'
```

This important line of code transforms the AWS SAM template into an AWS CloudFormation template. Without it, the AWS SAM template will not work.

Similar to the AWS CloudFormation, you also have a `Resources` property where you define infrastructure to provision. The difference is that you provision serverless services with a new `Type` called `AWS::Serverless::Function`. This provisions an AWS Lambda function to define all properties from an AWS Lambda point of view. AWS Lambda includes `Properties`, such as `MemorySize`, `Timeout`, `Role`, `Runtime`, `Handler`, and others.

While you can create an AWS Lambda function with AWS CloudFormation using `AWS::Lambda::Function`, the benefit of AWS SAM lies in a property called `Event`, where you can tie in a trigger to an AWS Lambda function, all from within the `AWS::Serverless::Function` resource. This `Event` property makes it simple to provision an AWS Lambda function and configure it with an Amazon API Gateway trigger. If you use AWS CloudFormation, you would have to declare an Amazon API Gateway separately with `AWS::ApiGateway::Resource`.

To summarize, AWS SAM allows you to provision serverless resources more rapidly with less code by extending AWS CloudFormation.

# AWS SAM CLI

Now that we've addressed AWS SAM, let's take a closer look at the AWS SAM CLI. With AWS SAM, you can define templates, in JSON or YAML, which are designed for provisioning serverless applications through AWS CloudFormation.

AWS SAM CLI is a command line interface tool that creates an environment in which you can develop, test, and analyze your serverless-based application, all locally. This allows you to test your AWS Lambda functions before uploading them to the AWS service. AWS SAM CLI also allows you to develop and test your code quickly, and this gives you the ability to test it locally, which allows you to develop it faster. Previously, you would have had to upload your code each time you wanted to test an AWS Lambda function. Now, with the AWS SAM CLI, you can develop faster and get your application out the door more quickly.

To use AWS SAM CLI, you must meet a few prerequisites. You must install Docker, have Python 2.7 or 3.6 installed, have pip installed, install the AWS CLI, and finally install the AWS SAM CLI. You can read more about how to install AWS SAM CLI at `https://github.com/awslabs/aws-sam-cli`.

With AWS SAM CLI, you must define three key things.

- You must have a valid AWS SAM template, which defines a serverless application.
- You must have the AWS Lambda function defined. This can be in any valid language that Lambda currently supports, such as Node.js, Java 8, Python, and so on.
- You must have an event source. An *event source* is simply an `event.json` file that contains all the data that the Lambda function expects to receive. Valid event sources are as follows:
  - Amazon Alexa
  - Amazon API Gateway
  - AWS Batch
  - AWS CloudFormation
  - Amazon CloudFront
  - AWS CodeCommit
  - AWS CodePipeline
  - Amazon Cognito
  - AWS Config
  - Amazon DynamoDB
  - Amazon Kinesis
  - Amazon Lex
  - Amazon Rekognition
  - Amazon Simple Storage Service (Amazon S3)
  - Amazon Simple Email Service (Amazon SES)
  - Amazon Simple Notification Service (Amazon SNS)
  - Amazon Simple Queue Service (Amazon SQS)
  - AWS Step Functions

To generate this JSON event source, you can simply run this command in the AWS SAM CLI:

```
sam local generate-event <service> <event>
```

AWS SAM CLI is a great tool that allows developers to iterate quickly on their serverless applications. You will learn how to create and test an AWS Lambda function locally in the "Exercises" section of this chapter.

# AWS Serverless Application Repository

The *AWS Serverless Application Repository* enables you to deploy code samples, components, and complete applications quickly for common use cases, such as web and mobile backends, event and data processing, logging, monitoring, Internet of Things (IoT), and more. Each application is packaged with an AWS SAM template that defines the AWS resources. Publicly shared applications also include a link to the application's source code. There is no additional charge to use the serverless application repository. You pay only for the AWS resources you use in the applications you deploy.

You can also use the serverless application repository to publish your own applications and share them within your team, across your organization, or with the community at large. This allows you to see what other people and organizations are developing.

# Serverless Application Use Cases

Case studies on running serverless applications are located at the following URLs:

The Coca-Cola Company:

```
https://aws.amazon.com/blogs/aws/things-go-better-with-step-functions/
```

FINRA:

```
https://aws.amazon.com/solutions/case-studies/finra-data-validation/
```

iRobot:

```
https://aws.amazon.com/solutions/case-studies/irobot/
```

Localytics:

```
https://aws.amazon.com/solutions/case-studies/localytics/
```

# Summary

This chapter covered the AWS serverless core services, how to store your static files inside of Amazon S3, how to use Amazon CloudFront in conjunction with Amazon S3, how to integrate your application with user authentication flows using Amazon Cognito, and how to deploy and scale your API quickly and automatically with Amazon API Gateway.

Serverless applications have three main benefits: no server management, flexible scaling, and automated high availability. Without server management, you no longer have to provision or maintain servers. With AWS Lambda, you upload your code, run it, and focus on your application updates. With flexible scaling, you no longer have to disable Amazon EC2 instances to scale them vertically, groups do not need to be auto-scaled, and you do not need to create Amazon CloudWatch alarms to add them to load balancers. With AWS Lambda, you adjust the units of consumption (memory and execution time), and AWS adjusts the rest of the instance appropriately. Finally, serverless applications have built-in availability and fault tolerance. When periods of low traffic occur, you do not spend money on Amazon EC2 instances that do not run at their full capacity.

You can use an Amazon S3 web server to create your presentation tier. Within an Amazon S3 bucket, you can store HTML, CSS, and JavaScript files. JavaScript can create HTTP requests. These HTTP requests are sent to a REST endpoint service called Amazon API Gateway, which allows the application to save and retrieve data dynamically by triggering a Lambda function.

After you create your Amazon S3 bucket, you configure it to use static website hosting in the AWS Management Console and enter an endpoint that reflects your AWS Region.

Amazon S3 allows you to configure web traffic logs to capture information, such as the number of visitors who access your website in the Amazon S3 bucket.

One way to decrease latency and improve your performance is to use Amazon CloudFront with Amazon S3 to move your content closer to your end users. Amazon CloudFront is a serverless service.

The Amazon API Gateway is a fully managed service designed to define, deploy, and maintain APIs. Clients integrate with the APIs using standard HTTPS requests. Amazon API Gateway can integrate with a service-oriented multitier architecture. The Amazon API Gateway provides dynamic data in the logic or app tier.

There are three types of endpoints for Amazon API Gateway: regional endpoints, edge-optimized endpoints, and private endpoints.

In the Amazon API Gateway service, you expose addressable resources as a tree of API Resources entities, with the root resource (/) at the top of the hierarchy. The root resource is relative to the API's base URL, which consists of the API endpoint and a stage name.

You use Amazon API Gateways to help drive down the total response-time latency of your API. Amazon API Gateway uses the HTTP protocol to process these HTTP methods and send/receive data to and from the backend. Serverless data is sent to AWS Lambda to process.

You can use Amazon Route 53 to create a more user-friendly domain name instead of using the default host name (Amazon S3 endpoint). To support two subdomains, you create two Amazon S3 buckets that match your domain name and subdomain.

A stage is a named reference to a deployment, which is a snapshot of the API. Use a stage to manage and optimize a particular deployment. You create stages for each of your environments such as DEV, TEST, and PROD, so you can develop and update your API and applications without affecting production. Use Amazon API Gateway to set up authorizers with Amazon Cognito user pools on an AWS Lambda function. This enables you to secure your APIs.

An Amazon Cognito user pool includes a prebuilt user interface (UI) that you can use inside your application to build a user authentication flow quickly. A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito. Users can also sign in through social identity providers such as Facebook or Amazon and through Security Assertion Markup Language (SAML) identity providers.

Amazon Cognito identity pools allow you to create unique identities and assign permissions for your users to help you integrate with authentication providers. With the combination of user pools and identity pools, you can create a serverless user authentication system.

You can choose how users sign in with a username, an email address, and/or a phone number and to select attributes. Attributes are properties that you want to store about your end users. You can also configure password policies. Multi-factor authentication (MFA) prevents anyone from signing in to a system without authenticating through two different sources, such as a password and a mobile device–generated token. You create an Amazon Cognito role to send Short Message Service (SMS) messages to users.

The AWS Serverless Application Model (AWS SAM) allows you to create and manage resources in your serverless application with AWS CloudFormation as a SAM template. A SAM template is a JSON or YAML file that describes the AWS Lambda function, API endpoints, and other resources. You upload the template to AWS CloudFormation to create a stack. When you update your AWS SAM template, you redeploy the changes to this stack, and AWS CloudFormation updates the resources. You can use AWS SAM to create a template of your serverless infrastructure, which you can then build into a DevOps pipeline.

The `Transform: 'AWS::Serverless-2016-10-31'` code converts the AWS SAM template into an AWS CloudFormation template.

The AWS Serverless Application Repository enables you to deploy code samples, components, and complete applications for common use cases. Each application is packaged with an AWS SAM template that defines the AWS resources.

Additionally, you learned the differences between the standard three-tier web applications and the AWS serverless stack. You learned how to build your infrastructure quickly with AWS SAM and AWS SAM CLI for testing and development purposes.

# Exam Essentials

**Know serverless applications' three main benefits.** The benefits are as follows:

- No server management
- Flexible scaling
- Automated high availability

**Know what no server management means.** Without server management, you no longer have to provision or maintain servers. With AWS Lambda, you upload your code, run it, and focus on your application updates.

**Know what flexible scaling means.** With flexible scaling, you no longer have to disable Amazon Elastic Compute Cloud (Amazon EC2) instances to scale them vertically, groups do not need to be auto-scaled, and you do not need to create Amazon CloudWatch alarms to add them to load balancers. With AWS Lambda, you adjust the units of consumption (memory and execution time), and AWS adjusts the rest of the instances appropriately.

**Know what serverless applications mean.** Serverless applications have built-in availability and fault tolerance. You do not need to architect for these capabilities, as the services that run the application provide them by default. Additionally, when periods of low traffic occur on the web application, you do not spend money on Amazon EC2 instances that do not run at their full capacity.

**Know what services are serverless.** On the exam, it is important to understand which Amazon services are serverless and which ones are not. The following services are serverless:

- Amazon API Gateway
- AWS Lambda
- Amazon SQS
- Amazon SNS
- Amazon Kinesis
- Amazon Cognito
- Amazon Aurora Serverless
- Amazon S3

**Know how to host a serverless web application.** Hosting a serverless application means that you need Amazon S3 to host your static website, which comprises your HTML, JavaScript, and CSS files. For your database infrastructure, you can use Amazon DynamoDB or Amazon Aurora Serverless. For your business logic tier, you can use AWS Lambda. For DNS services, you can utilize Amazon Route 53. If you need the ability to host an API, you can use Amazon API Gateway. Finally, if you need to decrease latency to portions of your application, you can utilize services like Amazon CloudFront, which allows you to host your content at the edge.

# Resources to Review

Serverless Computing and Applications:

`https://aws.amazon.com/serverless/`

### Deleting Objects

Amazon S3 provides several options for deleting objects from your bucket. You can delete one or more objects directly from your S3 bucket. If you want to delete a single object, you can use the Amazon S3 `Delete` API. If you want to delete multiple objects, you can use the Amazon S3 Multi-Object Delete API, which enables you to delete up to 1,000 objects with a single request. The Multi-Object Delete operation requires a single query string `Delete` parameter to distinguish it from other bucket `POST` operations.

When deleting objects from a bucket that is not version-enabled, provide only the object key name. However, when deleting objects from a version-enabled bucket, provide the version ID of the object to delete a specific version of the object.

### Deleting Objects from a Version-Enabled Bucket

If your bucket is version-enabled, then multiple versions of the same object can exist in the bucket. When working with version-enabled buckets, the `DELETE` API enables the following options:

**Specify a nonversioned delete request** Specify only the object's key, not the version ID. In this case, Amazon S3 creates a delete marker and returns its version ID in the response. This makes your object disappear from the bucket.

**Specify a versioned delete request** Specify both the key and a version ID. In this case, the following two outcomes are possible:

- If the version ID maps to a specific object version, then Amazon S3 deletes the specific version of the object.
- If the version ID maps to the delete marker of that object, Amazon S3 deletes the delete marker. This causes the object to reappear in your bucket.

## Performance Optimization

This section discusses Amazon S3 best practices for optimizing performance.

### Request Rate and Performance Considerations

Amazon S3 scales to support high request rates. If your request rate grows steadily, Amazon S3 automatically partitions your buckets as needed to support higher request rates. However, if you expect a rapid increase in the request rate for a bucket to more than 300 `PUT/LIST/DELETE` requests per second or more than 800 `GET` requests per second, AWS recommends that you open a support case to prepare for the workload and avoid any temporary limits on your request rate.

If your requests are typically a mix of `GET`, `PUT`, `DELETE`, or `GET` Bucket (List Objects), choose the appropriate key names for your objects to ensure better performance by providing low-latency access to the Amazon S3 index. It also ensures scalability regardless of the number of requests that you send per second. If the bulk of your workload consists of `GET` requests, AWS recommends using the Amazon CloudFront content delivery service.

The Amazon S3 best practice guidelines apply only if you are routinely processing 100 or more requests per second. If your typical workload involves only occasional bursts of 100 requests per second and fewer than 800 requests per second, you do not need to follow these recommendations.

## Workloads with Different Request Types

When you are uploading a large number of objects, you might use sequential numbers or date-and-time values as part of the key names. For instance, you might decide to use key names that include a combination of the date and time, as shown in the following example, where the prefix includes a timestamp:

```
demobucket/2018-31-05-16-00-00/order1234234/receipt1.jpg
demobucket/2018-31-05-16-00-00/order3857422/receipt2.jpg
demobucket/2018-31-05-16-00-00/order1248473/receipt2.jpg
demobucket/2018-31-05-16-00-00/order8474937/receipt2.jpg
demobucket/2018-31-05-16-00-00/order1248473/receipt3.jpg
…
demobucket/2018-31-05-16-00-00/order1248473/receipt4.jpg
demobucket/2018-31-05-16-00-00/order1248473/receipt5.jpg
demobucket/2018-31-05-16-00-00/order1248473/receipt6.jpg
demobucket/2018-31-05-16-00-00/order1248473/receipt7.jpg
```

The way these keys are named presents a performance problem. To get a better understanding of this issue, consider the way that Amazon S3 stores key names. Amazon S3 maintains an index of object key names in each AWS Region. These keys are stored in UTF-8 binary ordering across multiple partitions in the index. The key name dictates where (or which partition) the key is stored in. In this case, where you use a sequential prefix such as a timestamp, or an alphabetical sequence, would increase the chances that Amazon S3 targets a specific partition for a large number of your keys, overwhelming the I/O capacity of the partition. However, if you introduce randomness in your key name prefixes, the key names, and therefore the I/O load, distribute across more than one partition.

If you anticipate that your workload will consistently exceed 100 requests per second, avoid sequential key names. If you must use sequential numbers or date-and-time patterns in key names, add a random prefix to the key name. The randomness of the prefix more evenly distributes key names across multiple index partitions.

The guidelines for the key name prefixes also apply to the bucket names. When Amazon S3 stores a key name in the index, it stores the bucket names as part of the key name (`demobucket/object.jpg`).

One way to introduce randomness to key names is to add a hash string as a prefix to the key name. For instance, you can compute an MD5 hash of the character sequence that you plan to assign as the key name. From the hash, select a specific number of characters and add them as the prefix to the key name.

A hashed prefix of three or four characters should be sufficient. AWS strongly recommends using a hexadecimal hash as the prefix.

### *GET*-Intensive Workloads

If your workload consists mostly of sending `GET` requests, then in addition to the preceding guidelines, consider using Amazon CloudFront for performance optimization.

Integrating CloudFront with Amazon S3 enables you to deliver content with low latency and a high data transfer rate. You will also send fewer direct requests to Amazon S3, which helps to lower your costs.

Suppose that you have a few objects that are popular. CloudFront fetches those objects from Amazon S3 and caches them. CloudFront can then serve future requests for the objects from its cache, reducing the number of `GET` requests it sends to Amazon S3.

## Storing Large Attribute Values in Amazon S3

Amazon DynamoDB currently limits the size of each item that you store in a table to 400 KB. If your application needs to store more data in an item than the DynamoDB size limit permits, store the large attributes as an object in Amazon S3. You can then store the Amazon S3 object identifier in your item.

You can also use the object metadata support in Amazon S3 to store the primary key value of the corresponding table item as Amazon S3 object metadata. Doing this provides a link back to the parent item in DynamoDB, which helps with maintenance of the Amazon S3 objects.

**Example 11: Store Large Attribute Values in Amazon S3**

Suppose that you have a table that stores product data, such as item price, description, book authors, and dimensions for other products. If you want to store an *image* of each product that was too large to fit in an *item*, use Amazon S3 images as an item in DynamoDB.

When implementing this strategy, remember the following limitations and restrictions:

- DynamoDB does not support transactions that cross Amazon S3 and DynamoDB. Therefore, your application must deal with any failures, which could include cleaning up orphaned Amazon S3 objects.
- Amazon S3 limits the length of object identifiers. You must organize your data in a way that does not generate excessively long object identifiers or violate other Amazon S3 constraints.

# Optimizing Data Transfer

Optimizing data transfer ensures that you minimize data transfer costs. Review your user presence if global or local and how the data gets located in order to reduce the latency issues.

- Use *Amazon CloudFront*, a global content delivery network (CDN), to locate data closer to users. It caches data at edge locations across the world, which reduces the load on your resources. By using CloudFront, you can reduce the administrative effort in delivering content automatically to large numbers of users globally, with minimum latency. Depending on your application types, distribute your entire website, including dynamic, static, streaming, and interactive content through CloudFront instead of scaling out your infrastructure.
- *Amazon S3 transfer acceleration* enables fast transfer of files over long distances between your client and your S3 bucket. Transfer acceleration leverages Amazon CloudFront globally distributed edge locations to route data over an optimized network path. For a workload in an S3 bucket that has intensive `GET` requests, you should use Amazon S3 with CloudFront.
- When uploading large files, use *multipart uploads* with multiple parts uploading at once to help maximize network throughput. Multipart uploads provide the following advantages:
  - *Improved throughput*—You can upload parts in parallel to improve throughput.
  - *Quick recovery from any network issues*—Smaller part size minimizes the impact of restarting a failed upload due to a network error.
  - *Pause and resume object uploads*—You can upload object parts over time. After you initiate a multipart upload, there is no expiry; you must explicitly complete or abort the multipart upload.
  - *Begin an upload before you know the final object size*—You can upload an object as you are creating it.
- Using *Amazon Route 53*, you can reduce latency for your users by serving their requests from the AWS Region for which network latency is lowest. Amazon Route 53 latency-based routing lets you use Domain Name System (DNS) to route user requests to the AWS Region that will give your users the fastest response.

## Caching

Caching improves application performance by storing frequently accessed data items in memory so that they can be retrieved without accessing the primary data store. Cached information might include the results of I/O-intensive database queries or the outcome of computationally intensive processing.

## AWS Trusted Advisor

*AWS Trusted Advisor* inspects your AWS environment and makes recommendations that help to improve the speed and responsiveness of your applications.

The following are a few Trusted Advisor checks to improve the performance of your service. The Trusted Advisor checks your service limits, ensuring that you take advantage of provisioned throughput, and monitors for overutilized instances:

- Amazon EC2 instances that are consistently at high utilization can indicate optimized, steady performance, but this check can also indicate that an application does not have enough resources.
- Provisioned IOPS (SSD) volumes that are attached to an Amazon EC2 instance that is not Amazon EBS–optimized. Amazon EBS volumes are designed to deliver the expected performance only when they are attached to an EBS-optimized instance.
- Amazon EC2 security groups with a large number of rules.
- Amazon EC2 instances that have a large number of security group rules.
- Amazon EBS magnetic volumes (standard) that are potentially overutilized and might benefit from a more efficient configuration.
- CloudFront distributions for alternate domain names with incorrectly configured DNS settings.

  Some HTTP request headers, such as Date or User-Agent, significantly reduce the cache hit ratio. This increases the load on your origin and reduces performance because CloudFront must forward more requests to your origin.

# Summary

In this chapter, you learned about the following:

- Cost-optimizing practices
- Right sizing your infrastructure
- Optimizing using Reserved Instances, Spot Instances, and AWS Auto Scaling
- Optimizing storage and data transfer
- Optimizing using NoSQL database (Amazon DynamoDB)
- Monitoring your costs and performance
- Tools, such as AWS Trusted Advisor, Amazon CloudWatch, and AWS Budgets

Achieving an optimized system is a continual process. An optimized system uses all the provisioned resources efficiently and achieves your business goal at the lowest price point. Engineers must know the cost of deploying resources and how to architect for cost optimization. Practice eliminating the waste and bring accountability in every step of the build process. Use mandatory cost tags on all of your resources to gain precise insights into

usage. Define IAM policies to enforce tag usage, and use tagging tools, such as AWS Config and AWS Tag Editor, to manage tags. Be cost-conscious, reduce the usage by terminating unused instances, and delete old snapshots and unused keys.

Right size your infrastructure by matching instance types and sizes, and set periodic checks to ensure that the initial provision remains optimum as your business changes over time. With Amazon EC2, you can choose the combination of instance types and sizes most appropriate for your applications. Amazon RDS instances are also optimized for memory, performance, and I/O.

Amazon EC2 Reserved Instances provide you with a significant discount (up to 75 percent) compared to On-Demand Instance pricing. Using Convertible Reserved Instances, you can change instance families, OS types, and tenancies while benefitting from Reserved Instance pricing. Reserved Instance Marketplace allows you to sell the unused Reserved Instances or buy them from other AWS customers, usually at lower prices and shorter terms. With size flexibility, discounted rates for Amazon RDS Reserved Instances are automatically applied to the usage of any size within the instance family.

Spot Instances provide an additional option for obtaining compute capacity at a reduced cost and can be used along with On-Demand and Reserved Instances. Spot Fleets enable you to launch and maintain the target capacity and to request resources automatically to replace any that are disrupted or manually terminated. Using the termination notices and persistent requests in your application design help to maintain continuity as the result of interruptions.

AWS Auto Scaling automatically scales if your application experiences variable load and uses one or more scalable resources, such as Amazon ECS, Amazon DynamoDB, Amazon Aurora, Amazon EC2 Spot requests, and Amazon EC2 scaling groups. Predictive Scaling uses machine learning models to forecast daily and weekly patterns. Amazon EC2 Auto Scaling enables you to scale in response to demand and known load schedules. It supports the provisioning of scale instances across purchase options, Availability Zones, and instance families to optimize performance and cost.

Containers provide process isolation and improve the resource utilization. Amazon ECS lets you easily build all types of containerized applications and launch thousands of containers in seconds with no additional complexity. With AWS Fargate technology, you can manage containers without having to provision or manage servers. It enables you to focus on building and running applications, not the underlying infrastructure.

AWS Lambda takes care of receiving events or client invocations and then instantiates and runs the code. That means there's no need to manage servers. Serverless services have built-in automatic scaling, availability, and fault tolerance. These features allow you to focus on product innovation and rapidly construct applications, such as web applications, websites, web-hooked systems, chatbots, and clickstream.

AWS storage services are optimized to meet different storage requirements. Use the Amazon S3 analytics feature to analyze storage access patterns to help you decide when to transition the right data to the right storage class and to yield considerable savings. Monitor Amazon EBS volumes periodically to identify ones that are unattached or appear to be underutilized or overutilized, and adjust provisioning to match actual usage.

Optimizing data transfer ensures that you minimize data transfer costs. Use options such as Amazon CloudFront, Amazon S3 transfer acceleration, and Amazon Route 53 to let data reach Regions faster and reduce latency issues.

NoSQL database systems like Amazon DynamoDB use alternative models for data management, such as key-value pairs or document storage. DynamoDB enables you to offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. Follow best practices, such as distributing data evenly, effective partition and sort keys usage, efficient data scanning, and using sparse indexes for maximizing performance and minimizing throughput costs, when working with Amazon DynamoDB.

AWS provides several tools to help you identify those cost-saving opportunities and keep your resources right-sized. AWS Trusted Advisor inspects your AWS environment to identify idle and underutilized resources and provides real-time insight into service usage to help you improve system performance and save money. Amazon CloudWatch collects and tracks metrics, monitors log files, sets alarms, and reacts to changes in AWS resources automatically. AWS Cost Explorer checks patterns in AWS spend over time, projects future costs, identifies areas that need further inquiry, and provides Reserved Instance recommendations.

Optimization is an ongoing process. Always stay current with the pace of AWS new releases, and assess your existing design solutions to ensure that they remain cost-effective.

# Exam Essentials

**Know the importance of tagging.** By using tags, you can assign metadata to AWS resources. This tagging makes it easier to manage, search for, and filter resources in billing reports and automation activities and when setting up access controls.

**Know about various tagging tools and how to enforce the tag rules.** With AWS Tag Editor, you can add tags to multiple resources at once, search for the resources that you want to tag, and then add, remove, or edit tags for the resources in your search results. AWS Config identifies resources that do not comply with tagging policies. You can use IAM policy conditions to force the usage of tags while creating the resources.

**Know the fundamental practices in reducing the usage.** Follow the best practices of cost optimization in every step of your build process, such as turning off unused resources, spinning up instances only when needed, and spinning them down when not in use. Use tagging to help with the cost allocation. Use Amazon EC2 Spot Instances, Amazon EC2, and Reserved Instances where appropriate, and use alerts, notifications, and cost-management tools to stay on track.

**Know the various usage patterns for right sizing.** By understanding your business use case and backing up the analysis with performance metrics, you can choose the most

appropriate options, such as steady state; variable; predictable, but temporary; and development, test, and production usage.

**Know the various instance families for right sizing and the corresponding use cases.** Amazon EC2 provides a wide selection of instances to match capacity needs at the lowest cost and comes with different options for CPU, memory, and network resources. The families include General Purpose, Compute Optimized, Memory Optimized, Storage Optimized, and Accelerated Computing.

**Know Amazon EC2 Auto Scaling benefits and how this feature can make your solutions more optimized and highly available.** AWS Auto Scaling is a fast, easy way to optimize the performance and costs of your applications. It makes smart scaling decisions based on your preferences, automatically maintains performance even when your workloads are periodic, unpredictable, and continuously changing.

**Know how to create a single AWS Auto Scaling group to scale instances across different purchase options.** You can provision and automatically scale Amazon EC2 capacity across different Amazon EC2 instance types, Availability Zones, and On-Demand, Reserved Instances, and Spot purchase options in a single AWS Auto Scaling group. You can define the desired split between On-Demand and Spot capacity, select which instance types work for your application, and specify preferences for how Amazon EC2 Auto Scaling should distribute the AWS Auto Scaling group capacity within each purchasing model.

**Know how block, object, and file storages are different.** Block storage is commonly dedicated, low-latency storage for each host, and it is provisioned with each instance. Object storage is developed for the cloud, has vast scalability, is accessed over the web, and is not directly attached to an instance. File storage enables accessing shared files as a file system.

**Know key CloudWatch metrics available to measure the Amazon EBS efficiency and how to use them.** CloudWatch metrics are statistical data that you can use to view, analyze, and set alarms on the operational behavior of your volumes. Depending on your needs, set alarms and response actions that correspond to each data point. For example, if your I/O latency is higher than you require, check the metric `VolumeQueueLength` to make sure that your application is not trying to drive more IOPS than you have provisioned. Review and learn more about the available metrics that help optimize the block storage.

**Know tools and features that help in efficient data transfer.** Using Amazon CloudFront, you can locate data closer to users and reduce administrative efforts to minimize data transfer costs. Amazon S3 Transfer Acceleration enables fast data transfer over an optimized network path. Use the multipart upload file option while uploading a large file to improve network throughput.

**Know key differences between RDBMS and NoSQL databases to design efficient solutions using Amazon DynamoDB.** Schema flexibility and the ability to store related items together make DynamoDB a solution for solving problems associated with changing business needs and scalability issues unlike relational databases.

**Know the importance of distributing the data evenly when designing DynamoDB tables.** Use provisioned throughput more efficiently by making the partition key more distinct. That way, data spreads throughout the provisioned space. Use the sort key with the

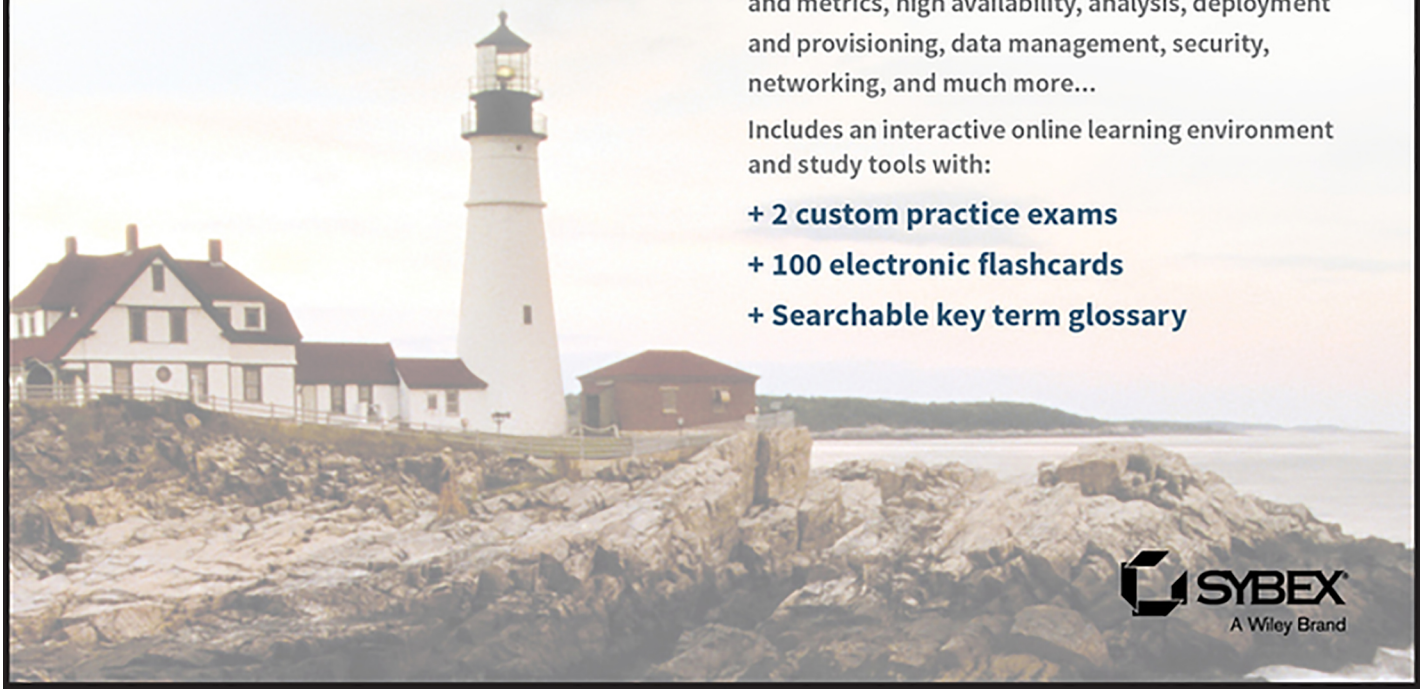
amazon web services

Stephen Cole, Gareth Digby, Chris Fitch,
Steve Friedberg, Shaun Qualheim, Jerry Rhoads,
Michael Roth, Blaine Sundrud

# AWS Certified SysOps Administrator

## OFFICIAL STUDY GUIDE

**ASSOCIATE EXAM**



Covers exam objectives, including monitoring and metrics, high availability, analysis, deployment and provisioning, data management, security, networking, and much more...

Includes an interactive online learning environment and study tools with:

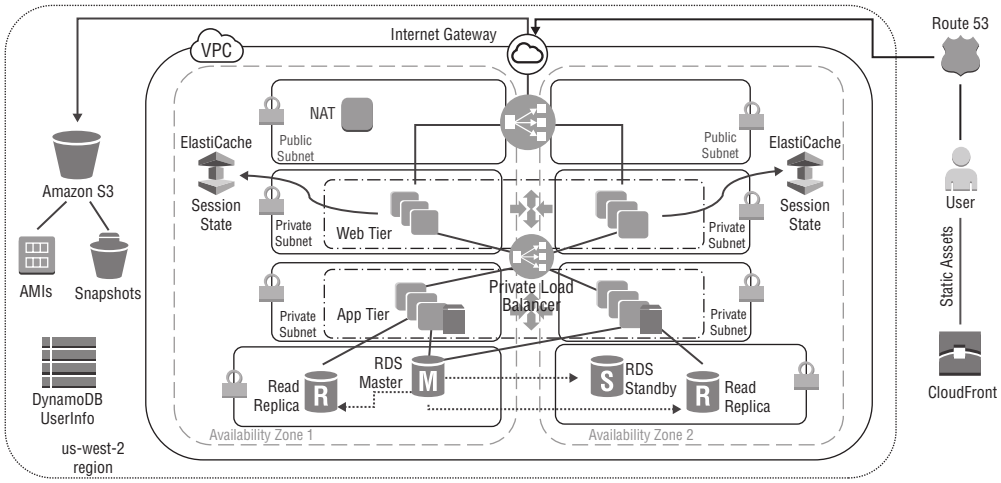
**+ 2 custom practice exams**

**+ 100 electronic flashcards**

**+ Searchable key term glossary**

SYBEX
A Wiley Brand

**The Solution**




As we examine the pieces of the solution, we start by breaking down the components of the architecture. Then we focus on how systems operators interact with the individual pieces and begin thinking about how those pieces fit into the certification exam.

## Environment

Architectures live inside *AWS Regions*; in this scenario, in us-west-2 (Oregon, United States). Regions are made up of multiple *Availability Zones*, which provide the foundation for highly available architectures. Although this is a systems operation exam, it is critical to understand the nature of AWS Regions and Availability Zones.


Each AWS Region is a separate geographic area. Each AWS Region has multiple, isolated locations known as *Availability Zones*. *AWS Regions* and *Availability Zones* are discussed in Chapter 5, "Networking."

## Networking

Networking components start inside the AWS Region with Amazon Virtual Private Cloud (Amazon VPC). *Amazon VPC* is a private network in the AWS Region that isolates all traffic from the millions of other applications running in AWS. A deep dive into Amazon VPC (and the rest of its components) is found in Chapter 5.

Amazon VPC is divided into *subnets*; all assets running in your Amazon VPC are assigned to a subnet. Unlike on-premises subnetting decisions that can affect latency between servers, Amazon VPC subnets only affect access. Access between subnets is

controlled through *network Access Control Lists (nACLs)*, and access in and out of Amazon VPC is controlled through attached gateways. In this scenario, the only gateway is the *Internet Gateway (IGW)*, and it allows traffic to and from external (public IP) sources.

By granting route table access to the gateway only to specific subnets, ingress and egress can be tightly controlled. In this scenario, public subnets indicate IGW access. Without IGW access, the subnets become private; that is, they are accessible only to private IP networks.


To learn about the other gateways that could be leveraged to create hybrid or other private architectures, refer to Chapter 5.

*Security groups* are often part of the networking discussion. They provide stateful firewalls that operate at the Hypervisor levels for all individual*Amazon Elastic Compute Cloud (Amazon EC2)* instances and other Amazon VPC objects. In this scenario, we potentially have seven different security groups:

**Public Elastic Load Balancing** The only security group that allows full public access

**Web Tier Amazon EC2** This accepts traffic only from public Elastic Load Balancing.

**Private Elastic Load Balancing** This accepts traffic only from Web Tier Amazon EC2.

**Application Tier Amazon EC2** This accepts traffic only from private Elastic Load Balancing.

**Amazon ElastiCache** This accepts traffic only from Application Tier Amazon EC2.

**Amazon Relational Database Service (Amazon RDS)** This accepts traffic only from Application Tier Amazon EC2.

**Network Address Translation (NAT)** This is used only for internally initiated outbound traffic.

By specifically stacking security groups in this manner, you can provide layers of network security that surround the database portion of the three-tier design.

## Compute

In this scenario, you use traditional compute methods, such as Linux servers running on Amazon EC2. Amazon EC2 comes in many sizes (how many CPUs, how much memory, how much network capacity, and so on), known as *instances*. Based on the Amazon Machine Image (AMI), each Amazon EC2 instance can run a wide range of Linux- or Windows-based operating systems as well as preinstalled software packages. Amazon EC2 instances also support runtime configuration as required.

The requirements for the scenario include scalable solutions. AWS provides Auto Scaling as an engine that can take predefined launch configurations and dynamically add or remove instances from the web or the Application Tier based on metrics.

Details on Amazon EC2, Auto Scaling, and other compute resources are found in Chapter 4, "Compute."

## Database

*Amazon RDS* runs in your Amazon VPC on Amazon EC2. You select the database engine and version (MySQL, Oracle, Postgres, and so forth) and the configuration (the size of the Amazon EC2 instance, which subnets to use, how often to take backups, and so on). Amazon RDS takes care of the infrastructure of the instances and the engine; your database administrator (DBA) takes care of the database schema and data.

This scenario also includes *Amazon DynamoDB*, a native NoSQL engine optimized for consistent low latency, high availability, and strongly consistent reads and writes. Unlike Amazon RDS (or do-it-yourself databases running on Amazon EC2), Amazon DynamoDB operates at the regional level through API access only.

For details on how Amazon DynamoDB and other databases function, refer to Chapter 7, "Databases."

## Storage

This scenario looks at storage in three different areas: the block storage used by the Amazon EC2 instances, the object storage keeping all of the media as well as backups and AMIs, and the caching storage used by Amazon CloudFront.

*Amazon EBS* is durable, persistent block storage used by most Amazon EC2 and Amazon RDS instances. It provides drive space for boot volumes and data volumes. Additionally, AWS provides ephemeral storage for many Amazon EC2 instance types through instance storage. Deciding which one to use becomes an operational value judgment, one that compares speed, persistence, and cost.

Object storage is provided by Amazon S3. *Amazon S3*, like Amazon DynamoDB, operates at the regional level outside Amazon VPC. It is only accessed through API commands that your operations team controls with fine-grained precision. Highly cost-effective and massively durable, Amazon S3 provides web-enabled storage for content as well as protected storage for database backups and AMI storage.

*Amazon CloudFront* is the *AWS content delivery network service (CDN).* This application leverages Amazon CloudFront to cache content close to consumers in order to improve performance (reduce latency) and reduce costs.

Storage systems, including shared file systems, the Amazon Elastic File System (Amazon EFS), and cold storage via Amazon Glacier, are discussed in Chapter 6, "Storage."

## User Management

Although not drawn in the sample three-tier architecture diagram, user management becomes one of the critical elements of the AWS operational design. Operator access is controlled through *AWS Identity and Access Management (IAM)*. IAM maintains control over validating authentication methods (passwords, access keys, and so on) and then grants access to authenticated operators.

Because everything in AWS is accessed through APIs, IAM becomes a comprehensive tool for controlling all permissions to AWS services and resources.

For established enterprise customers, IAM can be integrated with existing directory systems via AWS Directory Service.


AWS IAM controls access to AWS services and resources. It does not control access to the Amazon EC2 operating system or application-level authentication. For more details, refer to the shared responsibility model in Chapter 3, "Security and AWS Identity and Access Management (IAM)."

## Security, Monitoring, and Deployment

Security is integral to every part of the AWS platform. This means that security is part of each piece of the architecture.


There are some specific AWS security tools, such as Amazon Inspector, Amazon VPC Flow Logs, Amazon CloudWatch Logs, and others which provide a more focused toolset that the AWS operations team can leverage to ensure the security profile of the AWS application. These and many other tools are discussed in Chapter 3.

Monitoring of critical systems is provided by *Amazon CloudWatch*, which provides visibility into metrics that happen on the Customer side of the shared responsibility model. Thousands of metrics across more than 90 services keep track of everything from CPU consumption to latency, queue depths, and so on.

*AWS CloudTrail* records every API call in the AWS system, including:

- Who made the API call
- When the API call was performed
- Where the API call originated
- The result of the API call

These records and other log files are processed through Amazon CloudWatch Logs, which analyze text data for patterns that trigger alerts and corresponding actions.

Automated deployment methods ensure that human error does not disrupt rollouts or updates to production or sandbox environments. *AWS CloudFormation* turns infrastructure plans into code, allowing your operations team to build and tear down entire systems in a single action. Refer to Chapter 8, "Application Deployment and Management," for more details.

## Key Products: Three-Tier Design

As described above, the three-tier architecture consists of a web front end, an application layer, and database layer. In addition to the compute, storage, and database resources, additional AWS infrastructure may need to be deployed. Refer to Table 1.1 for a list of key products.

**TABLE 1.1** Key Products: Three-Tier Architecture

| Tools to Enable Hybrid Cloud Architectures | Description |
|---|---|
| AWS Regions and Availability Zones | Amazon EC2 is hosted in multiple locations world-wide. These locations are composed of regions and Availability Zones. Each region is a separate geographic area. Amazon EC2 provides you with the ability to place resources, such as instances and data, in multiple locations. |
| Availability Zones | Within a region are two or more Avalability Zones. The Availability Zone is isolated, but the Availability Zones in a region are connected through low-latency links. |
| Edge Locations | Edge Locations = AWS Lambda@Edge Amazon CloudFront, Amazon Route 53, AWS Shield, and AWS WAF services that are offered at AWS Edge Locations. |
| Hybrid cloud architecture | Integration of on-premises resources with cloud resources |
| Amazon Route 53 | Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. |
| Amazon CloudFront | Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content to your users. CloudFront delivers your content through a worldwide network of data centers called *edge locations*. |
| Amazon VPC | A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS cloud. You can launch your AWS resources, such as Amazon EC2 instances, into your VPC. |
| Internet Gateways | An Internet gateway is a horizontally-scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the Internet. |
| Subnets | A subnetwork or subnet is a logical subdivision of an IP network. |
| Route tables | A route table is a set of rules that is used to determine where data packets traveling over an Internet Protocol (IP) network will be directed. |

5. Use groups to assign permissions to IAM users.
6. Enable MFA for privileged users.
7. Use IAM roles for applications that run on Amazon EC2 instances.
8. Delegate by using IAM roles instead of sharing credentials.
9. Rotate credentials regularly.
10. Remove unnecessary credentials.
11. Use policy conditions for extra security.
12. Monitor activity on your AWS account.
13. Remove root credentials.
14. Use access levels to review IAM permissions.
15. Use AWS-defined policies to assign permissions whenever possible.
16. Use IAM roles to provide cross-account access.

So far, you have learned about the shared responsibility model and how to secure your IAM account. Now let's talk about securing your AWS resources.

# Securing Your AWS Cloud Services

In this section, we discuss Amazon EC2 key pairs, X.509 certificates, AWS Key Management Services (AWS KMS), and AWS CloudHSM.

## Key Pairs

Amazon EC2 instances created from a public AMI use a public/private key pair instead of a password for signing in via SSH. The public key is embedded in your instance, and you use the private key to sign in securely without a password. After you create your own AMIs, you can choose other mechanisms to log in securely to your new instances. You can have a key pair generated automatically for you when you launch the instance, or you can upload your own. Save the private key in a safe place on your system, and record the location where you saved it. For Amazon CloudFront, you use key pairs to create signed URLs for private content, such as when you want to distribute restricted content that someone paid for.


Amazon CloudFront key pairs can be created only by the root account and cannot be created by IAM users.

### X.509 Certificates

X.509 certificates are used to sign SOAP-based requests. X.509 certificates contain a public key and additional metadata (for example, an expiration date that AWS verifies when

## Amazon Inspector

*Amazon Inspector* is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for vulnerabilities or deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed list of security findings prioritized by level of severity.

Amazon Inspector includes a knowledge base of hundreds of rules mapped to common security best practices and vulnerability definitions. Examples of built-in rules include checking for remote root login being enabled or vulnerable software versions being installed. These rules are regularly updated by AWS security researchers. More information about AWS Inspector can be found in Chapters 4 and 9.

## AWS Certificate Manager

*AWS Certificate Manager* is a service that lets you provision, manage, and deploy SSL/TLS certificates for use with AWS Cloud services. SSL/TLS certificates are used to secure network communications and establish the identity of websites over the Internet. With AWS Certificate Manager, you can request a certificate and deploy it on AWS resources, such as Elastic Load Balancing load balancers or Amazon CloudFront distributions. AWS Certificate Manager then handles the certificate renewals.

## AWS Web Application Firewall (AWS WAF)

*AWS Web Application Firewall (AWS WAF)* is a web application firewall that helps protect web applications from attacks by allowing you to configure rules that allow, block, or monitor (count) web requests based on conditions that you define. These conditions include IP addresses, HTTP headers, HTTP body, Uniform Resource Identifier (URI) strings, SQL injection, and cross-site scripting.

As the underlying service receives requests for your websites, it forwards those requests to AWS WAF for inspection against your rules. Once a request meets a condition defined in your rules, AWS WAF instructs the underlying service either to block or allow the request based on the action you define.

AWS WAF is tightly integrated with Amazon CloudFront and the Application Load Balancer, services that AWS customers commonly use to deliver content for their websites and applications. When you use AWS WAF on Amazon CloudFront, your rules run in all AWS edge locations around the world close to your end users. This means that security doesn't come at the expense of performance. Blocked requests are stopped before they reach your web servers. When you use AWS WAF on Application Load Balancer, your rules run in-region and can be used to protect Internet-facing and internal load balancers.

## AWS Trusted Advisor

The *AWS Trusted Advisor* customer support service not only monitors cloud performance and resiliency, but also cloud security. AWS Trusted Advisor inspects your AWS environment and makes recommendations when opportunities may exist to save money, improve

This access can only be modified through the invocation of Amazon VPC APIs. AWS supports the ability to grant granular access to different administrative functions on the instances and the Internet gateway, enabling you to implement additional security through separation of duties.

**Dedicated instances** Within an Amazon VPC, you can launch Amazon EC2 instances that are physically isolated at the host hardware level (that is, they will run on single-tenant hardware). An Amazon VPC can be created with "dedicated" tenancy so that all instances launched into the Amazon VPC will use this feature. Alternatively, an Amazon VPC may be created with "default" tenancy, but you can specify dedicated tenancy for particular instances launched into it. More information on networking on AWS can be found in Chapter 5.

**Dedicated hosts** An *Amazon EC2 Dedicated Host* is a physical server with Amazon EC2 instance capacity fully dedicated to your use. Dedicated hosts allow you to use your existing per-socket, per-core, or per-virtual machine software licenses, including Windows Server, Microsoft SQL Server, SUSE, Linux Enterprise Server, and so on. More information on dedicated hosts can be found in Chapter 4.

## Amazon CloudFront Security

*Amazon CloudFront* gives customers an easy way to distribute content to end users with low latency and high data transfer speeds. It delivers dynamic, static, and streaming content using a global network of edge locations. Requests for customers' objects are automatically routed to the nearest edge location, so content is delivered with the best possible performance. Amazon CloudFront is optimized to work with other AWS Cloud services like Amazon S3, Amazon EC2, Elastic Load Balancing, and Amazon Route 53. It also works seamlessly with any non-AWS origin server that stores the original, definitive versions of your files.

Amazon CloudFront requires that every request made to its control API is authenticated so that only authorized users can create, modify, or delete their own Amazon CloudFront distributions. Requests are signed with an HMAC-SHA-1 signature calculated from the request and the user's private key. Additionally, the Amazon CloudFront control API is only accessible via SSL-enabled endpoints.

There is no guarantee of durability of data held in Amazon CloudFront edge locations. The service may sometimes remove objects from edge locations if those objects are not requested frequently. Durability is provided by Amazon S3, which works as the origin server for Amazon CloudFront by holding the original, definitive copies of objects delivered by Amazon CloudFront.

If you want control over who can download content from Amazon CloudFront, you can enable the service's private content feature. This feature has two components. The first controls how content is delivered from the Amazon CloudFront edge location to viewers on the Internet. The second controls how the Amazon CloudFront edge locations access objects in Amazon S3. Amazon CloudFront also supports geo-restriction, which restricts access to your content based on the geographic location of your viewers.

To control access to the original copies of your objects in Amazon S3, Amazon CloudFront allows you to create one or more origin access identities and associate these with your distributions. When an origin access identity is associated with an Amazon CloudFront distribution, the distribution will use that identity to retrieve objects from Amazon S3. You can then use Amazon S3's ACL feature, which limits access to that origin access identity so that the original copy of the object is not publicly readable.

To control who can download objects from Amazon CloudFront edge locations, the service uses a signed-URL verification system. To use this system, you first create a public-private key pair and upload the public key to your account via the AWS Management Console. You then configure your Amazon CloudFront distribution to indicate which accounts you would authorize to sign requests. You can indicate up to five AWS accounts that you trust to sign requests. As you receive requests, you will create policy documents indicating the conditions under which you want Amazon CloudFront to serve your content. These policy documents can specify the name of the object that is requested, the date and time of the request, and the source IP (or CIDR range) of the client making the request. You then calculate the SHA-1 hash of your policy document and sign this using your private key. Finally, you include both the encoded policy document and the signature as query string parameters when you reference your objects. When Amazon CloudFront receives a request, it will decode the signature using your public key. Amazon CloudFront will only serve requests that have a valid policy document and matching signature.

Note that private content is an optional feature that must be enabled when you set up your Amazon CloudFront distribution. Content delivered without this feature enabled will be publicly readable.

Amazon CloudFront provides the option to transfer content over an encrypted connection (HTTPS). By default, Amazon CloudFront will accept requests over both HTTP and HTTPS protocols. You can also configure Amazon CloudFront to require HTTPS for all requests or have Amazon CloudFront redirect HTTP requests to HTTPS. You can even configure Amazon CloudFront distributions to allow HTTP for some objects but require HTTPS for other objects. More information on Amazon CloudFront can be found in Chapters 5 and 6.

## Storage

AWS provides low-cost data storage with high durability and availability. AWS offers storage choices for backup, archiving, disaster recovery, and block and object storage.

### Amazon Simple Storage Service (Amazon S3) Security

*Amazon S3* allows you to upload and retrieve data at any time from anywhere on the web. Amazon S3 stores data as objects in buckets. An object can be any kind of file: a text file, a photo, a video, and more. When you add a file to Amazon S3, you have the option of including metadata with the file and setting permissions to control access to the file. For each bucket, you can control access to the bucket (who can create, delete, and list objects in the bucket), view access logs for the bucket and its objects, and choose the geographical region where Amazon S3 will store the bucket and its contents.

resource. If both record sets are unhealthy, Amazon Route 53 returns the primary resource record set. Health checks are discussed in greater detail in Chapter 10.

You can combine routing (for example, have geolocation routing backed up with failover routing) to make sure that you provide the highest level of availability possible. As you can imagine, this can get very complex (and confusing) very quickly, so good documentation is important.

## DNS Record Types

Explaining the various DNS records types is out of the scope of this book. However, Table 5.7 shows the supported record types for Amazon Route 53.

**TABLE 5.7** Amazon Route 53 Supported DNS Record Types

| Record Type | Description |
|---|---|
| A | Address mapping records |
| AAAA | IPv6 address records |
| CNAME | Canonical name records |
| MX | Mail exchanger record |
| NAPTR | Name authority pointer record |
| NS | Name server records |
| PTR | Reverse-lookup Pointer records |
| SOA | Start of authority records |
| SPF | Sender policy framework record |
| SRV | Service record |
| TXT | Text records |

In addition to the standard DNS record types supported, Amazon Route 53 supports a record type called *Alias*. An *Alias record type*, instead of pointing to an IP address or a domain name, points to one of the following:

- An Amazon CloudFront distribution
- An AWS Elastic Beanstalk environment

- An Elastic Load Balancing Classic or Application Load Balancer
- An Amazon S3 bucket that is configured as a static website
- Another Amazon Route 53 resource record set in the same hosted zone

## Health Checks

There are three types of health checks that you can configure with Amazon Route 53. They are as follows:

- The health of a specified resource, such as a web server
- The status of an Amazon CloudWatch alarm
- The status of other health checks

In this section, we explore each type. The level of detail covered may not be tested on the exam. However, as an AWS Certified Systems Operator, the material covered here is a must-know.

**The health of a specified resource, such as a web server** You can configure a health check that monitors an endpoint that you specify either by IP address or by domain name. At regular intervals that you specify, Amazon Route 53 submits automated requests over the Internet to your application, server, or other resource to verify that it's reachable, available, and functional. Optionally, you can configure the health check to make requests similar to those that your users make, such as requesting a web page from a specific URL.

**The status of an Amazon CloudWatch alarm** You can create CloudWatch alarms that monitor the status of CloudWatch metrics, such as the number of throttled read events for an Amazon DynamoDB database or the number of Elastic Load Balancing hosts that are considered healthy. After you create an alarm, you can create a health check that monitors the same data stream that CloudWatch monitors for the alarm.

To improve resiliency and availability, Amazon Route 53 doesn't wait for the CloudWatch alarm to go into the ALARM state. The status of a health check changes from healthy to unhealthy based on the data stream and on the criteria in the CloudWatch alarm. The status of a health check can change from healthy to unhealthy even before the state of the corresponding alarm has changed to ALARM in CloudWatch.

**The status of other health checks** You can create a health check that monitors whether Amazon Route 53 considers other health checks healthy or unhealthy. One situation where this might be useful is when you have multiple resources that perform the same function, such as multiple web servers, and your chief concern is whether some minimum number of your resources is healthy. You can create a health check for each resource without configuring notification for those health checks. Then you can create a health check that monitors the status of the other health checks and that notifies you only when the number of available web resources drops below a specified threshold.

## Amazon Route 53 Management

You can access Amazon Route 53 in the following ways:

- AWS Management Console
- AWS SDKs
- Amazon Route 53 API
- AWS CLI
- AWS Tools for Windows PowerShell

The best tool for monitoring the status of your domain is the Amazon Route 53 dashboard. This dashboard will give a status of any new domain registrations, domain transfers, and any domains approaching expiration.

The tools used to monitor your DNS service with Amazon Route 53 are health checks, Amazon CloudWatch, and AWS CloudTrail. Health checks are discussed in the management section. Amazon CloudWatch monitors metrics like the number of health checks listed as healthy, the length of time an SSL handshake took, and the time it took for the health check to receive the first byte, among other metrics. AWS CloudTrail can capture all of the API requests made for Amazon Route 53. You can determine the user who invoked a particular API.

### Amazon Route 53 Authentication and Access Control

To perform any operation on Amazon Route 53 resources, such as registering a domain or updating a resource record set, IAM requires you to authenticate to prove that you're an approved AWS user. If you're using the Amazon Route 53 console, you authenticate your identity by providing your AWS user name and a password. If you're accessing Amazon Route 53 programmatically, your application authenticates your identity for you by using access keys or by signing requests.

After you authenticate your identity, IAM controls your access to AWS by verifying that you have permissions to perform operations and to access resources. If you are an account administrator, you can use IAM to control the access of other users to the resources that are associated with your account.

# Amazon CloudFront

*Amazon CloudFront* is a web service that speeds up distribution of your static and dynamic web content—for example, `.html`, `.css`, `.php`, image, and media files—to end users. Amazon CloudFront delivers your content through a worldwide network of edge locations.

When an end user requests content that you're serving with Amazon CloudFront, the user is routed to the edge location that provides the lowest latency, so content is delivered with the

best possible performance. If the content is already in that edge location, Amazon CloudFront delivers it immediately. If the content is not currently in that edge location, Amazon CloudFront retrieves it from an Amazon S3 bucket or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

Amazon CloudFront can be used to distribute static content, dynamic content, and streaming media.

## Amazon CloudFront Implementation

When implementing Amazon CloudFront, the first step is to configure your origin servers, from which Amazon CloudFront gets your files for distribution from Amazon CloudFront edge locations all over the world. An origin server stores the original, definitive version of your objects.

If you're serving content over HTTP, your origin server is either an Amazon S3 bucket or an HTTP server, such as a web server. Your HTTP server can run on an Amazon EC2 instance or on a server that you manage; these servers are also known as custom origins. If you distribute media files on demand using the Adobe Media Server Real-Time Messaging Protocol (RTMP), your origin server is always an Amazon S3 bucket.

The next step is to upload your files to your origin servers. Your files, also known as objects, typically include web pages, images, and media files, but they can be anything that can be served over HTTP or a supported version of Adobe RTMP, the protocol used by Adobe Flash Media Server.

The final step is to create an Amazon CloudFront distribution, which tells Amazon CloudFront which origin servers to get your files from when users request the files through your website or application. In addition, you can configure your origin server to add headers to the files; the headers indicate how long you want the files to stay in the cache in Amazon CloudFront edge locations.

At this point, Amazon CloudFront assigns a domain name to your new distribution and displays it in the Amazon CloudFront console or returns it in the response to a programmatic request. You can also configure your Amazon CloudFront distribution so that you can use your own domain name.

Now that we have discussed (at a very high level) the steps required to Implement an Amazon CloudFront distribution, let's talk about how that distribution is configured. The following are features that relate to Amazon CloudFront web distributions.

### Cache Behaviors

A *cache behavior* is the set of rules that you configure for a given URL pattern based on file extensions, file names, or any portion of a URL path on your website (for example, `*.jpg`). You can configure multiple cache behaviors for your web distribution. Amazon CloudFront will match incoming viewer requests with your list of URL patterns, and if there is a match, the service will honor the cache behavior that you configure for that URL pattern. Each cache behavior can include the following Amazon CloudFront configuration values: origin server name, viewer connection protocol, minimum expiration period, query string parameters, and trusted signers for private content.

## Regional Edge Caches

You can use Amazon CloudFront to deliver content at improved performance for your viewers while reducing the load on your origin resources. *Regional Edge Caches* sit in between your origin web server and the global edge locations that serve traffic directly to your viewers. As the popularity of your objects is reduced, individual edge locations may evict those objects to make room for more popular content. Regional Edge Caches have larger cache width than any individual edge location, so objects remain in cache longer at these Regional Edge Caches. This helps keep more of your content closer to your viewers, reducing the need for CloudFront to go back to your origin web server and improves overall performance for viewers. For instance, all our edge locations in Europe now go to the Regional Edge Cache in Frankfurt to fetch an object before going back to your origin web server. Regional Edge Cache locations are currently utilized only for requests that need to go back to a custom origin; requests to Amazon S3 origins skip Regional Edge Cache locations.

You do not need to make any changes to your Amazon CloudFront distributions; this feature is enabled by default for all CloudFront distributions. There is no additional cost for using this feature.

## Origin Servers

You can configure one or more origin servers for your Amazon CloudFront web distribution. Origin servers can be an AWS resource, such as Amazon S3, Amazon EC2, Elastic Load Balancing, or a custom origin server outside of AWS. Amazon CloudFront will request content from each origin server by matching the URLs requested by the viewer with rules that you configure for your distribution. This feature allows you the flexibility to use each AWS resource for what it's designed for—Amazon S3 for storage, Amazon EC2 for compute, and so forth—without the need to create multiple distributions and manage multiple domain names on your website. You can also continue to use origin servers that you already have set up without the need to move data or re-deploy your application code. Furthermore, Amazon CloudFront allows the directory path as the origin name; that is, when you specify the origin for a CloudFront distribution, you can specify a directory path in addition to a domain name. This makes it easier for you to deliver different types of content via CloudFront without changing your origin infrastructure.

## Private Content

You can use Amazon CloudFront's private content feature to control who is able to access your content. This optional feature lets you use Amazon CloudFront to deliver valuable content that you prefer not to make publicly available by requiring your users to use a signed URL or have a signed HTTP cookie when requesting your content.

## Device Detection

Amazon CloudFront edge locations can look at the value of the User Agent header to detect the device type of all the incoming requests. Amazon CloudFront can determine whether the end user request came from a desktop, tablet, smart TV, or mobile device and pass that

information in the form of new HTTP headers to your origin server—Amazon EC2, Elastic Load Balancing, or your custom origin server. Your origin server can use the device type information to generate different versions of the content based on the new headers. Amazon CloudFront will also cache the different versions of the content at that edge location.

## Geo Targeting

Amazon CloudFront can also detect the country from where the end users are accessing your content. Amazon CloudFront can then pass the information about the country in a new HTTP header to your custom origin server. Your origin server can generate different versions of the content for users in different countries and cache these different versions at the edge location to serve subsequent users visiting your website from the same country.

## Cross-Origin Resource Sharing

Amazon CloudFront may be configured to forward the origin header value so that your origin server (Amazon S3 or a custom origin) can support cross-origin access via *Cross-Origin Resource Sharing (CORS)*. CORS defines a way for client web applications that are loaded in one domain to interact with resources in a different domain.

## Viewer Connection Protocol

Content can be delivered to viewers using either the HTTP or HTTPS protocol. By default, your web distribution will accept requests on either protocol. However, if you want all of your content or certain URLs delivered only over an HTTPS connection, you can configure your distribution to only accept requests that come over HTTPS for that content. You can configure this feature separately for each URL pattern in your web distribution as part of the cache behavior for that URL pattern.

## Protocol Detection

You can configure Amazon CloudFront to include the protocol (HTTP vs. HTTPS) of your end user's request as part of the cache key to identify an object uniquely in cache. This allows you to customize your content based on the protocol that your end users are using to access your content.

## Custom SSL

Custom SSL certificate support lets you deliver content over HTTPS using your own domain name and your own SSL certificate. This gives visitors to your website the security benefits of Amazon CloudFront over an SSL connection that uses your own domain name in addition to lower latency and higher reliability. You can also configure CloudFront to use HTTPS connections for origin fetches so that your data is encrypted end-to-end from your origin to your end users. Configuring custom SSL certificate support is easy; you don't need to learn any proprietary code or hire any consultants to configure it for you.

You can provision SSL/TLS certificates and associate them with Amazon CloudFront distributions within minutes. Simply provision a certificate using the new AWS Certificate

Manager (ACM) and deploy it to your CloudFront distribution with a couple of clicks. Then let ACM manage certificate renewals for you. ACM allows you to provision, deploy, and manage the certificate with no additional charges.


Amazon CloudFront still supports using certificates that you obtained from a third-party certificate authority and uploaded to the IAM certificate store.

## Geo Restriction

*Geo Restriction* or *Geoblocking* lets you choose the countries in which you want to restrict access to your content. By configuring either a whitelist or a blacklist of countries, you can control delivery of your content through Amazon CloudFront only to countries where you have the license to distribute. To enable this feature, you can either use the Amazon CloudFront API or the Amazon CloudFront Management Console. When a viewer from a restricted country submits a request to download your content, Amazon CloudFront responds with an HTTP status code 403 (Forbidden). You can also configure Custom Error Pages to customize the response that Amazon CloudFront sends to your viewers.

## TTL Settings: Min, Max, and Default TTL

Amazon CloudFront lets you configure a minimum time-to-live (Min TTL), a maximum TTL (Max TTL), and a default TTL to specify how long CloudFront caches your objects in edge locations. Amazon CloudFront uses the expiration period that your origin sets on your files (through Cache-Control headers) to determine whether CloudFront needs to check the origin for an updated version of the file. If you expect that your files will change frequently, you can configure your origin to set a short expiration period on your files. Amazon CloudFront accepts expiration periods as short as 0 seconds (in which case CloudFront will revalidate each viewer request with the origin). Amazon CloudFront also honors special Cache-Control directives such as private, no-store, and so on. These are often useful when delivering dynamic content that you don't want CloudFront to cache.

If you have not set a Cache-Control header on your files, Amazon CloudFront uses the value of Default TTL to determine how long the file should be cached at the edge before Amazon CloudFront checks the origin for an updated version of the file. If you don't want to rely on the Cache-Control headers set by your origin, you can now easily override the Cache-Control headers by setting the same value for Max TTL, Min TTL, and Default TTL. By setting both a Min TTL and a Max TTL, you can override origin misconfigurations that might cause objects to be cached for longer or shorter periods than you intend. Min TTL, Max TTL, and Default TTL values can be configured uniquely for each of the cache behaviors you define. This allows you to maximize the cache duration for different types of content on your site by setting a lower bound, upper bound, or a default value on the length of time each file can remain in cache.

## Query String Parameters

Query string parameters are often used to return customized content generated by a script running on the origin server. By default, Amazon CloudFront does not forward query string parameters (for example, `"?x=1&y=2"`) to the origin. In addition, the query string portion of the URL is ignored when identifying a unique object in the cache. However, you can optionally configure query strings to be forwarded to the origin servers and be included in the unique identity of the cached object. This feature can be enabled separately for each unique cache behavior that you configure. Query string parameters can thus help you customize your web pages for each viewer while still taking advantage of the performance and scale benefits offered by caching content at Amazon CloudFront edge locations.

## GZIP

You can configure Amazon CloudFront to apply GZIP compression automatically when browsers and other clients request a compressed object with text and other compressible file formats. This means that if you are already using Amazon S3, CloudFront can transparently compress this type of content. For origins outside S3, doing compression at the edge means that you don't need to use resources at your origin to do compression. The resulting smaller size of compressed objects makes downloads faster and reduces your CloudFront data transfer charges. To use the feature, simply specify within your cache behavior settings that you would like CloudFront to compress objects automatically and ensure that your client adds `Accept-Encoding: gzip` in the request header (most modern web browsers do this by default).

## HTTP Cookie Support

Amazon CloudFront supports delivery of dynamic content that is customized or personalized using HTTP cookies. To use this feature, you specify whether you want Amazon CloudFront to forward some or all of your cookies to your custom origin server. You may also specify wildcard characters in the cookie name to forward multiple cookies matching a string format. Amazon CloudFront then considers the forwarded cookie values when identifying a unique object in its cache. This way, your end users get both the benefit of content that is personalized just for them with a cookie and the performance benefits of Amazon CloudFront.

## Forward Headers to Origin

You can use Amazon CloudFront to forward all (or a whitelist of) request headers to your origin server. These headers contain information, such as the device used by your visitors or the country from which they accessed your content. You can configure CloudFront to cache your content based on the values in the headers, so that you can deliver customized content to your viewers. For example, if you are hosting multiple websites on the same web server, you can configure Amazon CloudFront to forward the Host header to your origin. When your origin returns different versions of the same object based on the values in the Host header, Amazon CloudFront will cache the objects separately based on those values.

## Add or Modify Request Headers Forwarded from Amazon CloudFront to Origin

You can configure Amazon CloudFront to add custom headers or override the value of existing request headers when CloudFront forwards requests to your origin. You can use these headers to help validate that requests made to your origin were sent from CloudFront (shared secret) and configure your origin only to allow requests that contain the custom header values that you specify. This feature also helps with setting up Cross-Origin Request Sharing (CORS) for your CloudFront distribution: You can configure CloudFront always to add custom headers to your origin to accommodate viewers that don't automatically include those headers in requests. It also allows you to disable varying on the origin header, which improves your cache hit ratio, yet forward the appropriate headers so that your origin can respond with the CORS header.

## Enforce HTTPS-Only Connection Between Amazon CloudFront and Your Origin Web Server

You can configure Amazon CloudFront to connect to your origin server using HTTPS regardless of whether the viewer made the request by using HTTP or HTTPS.

## Support for TLSv1.1 and TLSv1.2 Between Amazon CloudFront and Your Origin Web Server

Amazon CloudFront supports the TLSv1.1 and TLSv1.2 protocols for HTTPS connections between CloudFront and your custom origin web server (along with SSLv3 and TLSv1.0). You can choose the protocols that you want CloudFront to use when communicating with your origin so that you can, for example, choose not to allow CloudFront to communicate with your origin by using SSLv3, which is less secure than TLS.

## Default Root Object

You can specify a default file (for example, `index.html`) that will be served for requests made for the root of your distribution without an object name specified, for instance, requests made to `http://abc123.cloudfront.net/` alone, without a file name.

## Object Versioning and Cache Invalidation

You have two options to update your files cached at the Amazon CloudFront edge locations. You can use *object versioning* to manage changes to your content. To implement object versioning, you create a unique file name in your origin server for each version of your file and use the file name corresponding to the correct version in your web pages or applications. With this technique, Amazon CloudFront caches the version of the object that you want without needing to wait for an object to expire before you can serve a newer version.

You can also remove copies of an object from all Amazon CloudFront edge locations at any time by calling the *invalidation* API. This feature removes the object from every Amazon CloudFront edge location regardless of the expiration period you set for that object

on your origin server. If you need to remove multiple objects at once, you may send a list of invalidation paths (up to 3,000) in an XML document. Additionally, you can request up to 15 invalidation paths with a wildcard character. The invalidation feature is designed to be used in unexpected circumstances; for example, to correct an encoding error on a video you uploaded or an unanticipated update to your website's CSS file. However, if you know beforehand that your files will change frequently, it is recommended that you use object versioning to manage updates to your files. This technique gives you more control over when your changes take effect, and it also lets you avoid potential charges for invalidating objects.

## Access Logs

You can choose to receive more information about the traffic delivered or streamed by your Amazon CloudFront distribution by enabling access logs. *Access logs* are activity records that show you detailed information about each request made for your content. CloudFront access files are automatically delivered multiple times per hour and the logs in those files will typically be available within an hour of your viewers requesting that object.

## Amazon CloudFront Usage Charts

Amazon *CloudFront Usage Charts* let you track trends in data transfer and requests (both HTTP and HTTPS) for each of your active CloudFront web distributions. These charts show your usage from each CloudFront region at daily or hourly granularity, going back up to 60 days. They also include totals, average, and peak usage during the time interval selected.

## Amazon CloudFront Monitoring and Alarming Using Amazon CloudWatch

You can monitor, alarm, and receive notifications on the operational performance of your Amazon CloudFront distributions using Amazon CloudWatch, giving you more visibility into the overall health of your web application. CloudFront automatically publishes six operational metrics, each at 1-minute granularity, into Amazon CloudWatch. You can then use CloudWatch to set alarms on any abnormal patterns in your CloudFront traffic. These metrics appear in CloudWatch within a few minutes of the viewer's request for each of your Amazon CloudFront web distributions.

## Zone Apex Support

You can use Amazon CloudFront to deliver content from the root domain, or "zone apex" of your website. For example, you can configure both `http://www.example.com` and `http://example.com` to point at the same CloudFront distribution, without the performance penalty or availability risk of managing a redirect service.

## Using Amazon CloudFront with AWS WAF to Protect Your Web Applications

*AWS WAF* is a web application firewall that helps detect and block malicious web requests targeted at your web applications. AWS WAF allows you to create rules based on IP addresses, HTTP headers, and custom URIs. Using these rules, AWS WAF can block, allow, or monitor (count) web requests for your web application.

## HTTP Streaming of On-Demand Media

Amazon CloudFront can be used to deliver your on-demand adaptive bit-rate media content at scale to a global audience. Whether you want to stream your content using Microsoft Smooth Streaming format to Microsoft Silverlight players or stream to iOS devices using HTTP Live Streaming (HLS) format, you can do so using Amazon CloudFront without the need to set up and manage any third-party media servers. Furthermore, there are no additional charges for using this capability beyond Amazon CloudFront's standard data transfer and request fees. Simply encode your media files for the format you want to use and upload it to the origin they plan to use.

**On-demand Smooth Streaming** You can specify in the cache behavior of an Amazon CloudFront web distribution to support Microsoft Smooth Streaming format for that origin.

**On-demand HLS Streaming** Streaming on-demand content using the HLS format is supported in Amazon CloudFront without having to do any additional configurations. You store your content in your origin (for example, Amazon S3). Amazon CloudFront delivers this content at a global scale to a player (such as the iOS player) requesting the HLS segments for playback.

## RTMP Distributions for On-Demand Media Delivery

Amazon CloudFront lets you create *RTMP distributions*, which deliver content to end users in real time—the end users watch the bytes as they are delivered. Amazon CloudFront uses Adobe's Flash Media Server 3.5 to power its RTMP distributions. RTMP distributions use the Real-Time Messaging Protocol (RTMP) and several of its variants, instead of the HTTP or HTTPS protocols used by other Amazon CloudFront distributions.

## Content Delivery

Now that you have configured Amazon CloudFront to deliver your content, the following will happen when users request your objects:

1. A user accesses your website or application and requests one or more objects.
2. DNS routes the request to the Amazon CloudFront edge location that can best serve the user's request, typically the nearest Amazon CloudFront edge location in terms of latency, and routes the request to that edge location.
3. In the edge location, Amazon CloudFront checks its cache for the requested files. If the files are in the cache, Amazon CloudFront returns them to the user. If the files are not in the cache, CloudFront does the following:
    a. Amazon CloudFront compares the request with the specifications in your distribution and forwards the request for the files to the applicable origin server for the corresponding file type.
    b. The origin servers send the files back to the Amazon CloudFront edge location.
    c. As soon as the first byte arrives from the origin, Amazon CloudFront begins to forward the files to the user.
    d. Amazon CloudFront also adds the files to the cache in the edge location for the next time someone requests those files.

As mentioned earlier, you can configure headers to indicate how long you want the files to stay in the cache in Amazon CloudFront edge locations. By default, each object stays in an edge location for 24 hours before it expires. The minimum expiration time is 0 seconds; there isn't a maximum expiration time limit.

The Amazon CloudFront console includes a variety of reports:

**Amazon CloudFront cache statistics reports** These reports use the Amazon CloudFront console to display a graphical representation of statistics related to CloudFront edge locations. Data for these statistics are drawn from the same source as CloudFront access logs. You can display charts for a specified date range in the last 60 days, with data points every hour or every day.

**Amazon CloudFront popular objects reports** These reports are available via the Amazon CloudFront console. You can display a list of the 50 most popular objects for a distribution during a specified date range in the previous 60 days.

**Amazon CloudFront top referrers reports** These reports provide a list of the 25 domains of the websites that originated the most HTTP and HTTPS requests for objects that CloudFront is distributing for a specified distribution. These top referrers can be search engines, other websites that link directly to your objects, or your own website.

**Amazon CloudFront usage reports** These are reports which are more detailed than the billing report but less detailed than CloudFront access logs. The usage report provides aggregate usage data by hour, day, or month, and it lists operations by region and usage type.

**Amazon CloudFront viewers reports** These reports show devices, browsers, and operating systems' versions used to access your content, and also from what locations they are accessing your content.

## Amazon CloudFront Management

You can configure Amazon CloudFront using the AWS Management Console, the Amazon CloudFront console, the AWS CLI, or various SDKs available for Amazon CloudFront.

Amazon CloudFront metrics are also accessible in the Amazon CloudWatch Console.

## Amazon CloudFront Security

You can control user access to your private content in two ways:

1. Restrict access to objects in the Amazon CloudFront edge cache using either signed URLs or signed cookies.
2. Restrict access to objects in your Amazon S3 bucket so that users can access it through Amazon CloudFront, thus direct access to the S3 bucket. See Amazon S3 bucket policies in Chapter 6, "Storage Systems," for more details.


Understanding how Amazon CloudFront works and how it can deliver content to your end users is important. Knowing how to manage content both at the origin and at the edge is crucial.

# Summary

We covered a lot of material in this chapter. While the exam questions may not go into as great a depth in terms of detail, understanding this material will assist you in providing the best answers to the questions on the exam.

In this chapter, we discussed the following:

- Amazon VPC as a logically-isolated section of the AWS Cloud
- AWS Direct Connect and how it allows you to establish a dedicated network connection from your premises to AWS
- The two types of Elastic Load Balancers and their features (Application and Classic Load Balancers)
- How Elastic Load Balancers automatically distribute incoming application traffic across multiple Amazon Elastic Compute Cloud (Amazon EC2) instances within an AWS Region
- Virtual Private Network (VPN) connections and how you can connect Amazon VPC to remote networks to make your AWS infrastructure highly available
- Internet gateways, which are used to connect your public subnets to the Internet
- NAT gateways, which are used to provide connectivity to the Internet from your private instances
- Elastic Network interfaces, which are used to multi-hone an Amazon EC2 instance and can be re-assigned to another Amazon EC2 instance
- Elastic IP addresses (EIP), which are public IPv4 addresses that can be assigned and reassigned to Amazon EC2 instances. IPv6 is not (currently) supported with EIP.
- You learned that Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. We discussed the various routing types: Simple, Weighted Round Robin (WRR), Latency Based Routing (LBR), geolocation, and Failover routing.
- You learned that Amazon CloudFront is a global Content Delivery Network (CDN) service that accelerates delivery of your websites, Application Programming Interfaces (APIs), video content, or other web assets.

# Resources to Review

**AWS YouTube channel:** `https://www.youtube.com/user/AmazonWebServices`

**AWS What's new:** `https://aws.amazon.com/new`

**Jeff Barr's blog:** `https://aws.amazon.com/blogs/aws/`

**Amazon VPC documentation:** `https://aws.amazon.com/documentation/vpc`

**Amazon VPC peering guide:**
`http://docs.aws.amazon.com/AmazonVPC/latest/PeeringGuide/Welcome.html`

**VPN options:**
`http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpn-connections.html`

**NAT gateway fundamentals on AWS:**
`http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-nat-gateway.html`

**Amazon CloudFront documentation:**
`https://aws.amazon.com/documentation/cloudfront/`

**Amazon Route 53 documentation:** `https://aws.amazon.com/documentation/route53`

**Elastic Load Balancing documentation:**
`https://aws.amazon.com/documentation/elastic-load-balancing/`

**AWS Direct Connect and VPN deep dive:**
`https://www.youtube.com/watch?v=Qep11X1r1QA`

**Amazon CloudFront best practices:** `https://www.youtube.com/watch?v=fgbJJ412qRE`

# Exam Essentials

**Understand what a VPC is.** Know how to set up a VPC, and what are the minimum and maximum size of both a VPC and subnets.

**Understand the purpose and use of route tables, network ACLs, and security groups.** Know how to use each for controlling access and providing security.

**Know what are the default values for route tables, network ACLs, and security groups.** Know where those default values come from, how to modify them, and why you would modify them.

**Understand the difference between a private and public subnet.** Public subnets allow traffic to the Internet; private subnets do not. Know how to use Amazon EC2 instances in private subnets to have access the Internet.

**Understand the role and function of the various ways to connect the VPC with outside resources.** This includes Internet gateway, VPN gateway, Amazon S3 endpoint, VPC peering, NAT instances, and NAT gateways. Understand how to configure these services.

**Understand what is an Elastic IP (EIP).** Elastic supports public IPv4 addresses (as of this publication). Understand the difference between EIP and an ENI.

**Understand what is an Elastic Network Interface (ENI).** Elastic Network Interfaces can be assigned and reassigned to an Amazon EC2 instance. Understand why this is important.

**Know what services operate within a VPC and what services operate outside a VPC.** Amazon EC2 lives within a VPC. Services such as Amazon S3 live outside the VPC. Know the various ways to access these services.

**Know what AWS Direct Connect is.** Understand why it is used and the basic steps for setting it up. (Remember the seven steps listed in the AWS Direct Connect section of this chapter.)

**Understand the concept of VIFs.** Understand what a VIF is and the difference between a public and private VIF. Why would you use one versus the other? Understanding these concepts will be very helpful on the exam.

**Understand the options for Elastic Load Balancing (Classic Load Balancer vs. Application Load Balancer).** Know how each type of load balancer operates, why you would choose one over the other, and how to configure each.

**Understand how health checks work in each type of load balancer.** Classic Load Balancers and Application Load Balancers have different health check options. Know what they are!

**Understand how listeners work.** Understand rules, priorities, and conditions and how they interact.

**Know how Amazon CloudWatch, AWS CloudTrail, and access logs work.** Know what type of information each one provides.

**Understand the role of security groups with load balancers.** Be able to configure a security group and know how rules are applied.

**Understand the various options for establishing an IPsec VPN tunnel from an Amazon VPC to a customer location.** Know the operational and security implications of these options.

**Know how Amazon Route 53 works as a DNS provider.** Understand how it can be used for both public and private hosted zones.

**Know what the different routing options are for Amazon Route 53.** Understand how to configure the various routing options and how they work.

**Know what an Amazon Route 53 routing policy is.** Understand how it is applied in Amazon Route 53.

**Understand what record types Amazon Route 53 supports and how they work.** Know both standard and non-standard record sets.

**Know the tools for managing and monitoring Amazon Route 53.** Understand how Amazon CloudWatch and AWS CloudTrail work with Amazon Route 53. Go deep in understanding how all of the services in this chapter are monitored.

**Know the purpose of Amazon CloudFront and how it works.** Know what a distribution is and what an origin is. Know what types of files Amazon CloudFront can handle.

**Know the steps to implement Amazon CloudFront.** Remember there are three steps to do this.

**Know the various methods for securing content in Amazon CloudFront.** Know how to secure your content at the edge and at the origin.

### Production and Archive

By storing the content (text and media) in Amazon S3 (or Amazon S3-RRS), the CMS is already positioned with the most cost-effective, scalable option to deliver the content.

At some point, the content will be determined to be out of date. In the case of a news website, it may take months before consumers stop searching for or clicking on old stories, but eventually all content will reach a point of diminishing results. When the content reaches that point, automated processes can transition the content to Amazon S3-IA. Why not Amazon Glacier? Because in the case of a news archive, even if consumers don't care about the old story for years, once someone is doing research and wants to open that file (or video), odds are they don't want to wait five hours for the content to be available. By using Amazon S3-IA, the archive is still cost effective but immediately available and searchable.

# Additional Storage Solutions

Amazon S3, Amazon EBS, Amazon EFS, and Amazon Glacier represent the core of the AWS storage solutions. When studying to be successful on the exam, you must be very comfortable with these services.

AWS does provide a number of other services that may be considered part of the storage solutions discussion. You should be familiar with them, but they will not likely be a significant portion of the AWS Certified SysOps Administrator – Associate exam.

## Amazon CloudFront

Global content distribution can be delivered using *Amazon CloudFront*. This Content Delivery Network (CDN) can help accelerate the delivery of your content by caching copies close to consumers.

Content is stored in origin locations that can be on AWS in an Amazon S3 bucket or served from an Amazon EC2 instance. Amazon CloudFront can even cache content stored on-premises.

Amazon CloudFront is a good choice for distribution of frequently accessed static content that benefits from edge delivery, like popular website images, videos, media files, or software downloads. For on-demand media files, you can also choose to stream your content using Real-Time Messaging Protocol (RTMP) delivery. Amazon CloudFront also supports delivery of live media over HTTP.

Lastly, Amazon CloudFront has a geo-restriction feature that lets you specify a list of countries in which your users can access your content. Alternatively, you can specify the countries in which your users cannot access your content. In both cases, Amazon CloudFront responds to a request from a viewer in a restricted country with an HTTP status code 403 (Forbidden).

# AWS Storage Gateway

*AWS Storage Gateway* allows existing on-premises storage solutions to be extended into AWS. By installing a virtual appliance in the local datacenter, storage can be duplicated or extended into the AWS Region.

For the exam, you will likely see at least a couple of questions that involve AWS Storage Gateway. The important things to understand are the different options available when configuring the appliance.

## AWS Storage Gateway Options

The first option is called the *file interface*. This enables you to use Amazon S3 with your on-premises workflows by accessing the Amazon S3 bucket through a Network File System (NFS) mount point. This allows customers to migrate their files into Amazon S3 through object-based workloads while maintaining their on-premises investments.

The second option is the *volume interface*. In this case, you are presented with an iSCSI block storage endpoint. Data is accessed on the local volumes, which is then stored on AWS in the form of Amazon EBS snapshots.

There are two configuration modes using the volume interface. The *cached mode* is designed to extend your local storage. All content is saved in AWS, and only the frequently accessed data is cached locally. This is an excellent operational solution for on-premises storage solutions that are exceeding their local capacity limits. The other mode is the *stored mode*, which is a one-to-one backup of all local content on AWS. This is a primary solution when durable and inexpensive off-site backups are needed. This becomes an excellent start to a hybrid disaster recovery plan.

The third option for AWS Storage Gateway is the *tape interface*. In this configuration, you connect to your existing backup application using an iSCSI Virtual Tape Library (VTL). These virtual tapes are then asynchronously stored in Amazon S3 or Amazon Glacier if less expensive, longer-term storage is needed.

# AWS Snowball

For massive data transfers, ground transport can be faster than the Internet. Being able simply to ship large hard drives will get the data into AWS faster (and often cheaper) than public Internet speeds can accommodate. For example, suppose you had 100 TB of data to transfer to AWS. Even over a 100 Mbps data transfer line, it would take 120 days to complete the data transfer. Physically shipping the data is a far faster option.

For those data transfer cases, AWS provides *AWS Snowball*, a physically hardened, secure appliance that ships directly to your on-premises location. Each device weighs around 50 pounds and stores 80 TB (as of this writing). Using multiple AWS Snowball appliances in parallel can provide an easy to implement a migration solution for datasets that are multiple petabytes in size.

AWS Snowball is secured through tamper-resistant seals and a built-in Trusted Platform Module (TPM) that uses a dedicated processor designed to detect any unauthorized modifications to the hardware, firmware, or software.

Data is automatically encrypted when using AWS Snowball with encryption keys managed through AWS KMS. The actual encryption keys are never stored on the appliance.

### AWS Snowball with AWS Greengrass

As AWS Snowball has increased in popularity, AWS has added additional functionality, including the ability to access APIs run on the device by leveraging *AWS Greengrass* technology. Those APIs allow the AWS Snowball appliance to act as if it was an Amazon S3 endpoint itself, even when disconnected from the Internet.

### AWS Snowmobile

Some companies have dataset transfer needs that range in the exabyte scale. For those unique transfers, AWS can deploy 45-foot-long shipping containers called *AWS Snowmobiles*. Pulled by tractor trailers, each container can transfer up to 100 PB of data.

For security, AWS Snowmobile uses dedicated security personnel, GPS tracking, alarm monitoring, 24/7 video surveillance, AWS KMS encryption, and an optional escort security vehicle while in transit. For more information on AWS KMS, refer to Chapter 3.

Like AWS Snowball, AWS Snowmobile can deliver multiple containers to a single location, providing exabytes of capacity where needed.

## Summary

There is no one-size-fits-all solution when it comes to storage. As you prepare for the exam, you need to dive deep into the core storage solutions: the block storage options of Amazon EBS and Amazon EFS and the object storage solutions of Amazon S3 and Amazon Glacier. Make sure that you are comfortable with their use cases and when to deploy each option.

## Resources to Review

**Amazon EBS documentation:** `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html`

**Amazon EFS main page:** `https://aws.amazon.com/efs/`

**Amazon EFS documentation:** `https://aws.amazon.com/documentation/efs/`

**Amazon S3 main page:** `http://aws.amazon.com/s3/`

**Amazon S3 documentation:** `http://aws.amazon.com/documentation/s3`

**"IAM Policies and Bucket Policies and ACLs! Oh, My! (Controlling Access to Amazon S3 Resources)" blog post:** `https://aws.amazon.com/blogs/security/iam-policies-and-bucket-policies-and-acls-oh-my-controlling-access-to-s3-resources/`

**Configuring Static Website Hosting section of the Amazon S3 user guide:** `http://docs.aws.amazon.com/AmazonS3/latest/user-guide/static-website-hosting.html`

**Amazon Glacier main page:** `https://aws.amazon.com/glacier/`

**Amazon Glacier documentation:** `https://aws.amazon.com/documentation/glacier/`

**Amazon CloudFront main page:** `https://aws.amazon.com/cloudfront/`

**Amazon CloudFront documentation:** `https://aws.amazon.com/documentation/cloudfront/`

**AWS Storage Gateway main page:** `https://aws.amazon.com/storagegateway/`

**AWS Storage Gateway documentation:**

`https://aws.amazon.com/documentation/storage-gateway/`

**"File Interface to Storage Gateway" blog post:** `https://aws.amazon.com/blogs/aws/category/aws-storage-gateway/`

**AWS Snowball main page:** `https://aws.amazon.com/snowball/`

**AWS Snowball documentation:** `https://aws.amazon.com/documentation/snowball/`

**What's new at AWS:** `https://aws.amazon.com/new`

**AWS Simple Monthly Calculator:** `http://calculator.s3.amazonaws.com/index.html`

# Exam Essentials

**Understand block storage vs. object storage.** The difference between block storage and object storage is the fundamental unit of storage. With block storage, each file being saved to the drive is broken down into "blocks" of a specific size. With object storage, each file is saved as a single object regardless of size.

**Understand when to use Amazon S3 and when to use Amazon EBS or Amazon EFS.** This is an architectural decision based on the content type and the rate of change. Amazon S3 can hold any type of data; however, Amazon S3 would not be a good choice for a database or any rapidly changing data types. Remember the case for eventual consistency.

**Understand the lifecycle of Amazon EBS volumes.** Amazon EBS volumes can be provisioned at the launch of an Amazon EC2 instance or created and attached at a later time. Amazon EBS volumes can persist through the lifecycle of an Amazon EC2 instance, provided the Deleted on Termination flag is set to false. The AWS best practice for an unused Amazon EBS volume is to snapshot it and delete it. The Amazon EBS volume can be created from the snapshot if needed. Remember that with Amazon EBS, you pay for what you provision, as opposed to paying for what you use with Amazon S3.

**Understand different types of Amazon EBS volumes.** Understand bursting. Amazon EBS volumes are tied to the Availability Zone.

**Understand how Amazon EBS snapshots work.** Snapshots are stored in Amazon S3; however, you do not access them via the Amazon S3 Console. They are accessed via the Amazon EC2 console under Snapshots.

**Understand instance store storage.** Depending on the Amazon EC2 instance family, you have access to the local storage. This storage is called the instance store. Snapshots cannot be taken of the instance store. Instance store storage is ephemeral and doesn't survive a restart or termination of the Amazon EC2 instance. For more information, refer to Chapter 4.

**Have a detailed understanding of how Amazon S3 lifecycle polices work.** Lifecycle configuration enables you to specify the lifecycle management of objects in a bucket. The configuration is a set of one or more rules, where each rule defines an action for Amazon S3 to apply to a group of objects. Know the two types: Transition actions and expiration actions.

**Understand Amazon S3 versioning.** When Amazon S3 versioning is turned on, it cannot be turned off, only paused. Understand that when versioning is turned on, items deleted are assigned a delete marker and are unable to be accessed. The deleted objects are still in Amazon S3 and you still pay for them.

**Understand how to interface with Amazon Glacier.** When objects are moved from Amazon S3 to Amazon Glacier, they can only be accessed from the Amazon S3 APIs (for example, in the case of an object that has been moved to Amazon Glacier as the result of a lifecycle policy).

**Understand Amazon Glacier vaults.** Amazon Glacier stores objects as archives and stores the archives in vaults. You can have (as of this writing) 1,000 vaults per account per region. Vaults are immutable; when retrieving an object, it is not removed from Amazon Glacier but is copied to Amazon S3.

**Know how MFA Delete works with Amazon S3.** You can optionally add another layer of security by configuring a bucket to enable MFA Delete, which requires additional authentication for changing the versioning state of your bucket and for permanently deleting an object version. MFA Delete requires two forms of authentication together: your security credentials and the combination of a valid serial number, a space, and the six-digit code displayed on an approved authentication device. MFA Delete thus provides added security in the event, for example, that your security credentials are compromised.

**Know how to control access to Amazon S3 resources.** IAM policies specify what actions are allowed or denied on what AWS resources. Amazon S3 bucket policies, on the other hand, are attached only to Amazon S3 buckets. Amazon S3 bucket policies specify what actions are allowed or denied for which principles on the bucket to which the bucket policy is attached. Amazon S3 bucket policies are a type of ACL.

**Understand the features of Amazon CloudFront.** Amazon CloudFront is a CDN that can help accelerate the delivery of content by caching copies close to consumers. Content is stored in origin locations, which can be on AWS in an Amazon S3 bucket or served from an Amazon EC2 instance. Amazon CloudFront can even cache content stored on-premises.

**Understand media that can be delivered from Amazon CloudFront.** This media includes static website CSS style sheets, HTML pages, images, videos, media files, or software downloads. For on-demand media files, you can also choose to stream your content using RTMP delivery. Amazon CloudFront also supports delivery of live media over HTTP.

**Understand Amazon CloudFront's geo-restriction feature.** This feature lets you specify a list of countries in which your users can access your content. Alternatively, you can specify the countries in which your users cannot access your content.

**Understand AWS Storage Gateway's interfaces and modes of operation.** The first option is called the file interface. This enables you to use Amazon S3 with your on-premises workflows by accessing the Amazon S3 bucket through an NFS mount point. The second option is the volume interface. In this case, you are presented with an iSCSI block storage endpoint. The third option for AWS Storage Gateway is the tape interface. In this configuration, you connect to your existing backup application using an iSCSI VTL. The file volume interface operates in one of two modes: cached mode and stored mode. Understand the differences.

**Test Taking Tip**

Multiple choice questions that ask you to choose two or three true answers require that ALL of your answers be correct. There is no partial credit for getting a fraction correct. Pay extra attention to those questions when doing your review.

# Exercises

By now you should have set up an account in AWS. If you haven't, now would be the time to do so. It is important to note that these exercises are in your AWS account and thus are not free.

Use the Free Tier when launching resources. The AWS Free Tier applies to participating services across the following AWS Regions: US East (Northern Virginia), US West (Oregon), US West (Northern California), Canada (Central), EU (London), EU (Ireland), EU (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Tokyo), Asia Pacific (Sydney), and South America (Sao Paulo). For more information, see `https://aws.amazon.com/s/dm/optimization/server-side-test/free-tier/free_np/`.

If you have not yet installed the AWS Command Line utilities, refer to Chapter 2, "Working with AWS Cloud Services," Exercise 2.1 (Linux) or Exercise 2.2 (Windows).

The reference for the AWS CLI can be found at `http://docs.aws.amazon.com/cli/latest/reference/`.

Getting to know storage systems can be simple when using your own account. Follow the steps in previous chapters to log in to your account. As always, remember that although AWS is a very cost-effective solution, you will be responsible for all charges incurred. If you don't want to keep the items permanently, remember to delete them after your practice session.

Remember, the goal of this book isn't just to prepare you to pass the AWS Certified SysOps Administrator – Associate exam. It should also serve as a reference companion in your day-to-day duties as an AWS Certified SysOps Administrator.

JON BONSO AND KENNETH SAMONTE

# AWS CERTIFIED DEVOPS ENGINEER PROFESSIONAL

Tutorials Dojo
Study Guide and Cheat Sheets

## Amazon Route 53

- A highly available and scalable Domain Name System (DNS) web service used for domain registration, DNS routing, and health checking.

### Routing Internet Traffic to your Website or Web Application

- Use the Route 53 console to register a domain name and configure Route 53 to route internet traffic to your website or web application.
- After you register your domain name, Route 53 automatically creates a **public hosted zone** that has the same name as the domain.
- To route traffic to your resources, you create **records**, also known as *resource record sets*, in your hosted zone.
- You can create special Route 53 records, called **alias records**, that route traffic to S3 buckets, CloudFront distributions, and other AWS resources.
- Each record includes information about how you want to route traffic for your domain, such as:
    - Name - name of the record corresponds with the domain name or subdomain name that you want Route 53 to route traffic for.
    - Type - determines the type of resource that you want traffic to be routed to.
    - Value

### Route 53 Health Checks

    - Create a health check and specify values that define how you want the health check to work, such as:
        - The IP address or domain name of the endpoint that you want Route 53 to monitor.
        - The protocol that you want Route 53 to use to perform the check: HTTP, HTTPS, or TCP.
        - The **request interval** you want Route 53 to send a request to the endpoint.
        - How many consecutive times the endpoint must fail to respond to requests before Route 53 considers it unhealthy. This is the **failure threshold**.
    - You can configure a health check to check the health of one or more other health checks.
    - You can configure a health check to check the status of a CloudWatch alarm so that you can be notified on the basis of a broad range of criteria.

### Know the following Concepts

- Domain Registration Concepts - domain name, domain registrar, domain registry, domain reseller, top-level domain
- DNS Concepts
    - **Alias record** - a type of record that you can create to route traffic to AWS resources.

  - **Hosted zone** - a container for records, which includes information about how to route traffic for a domain and all of its subdomains.
  - **Name servers** - servers in the DNS that help to translate domain names into the IP addresses that computers use to communicate with one another.
  - **Record** (DNS record) - an object in a hosted zone that you use to define how you want to route traffic for the domain or a subdomain.
  - **Routing policy -** policy on how to redirect users based on configured routing policy
  - **Subdomain** - name below the zone apex. Example: portal.tutorialsdojo.com
  - Time to live (TTL) - time that the DNS record is cached by querying servers.
- Health Checking Concepts
  - **DNS failover** - a method for routing traffic away from unhealthy resources and to healthy resources.
  - Endpoint - the URL or endpoint on which the health check will be performed.
  - Health check - the metric on which to determine if an endpoint is healthy or not.

**Records**

- Create records in a hosted zone. Records define where you want to route traffic for each domain name or subdomain name. The name of each record in a hosted zone must end with the name of the hosted zone.
- Alias Records
  - Route 53 **alias records** provide a Route 53–specific extension to DNS functionality. Alias records let you route traffic to selected AWS resources. They also let you route traffic from one record in a hosted zone to another record.
  - You can create an alias record at the top node of a DNS namespace, also known as the zone apex.
- CNAME Record
  - You cannot create an alias record at the top node (zone apex) of a DNS namespace using a CNAME record.

## AWS Elastic Load Balancing (ELB)

- Distributes incoming application or network traffic across multiple targets, such as **EC2 instances**, **containers (ECS)**, **Lambda functions,** and **IP addresses**, in multiple Availability Zones.
- When you create a load balancer, you must specify one public subnet from at least two Availability Zones. You can specify only one public subnet per Availability Zone.

### General features

- Accepts incoming traffic from clients and routes requests to its registered targets.
- Monitors the health of its registered targets and routes traffic only to healthy targets.
- **Cross Zone Load Balancing** - when enabled, each load balancer node distributes traffic across the registered targets in all enabled AZs.

### Three Types of Load Balancers

- **Application Load Balancer**
- **Network Load Balancer**
- **Classic load Balancer**
- **Slow Start Mode** gives targets time to warm up before the load balancer sends them a full share of requests.
- **Sticky sessions** route requests to the same target in a target group. You enable sticky sessions at the target group level. You can also set the duration for the stickiness of the load balancer-generated cookie, in seconds. Useful if you have stateful applications.
- **Health checks** verify the status of your targets. The statuses for a registered target are:

## AWS Security & Identity Services

### Amazon GuardDuty

- An intelligent threat detection service. It analyzes billions of events across your AWS accounts from AWS CloudTrail (AWS user and API activity in your accounts), Amazon VPC Flow Logs (network traffic data), and DNS Logs (name query patterns).
- GuardDuty is a regional service.
- Threat detection categories
    - **Reconnaissance** -- Activity suggesting reconnaissance by an attacker, such as unusual API activity, intra-VPC port scanning, unusual patterns of failed login requests, or unblocked port probing from a known bad IP.
    - **Instance compromise** -- Activity indicating an instance compromise, such as cryptocurrency mining, backdoor command and control activity, malware using domain generation algorithms, outbound denial of service activity, unusually high volume of network traffic, unusual network protocols, outbound instance communication with a known malicious IP, temporary Amazon EC2 credentials used by an external IP address, and data exfiltration using DNS.
    - **Account compromise** -- Common patterns indicative of account compromise include API calls from an unusual geolocation or anonymizing proxy, attempts to disable AWS CloudTrail logging, changes that weaken the account password policy, unusual instance or infrastructure launches, infrastructure deployments in an unusual region, and API calls from known malicious IP addresses.
- Amazon GuardDuty provides three severity levels (Low, Medium, and High) to allow you to prioritize response to potential threats.
- CloudTrail Event Source
    - Currently, GuardDuty only analyzes CloudTrail management events. (Read about types of CloudTrail trails for more information)
    - GuardDuty processes all CloudTrail events that come into a region, including global events that CloudTrail sends to all regions, such as AWS IAM, AWS STS, Amazon CloudFront, and Route 53.
- VPC Flow Logs Event Source
    - VPC Flow Logs capture information about the IP traffic going to and from Amazon EC2 network interfaces in your VPC.
- DNS Logs Event Source
    - If you use AWS DNS resolvers for your EC2 instances (the default setting), then GuardDuty can access and process your request and response DNS logs through the internal AWS DNS resolvers. Using other DNS resolvers will not provide GuardDuty access to its DNS logs.
- GuardDuty vs Macie
    - Amazon GuardDuty provides broad protection of your AWS accounts, workloads, and data by helping to identify threats such as attacker reconnaissance, instance compromise, and account compromise. Amazon Macie helps you protect your data in Amazon S3 by helping you classify