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

Route 53

Route 53 is Amazon's global Domain Name System (DNS) service. The primary purpose of DNS is to translate human-readable domain names (such as `example.com`) into IP addresses. Here's a simplified version of how it works: when you enter the domain name `example.com` into your web browser, your computer sends out a query to its configured DNS server asking for the IP address of that domain. The DNS server then sends the query to the domain's authoritative DNS server—the one that's in charge of the `example.com` domain name. The authoritative DNS server responds with the IP address for `example.com`. This process of translating a domain name to an IP address is called *name resolution*.

Resource Records

Name resolution goes beyond just mapping domain names to IP addresses. DNS can store mappings for different types of data, including IPv6 addresses, mail servers, and even arbitrary text. When you send an email to someone, DNS provides the lookup mechanism to ensure it gets routed to the correct mail server for that domain.

For DNS to work, someone must first define some resource records for a domain. A resource record consists of several fields, but the most important are the name, type, and value. Refer to Table 10.1 for some example resource records.

TABLE 10.1 Resource Records for the benpiper.com Domain

Name	Type	Value
`benpiper.com`	A - IPv4 address	93.184.216.34
`www.benpiper.com`	A - IPv4 address	93.184.216.34
`benpiper.com`	MX - Mail exchange	10 in1-smtp.messagingengine.com

Domain Name Registration

A public domain name is one that anyone on the internet can resolve. To ensure that no two entities try to use the same domain name, anyone who wants to have a public domain name must register it with a domain registrar. When you register a domain name, you must do so under a top-level domain (TLD) such as `.com`, `.net`, or `.org`. For example, you might register the name `example.com` or `example.org`. Route 53 is a domain registrar for hundreds of TLDs.

Registering a domain gives you control of it for the duration of the lease, which can be in yearly increments between 1 year and 10 years. Regardless of how long you initially register a domain for, you can renew it in yearly increments indefinitely. If you have an

existing domain name with another registrar, you can transfer it to Route 53. Transferring a domain entails extending the registration by at least one year.

It's important to understand that domain name registration and DNS hosting are two different functions. Registering a domain name gives you control over it for the duration of the lease, including the right to specify the service you want to provide DNS hosting for the domain. This means the domain registrar and DNS hosting provider don't have to be the same company, but they often are. Route 53 is both a registrar and a DNS hosting provider.

Hosted Zones

To have Route 53 host the DNS for a public domain name, you create a public hosted zone and specify the domain name. You can then define the resource records for that domain. If you use Route 53 to register a domain name, it automatically takes care of creating a public hosted zone for the domain.

Route 53 can also provide name resolution for private domain names. A private domain name is one used on a network other than the internet. Route 53 private hosted zones provide DNS resolution for a single domain name within multiple VPCs. This is useful for assigning user-friendly domain names to VPC resources such as EC2 instances or application load balancers. For example, instead of hardcoding a database server's IP in an application, you can define a record in a private hosted zone with the name db.`example.com` that points to the database server's IP address. Because private domain names aren't accessible from the internet, there are no registrars, so you can pick any domain name you want. Name resolution for private hosted zones is not available outside of the VPC you select.

Routing Policies

In some cases, you just need a domain name to resolve to a particular IP address. But there are other times when you want the value of a resource record to change dynamically to work around failures or ensure users get pointed to the least busy server. Route 53 lets you accomplish this with a variety of routing policies.

Simple The Simple routing policy is the default for new resource records. It simply lets you map a domain name to a single static value, such as an IP address. It doesn't check whether the resource the record points to is available.

Weighted A Weighted policy distributes traffic across multiple resources according to a ratio. For example, when introducing a new web server, you may want to route only 10 percent of the traffic to the new server while evenly distributing the load across the rest.

Latency A Latency policy sends users to resources in the AWS Region that's closest to them. This is useful if, for instance, you want to send European users to the eu-west-1 region while sending users in the United States to the us-east-1 region.

Failover A Failover policy lets you route traffic to a primary resource unless it's unavailable. In that case, traffic will be redirected to a secondary resource.

Geolocation A Geolocation policy lets you route users based on their specific continent, country, or state.

Multivalue Answer A Multivalue Answer policy allows you to evenly distribute traffic across multiple resources. Unlike Weighted policies that return a single record, a Multivalue Answer policy returns all records, sorted in a random order.

Health Checks

All routing policies with the exception of Simple can use health checks to determine whether they should route users to a given resource. A health check can check one of three things: an endpoint, a CloudWatch alarm, or another health check. All health checks occur every 10 seconds or 30 seconds.

Endpoint Endpoint health checks work by connecting to the endpoint you want to monitor via HTTP, HTTPS, or TCP. Route 53 has health checkers in several AWS Regions, and you can choose which health checkers a health check uses. This lets you ensure that an endpoint is reachable from various locations around the world.

CloudWatch alarm A Route 53 health check can monitor the status of a CloudWatch alarm. This is useful if you want to consider a resource unhealthy if it's experiencing high latency or is servicing a high number of connections.

Calculated This type of health check monitors the status of other health checks. For example, if you want to consider the status of both an Endpoint health check and a CloudWatch alarm health check, you can create a Calculated health check to take both into account.

Traffic Flow and Traffic Policies

If you require complex routing scenarios for a public hosted zone, creating multiple resource records with a variety of different routing policies can become an administrative nightmare. As an alternative to manually engineering routing policies, you can use the Route 53 Traffic Flow visual editor to create a diagram to represent the desired routing.

The diagram you create represents a traffic policy that you can save and associate with a domain name by creating a policy record. Route 53 doesn't create the individual resource records but instead hides the routing behind the single policy record. The cost is currently $50 USD per month per policy record.

You can use the same routing policies that are available with normal resource records: Simple, Weighted, Latency, Failover, Geolocation, and Multivalue Answer. But in addition, Traffic Flow offers another routing policy that's not otherwise available: Geoproximity. The Geoproximity routing policy lets you direct users to a resource based on how close they are to a geographic location. This differs from the Geolocation routing policy that routes based on the user's specific continent, country, or state.

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.

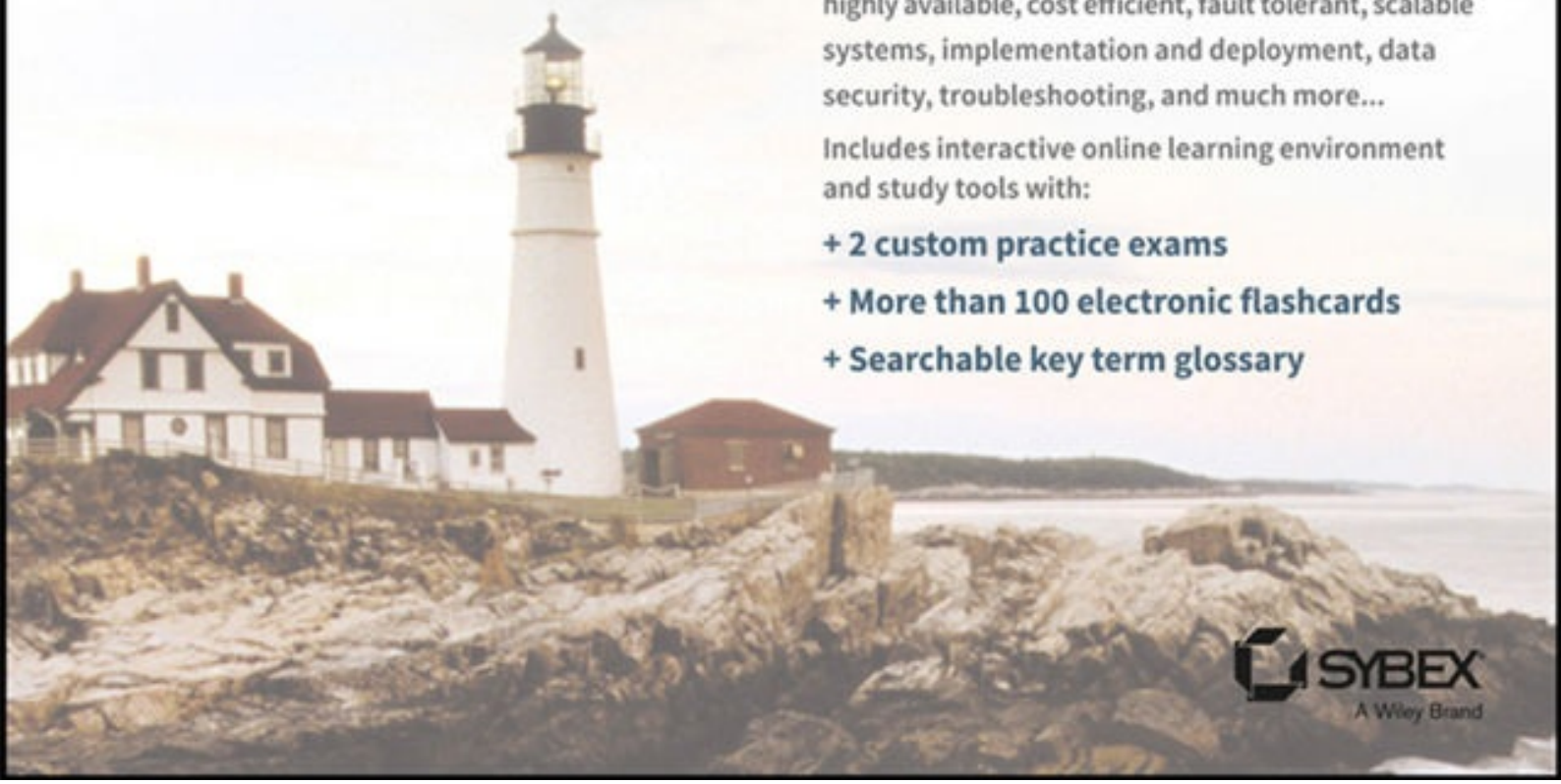
amazon web services

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

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

Domain Name System (DNS)

The *Domain Name System (DNS)* is sometimes a difficult concept to understand because it is so ubiquitously used in making the Internet work. Before we get into the details, let's start with a simple analogy. The *Internet Protocol (IP)* address of your website is like your phone number—it could change if you move to a new area (at least your land line could change). DNS is like the phonebook. If someone wants to call you at your new house or location, they might look you up by name in the phonebook. If their phonebook hasn't been updated since you moved, however, they might call your old house. When a visitor wants to access your website, their computer takes the domain name typed in (`www.amazon .com`, for example) and looks up the IP address for that domain using DNS.

More specifically, DNS is a globally-distributed service that is foundational to the way people use the Internet. DNS uses a hierarchical name structure, and different levels in the hierarchy are each separated with a dot (`.`). Consider the domain names `www.amazon .com` and `aws.amazon.com`. In both these examples, `com` is the Top-Level Domain (TLD) and `amazon` is the Second-Level Domain (SLD). There can be any number of lower levels (for example, `www` and `aws`) below the SLD.

Computers use the DNS hierarchy to translate human readable names (for example, `www.amazon.com`) into the IP addresses (for example, `192.0.2.1`) that computers use to connect to one another. Every time you use a domain name, a DNS service must translate the name into the corresponding IP address. In summary, if you've used the Internet, you've used DNS.

Amazon Route 53 is an *authoritative DNS system*. An authoritative DNS system provides an update mechanism that developers use to manage their public DNS names. It then answers DNS queries, translating domain names into IP addresses so that computers can communicate with each other.

This chapter is intended to provide you with a baseline understanding of DNS and the Amazon Route 53 service that is designed to help users find your website or application over the Internet.

Domain Name System (DNS) Concepts

This section of the chapter defines DNS terms, describes how DNS works, and explains commonly used *record types*.

Top-Level Domains (TLDs)

A *Top-Level Domain (TLD)* is the most general part of the domain. The TLD is the farthest portion to the right (as separated by a dot). Common TLDs are `.com`, `.net`, `.org`, `.gov`, `.edu`, and `.io`.

TLDs are at the top of the hierarchy in terms of domain names. Certain parties are given management control over TLDs by the Internet Corporation for Assigned Names and Numbers (ICANN). These parties can then distribute domain names under the TLD, usually through a domain registrar. These domains are registered with the Network Information Center (InterNIC), a service of ICANN, which enforces the uniqueness of domain names

across the Internet. Each domain name becomes registered in a central database, known as the WhoIS database.

Domain Names

A *domain name* is the human-friendly name that we are used to associating with an Internet resource. For instance, `amazon.com` is a domain name. Some people will say that the `amazon` portion is the domain, but we can generally refer to the combined form as the domain name.

The URL `aws.amazon.com` is associated with the servers owned by AWS. The DNS allows users to reach the AWS servers when they type `aws.amazon.com` into their browsers.

IP Addresses

An *IP address* is a network addressable location. Each IP address must be unique within its network. For public websites, this network is the entire Internet.

IPv4 addresses, the most common form of addresses, consist of four sets of numbers separated by a dot, with each set having up to three digits. For example, `111.222.111.222` could be a valid IPv4 IP address. With DNS, we map a name to that address so that you do not have to remember a complicated set of numbers for each place you want to visit on a network.

Due to the tremendous growth of the Internet and the number of devices connected to it, the IPv4 address range has quickly been depleted. IPv6 was created to solve this depletion issue, and it has an address space of 128 bits, which allows for 340,282,366,920,938,463, 463,374,607,431,768,211,456, or 340 undecillion, unique addresses. For human beings, this number is difficult to imagine, so consider this: If each IPv4 address were one grain of sand, you would have enough addresses to fill approximately one dump truck with sand. If each IPv6 address were one grain of sand, you would have enough sand to equal the approximate size of the sun. Today, most devices and networks still communicate using IPv4, but migration to IPv6 is proceeding gradually over time.

Hosts

Within a domain, the domain owner can define individual *hosts*, which refer to separate computers or services accessible through a domain. For instance, most domain owners make their web servers accessible through the base domain (`example.com`) and also through the host definition `www` (as in `www.example.com`).

You can have other host definitions under the general domain, such as Application Program Interface (API) access through an API host (`api.example.com`) or File Transfer Protocol (FTP) access with a host definition of FTP or files (ftp.example.com or files.example.com). The host names can be arbitrary if they are unique for the domain.

Subdomains

DNS works in a hierarchal manner and allows a large domain to be partitioned or extended into multiple subdomains. TLDs can have many subdomains under them. For instance, zappos.com and audible.com are both subdomains of the .com TLD (although they are typically just called domains). The zappos or audible portion can be referred to as an SLD.

Likewise, each SLD can have subdomains located under it. For instance, the URL for the history department of a school could be `www.history.school.edu`. The `history` portion is a subdomain.

The difference between a host name and a *subdomain* is that a host defines a computer or resource, while a subdomain extends the parent domain. Subdomains are a method of subdividing the domain itself.

Whether talking about subdomains or hosts, you can see that the left-most portions of a domain are the most specific. This is how DNS works: from most to least specific as you read from left to right.

Fully Qualified Domain Name (FQDN)

Domain locations in a DNS can be relative to one another and, as such, can be somewhat ambiguous. A *Fully Qualified Domain Name (FQDN)*, also referred to as an absolute domain name, specifies a domain's location in relation to the absolute root of the DNS.

This means that the FQDN specifies each parent domain including the TLD. A proper FQDN ends with a dot, indicating the root of the DNS hierarchy. For example, mail .amazon.com is an FQDN. Sometimes, software that calls for an FQDN does not require the ending dot, but it is required to conform to ICANN standards.

In Figure 9.1, you can see that the entire string is the FQDN, which is composed of the domain name, subdomain, root, TLD, SLD and host.

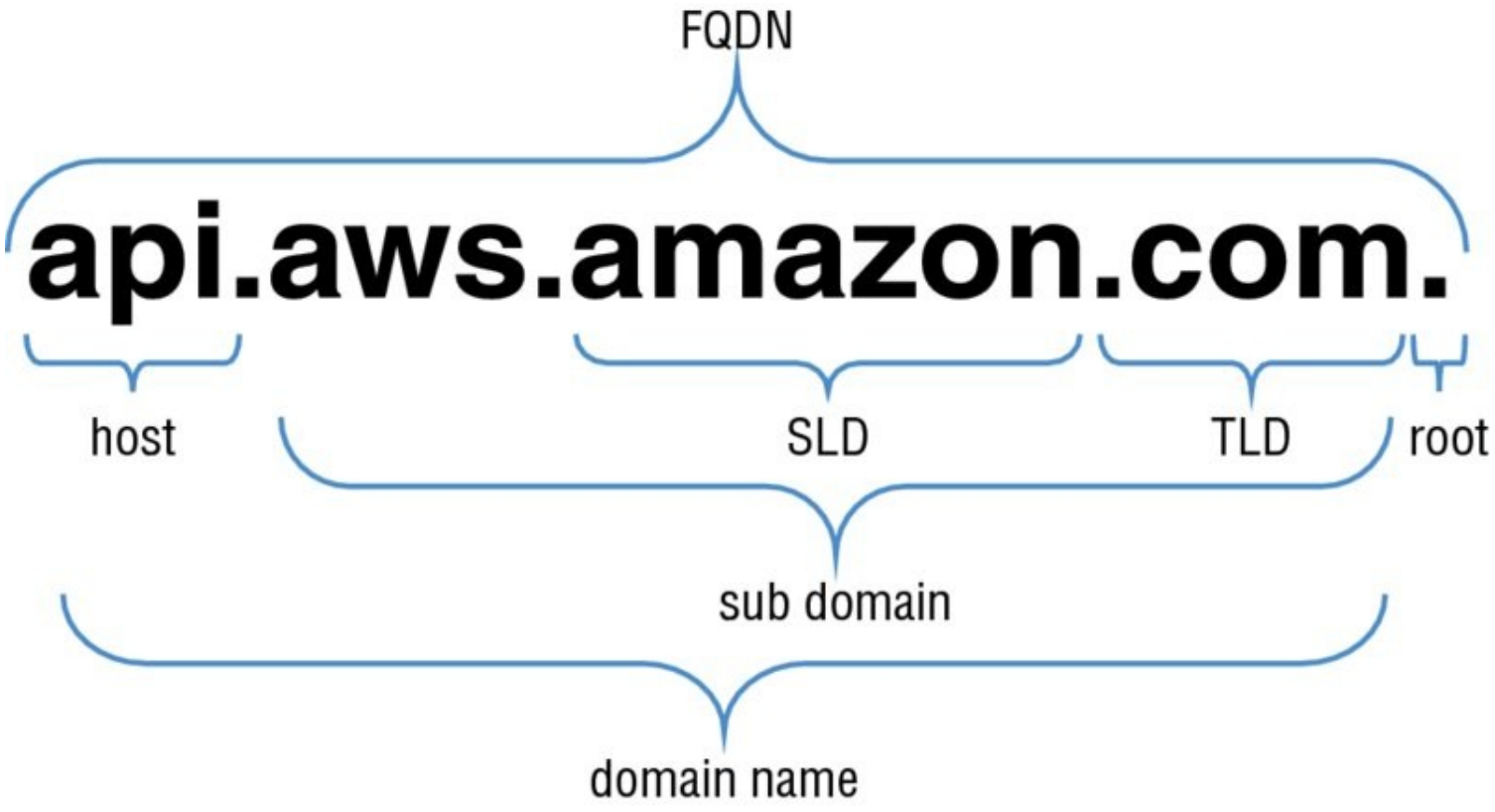


FIGURE 9.1 FQDN components

Name Servers

A *name server* is a computer designated to translate domain names into IP addresses. These

servers do most of the work in the DNS. Because the total number of domain translations is too much for any one server, each server may redirect requests to other name servers or delegate responsibility for the subset of subdomains for which they are responsible.

Name servers can be authoritative, meaning that they give answers to queries about domains under their control. Otherwise, they may point to other servers or serve cached copies of other name servers' data.

Zone Files

A *zone file* is a simple text file that contains the mappings between domain names and IP addresses. This is how a DNS server finally identifies which IP address should be contacted when a user requests a certain domain name.

> Zone files reside in name servers and generally define the resources available under a specific domain, or the place where one can go to get that information.

Top-Level Domain (TLD) Name Registrars

Because all of the names in a given domain must be unique, there needs to be a way to organize them so that domain names aren't duplicated. This is where *domain name registrars* come in. A domain name registrar is an organization or commercial entity that manages the reservation of Internet domain names. A domain name registrar must be accredited by a generic TLD (gTLD) registry and/or a country code TLD (ccTLD) registry. The management is done in accordance with the guidelines of the designated domain name registries.

Steps Involved in Domain Name System (DNS) Resolution

When you type a domain name into your browser, your computer first checks its host file to see if it has that domain name stored locally. If it does not, it will check its DNS cache to see if you have visited the site before. If it still does not have a record of that domain name, it will contact a DNS server to resolve the domain name.

DNS is, at its core, a hierarchical system. At the top of this system are root servers. ICANN delegates the control of these servers to various organizations.

As of this writing, there are 13 root servers in operation. Root servers handle requests for information about TLDs. When a request comes in for a domain that a lower-level name server cannot resolve, a query is made to the root server for the domain.

In order to handle the incredible volume of resolutions that happen every day, these root servers are mirrored and replicated. When requests are made to a certain root server, the request will be routed to the nearest mirror of that root server.

The root servers won't actually know where the domain is hosted. They will, however, be able to direct the requester to the name servers that handle the specifically-requested TLD.

For example, if a request for `www.wikipedia.org` is made to the root server, it will check its zone files for a listing that matches that domain name, but it will not find one in its records. It will instead find a record for the `.org` TLD and give the requesting entity the address of the name server responsible for `.org` addresses.

Top-Level Domain (TLD) Servers

After a root server returns the IP address of the appropriate server that is responsible for the TLD of a request, the requester then sends a new request to that address.

To continue the example from the previous section, the requesting entity would send a request to the name server responsible for knowing about `.org` domains to see if it can locate `www.wikipedia.org`.

Once again, when the name server searches its zone files for a `www.wikipedia.org` listing, it will not find one in its records. However, it will find a listing for the IP address of the name server responsible for `wikipedia.org`. This is getting much closer to the correct IP address.

Domain-Level Name Servers

At this point, the requester has the IP address of the name server that is responsible for knowing the actual IP address of the resource. It sends a new request to the name server asking, once again, if it can resolve `www.wikipedia.org`.

The name server checks its zone files, and it finds a zone file associated with `wikipedia.org`. Inside of this file, there is a record that contains the IP address for the `.www` host. The name server returns the final address to the requester.

Resolving Name Servers

In the previous scenario, we referred to a requester. What is the requester in this situation?

In almost all cases, the requester will be what is called a *resolving name server,* which is a server that is configured to ask other servers questions. Its primary function is to act as an intermediary for a user, caching previous query results to improve speed and providing the addresses of appropriate root servers to resolve new requests.

A user will usually have a few resolving name servers configured on their computer system. The resolving name servers are typically provided by an Internet Service Provider (ISP) or other organization. There are several public resolving DNS servers that you can query. These can be configured in your computer either automatically or manually.

When you type a URL in the address bar of your browser, your computer first looks to see if it can find the resource's location locally. It checks the host file on the computer and any locally stored cache. It then sends the request to the resolving name server and waits to receive the IP address of the resource.

The resolving name server then checks its cache for the answer. If it doesn't find it, it goes through the steps outlined in the previous sections.

Resolving name servers compress the requesting process for the end user. The clients simply have to know to ask the resolving name servers where a resource is located, and the resolving name servers will do the work to investigate and return the final answer.

More About Zone Files

Zone files are the way that name servers store information about the domains they know. The more zone files that a name server has, the more requests it will be able to answer authoritatively. Most requests to the average name server, however, are for domains that are

not in the local zone file.

If the server is configured to handle recursive queries, like a resolving name server, it will find the answer and return it. Otherwise, it will tell the requesting entity where to look next.

A zone file describes a DNS zone, which is a subset of the entire DNS. Zone files are generally used to configure a single domain, and they can contain a number of records that define where resources are for the domain in question.

The zone file's `$ORIGIN` directive is a parameter equal to the zone's highest level of authority by default. If a zone file is used to configure the `example.com` domain, the `$ORIGIN` would be set to `example.com`.

This parameter is either configured at the top of the zone file or defined in the DNS server's configuration file that references the zone file. Either way, this parameter defines what authoritative records the zone governs.

Similarly, the `$TTL` directive configures the default Time to Live (TTL) value for resource records in the zone. This value defines the length of time that previously queried results are available to a caching name server before they expire.

Record Types

Each zone file contains records. In its simplest form, a *record* is a single mapping between a resource and a name. These can map a domain name to an IP address or define resources for the domain, such as name servers or mail servers. This section describes each record type in detail.

Start of Authority (SOA) Record

A *Start of Authority (SOA) record* is mandatory in all zone files, and it identifies the base DNS information about the domain. Each zone contains a single SOA record.

The SOA record stores information about the following:

- The name of the DNS server for that zone
- The administrator of the zone
- The current version of the data file
- The number of seconds that a secondary name server should wait before checking for updates
- The number of seconds that a secondary name server should wait before retrying a failed zone transfer
- The maximum number of seconds that a secondary name server can use data before it must either be refreshed or expire
- The default TTL value (in seconds) for resource records in the zone

A and AAAA

Both types of address records map a host to an IP address. The A record is used to map a host to an IPv4 IP address, while AAAA records are used to map a host to an IPv6 address.

Canonical Name (CNAME)

A *Canonical Name (CNAME) record* is a type of resource record in the DNS that defines an alias for the CNAME for your server (the domain name defined in an A or AAAA record).

Mail Exchange (MX)

*Mail Exchange (MX) record*s are used to define the mail servers used for a domain and ensure that email messages are routed correctly. The MX record should point to a host defined by an A or AAAA record and not one defined by a CNAME.

Name Server (NS)

*Name Server (NS) record*s are used by TLD servers to direct traffic to the DNS server that contains the authoritative DNS records.

Pointer (PTR)

A *Pointer (PTR) record* is essentially the reverse of an A record. PTR records map an IP address to a DNS name, and they are mainly used to check if the server name is associated with the IP address from where the connection was initiated.

Sender Policy Framework (SPF)

S*ender Policy Framework (SPF) record*s are used by mail servers to combat spam. An SPF record tells a mail server what IP addresses are authorized to send an email from your domain name. For example, if you wanted to ensure that only your mail server sends emails from your company's domain, such as `example.com`, you would create an SPF record with the IP address of your mail server. That way, an email sent from your domain, such as `marketing@example.com`, would need to have an originating IP address of your company mail server in order to be accepted. This prevents people from spoofing emails from your domain name.

Text (TXT)

*Text (TXT) record*s are used to hold text information. This record provides the ability to associate some arbitrary and unformatted text with a host or other name, such as human readable information about a server, network, data center, and other accounting information.

Service (SRV)

A *Service (SRV) record* is a specification of data in the DNS defining the location (the host name and port number) of servers for specified services. The idea behind SRV is that, given a domain name (for example, `example.com`) and a service name (for example, web [HTTP], which runs on a protocol [TCP]), a DNS query may be issued to find the host name that provides such a service for the domain, which may or may not be within the domain.

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.

Supported Record Types

Amazon Route 53 supports the following DNS resource record types. When you access Amazon Route 53 using the API, you will see examples of how to format the `Value` element for each record type. Supported record types include:

- A
- AAAA
- CNAME
- MX
- NS
- PTR
- SOA
- SPF
- SRV
- TXT
- Routing Policies

When you create a resource record set, you choose a *routing policy*, which determines how Amazon Route 53 responds to queries. Routing policy options are simple, weighted, latency-based, failover, and geolocation. When specified, Amazon Route 53 evaluates a resource's relative weight, the client's network latency to the resource, or the client's geographical location when deciding which resource to send back in a DNS response.

Routing policies can be associated with health checks, so resource health status is considered before it even becomes a candidate in a conditional decision tree. A description of possible routing policies and more on health checking is covered in this section.

Simple

This is the default routing policy when you create a new resource. Use a simple routing policy when you have a single resource that performs a given function for your domain (for example, one web server that serves content for the `example.com` website). In this case, Amazon Route 53 responds to DNS queries based only on the values in the resource record set (for example, the IP address in an A record).

Weighted

With weighted DNS, you can associate multiple resources (such as Amazon Elastic Compute Cloud [Amazon EC2] instances or Elastic Load Balancing load balancers) with a single DNS name.

Use the weighted routing policy when you have multiple resources that perform the same function (such as web servers that serve the same website), and you want Amazon Route 53 to route traffic to those resources in proportions that you specify. For example, you may use this for load balancing between different AWS regions or to test new versions of your website

(you can send 10 percent of traffic to the test environment and 90 percent of traffic to the older version of your website).

To create a group of weighted resource record sets, you need to create two or more resource record sets that have the same DNS name and type. You then assign each resource record set a unique identifier and a relative weight.

When processing a DNS query, Amazon Route 53 searches for a resource record set or a group of resource record sets that have the same name and DNS record type (such as an A record). Amazon Route 53 then selects one record from the group. The probability of any resource record set being selected is governed by the following formula:

$$\frac{\textit{Weight for a given resource record set}}{\textit{Sum of the weights for the resource record sets in the group}}$$

Latency-Based

Latency-based routing allows you to route your traffic based on the lowest network latency for your end user (for example, using the AWS region that will give them the fastest response time).

Use the latency routing policy when you have resources that perform the same function in multiple AWS Availability Zones or regions and you want Amazon Route 53 to respond to DNS queries using the resources that provide the best latency. For example, suppose you have Elastic Load Balancing load balancers in the U.S. West (Oregon) region and in the Asia Pacific (Singapore) region, and you created a latency resource record set in Amazon Route 53 for each load balancer. A user in London enters the name of your domain in a browser, and DNS routes the request to an Amazon Route 53 name server. Amazon Route 53 refers to its data on latency between London and the Singapore region and between London and the Oregon region. If latency is lower between London and the Oregon region, Amazon Route 53 responds to the user's request with the IP address of your load balancer in Oregon. If latency is lower between London and the Singapore region, Amazon Route 53 responds with the IP address of your load balancer in Singapore.

Failover

Use a failover routing policy to configure active-passive failover, in which one resource takes all the traffic when it's available and the other resource takes all the traffic when the first resource isn't available. Note that you can't create failover resource record sets for private hosted zones.

For example, you might want your primary resource record set to be in U.S. West (N. California) and your secondary, Disaster Recovery (DR), resource(s) to be in U.S. East (N. Virginia). Amazon Route 53 will monitor the health of your primary resource endpoints using a health check.

A health check tells Amazon Route 53 how to send requests to the endpoint whose health you want to check: which protocol to use (HTTP, HTTPS, or TCP), which IP address and port to use, and, for HTTP/HTTPS health checks, a domain name and path.

After you have configured a health check, Amazon will monitor the health of your selected DNS endpoint. If your health check fails, then failover routing policies will be applied and your DNS will fail over to your DR site.

Geolocation

Geolocation routing lets you choose where Amazon Route 53 will send your traffic based on the geographic location of your users (the location from which DNS queries originate). For example, you might want all queries from Europe to be routed to a fleet of Amazon EC2 instances that are specifically configured for your European customers, with local languages and pricing in Euros.

You can also use geolocation routing to restrict distribution of content to only the locations in which you have distribution rights. Another possible use is for balancing load across endpoints in a predictable, easy-to-manage way so that each user location is consistently routed to the same endpoint.

You can specify geographic locations by continent, by country, or even by state in the United States. You can also create separate resource record sets for overlapping geographic regions, and priority goes to the smallest geographic region. For example, you might have one resource record set for Europe and one for the United Kingdom. This allows you to route some queries for selected countries (in this example, the United Kingdom) to one resource and to route queries for the rest of the continent (in this example, Europe) to a different resource.

Geolocation works by mapping IP addresses to locations. You should be cautious, however, as some IP addresses aren't mapped to geographic locations. Even if you create geolocation resource record sets that cover all seven continents, Amazon Route 53 will receive some DNS queries from locations that it can't identify.

In this case, you can create a default resource record set that handles both queries from IP addresses that aren't mapped to any location and queries that come from locations for which you haven't created geolocation resource record sets. If you don't create a default resource record set, Amazon Route 53 returns a "no answer" response for queries from those locations.

You cannot create two geolocation resource record sets that specify the same geographic location. You also cannot create geolocation resource record sets that have the same values for "Name" and "Type" as the "Name" and "Type" of non-geolocation resource record sets.

More on Health Checking

Amazon Route 53 health checks monitor the health of your resources such as web servers and email servers. You can configure Amazon CloudWatch alarms for your health checks so that you receive notification when a resource becomes unavailable. You can also configure Amazon Route 53 to route Internet traffic away from resources that are unavailable.

Health checks and DNS failover are major tools in the Amazon Route 53 feature set that help make your application highly available and resilient to failures. If you deploy an application in multiple Availability Zones and multiple AWS regions, with Amazon Route 53 health checks attached to every endpoint, Amazon Route 53 can send back a list of healthy endpoints only. Health checks can automatically switch to a healthy endpoint with minimal

disruption to your clients and without any configuration changes. You can use this automatic recovery scenario in active-active or active-passive setups, depending on whether your additional endpoints are always hit by live traffic or only after all primary endpoints have failed. Using health checks and automatic failovers, Amazon Route 53 improves your service uptime, especially when compared to the traditional monitor-alert-restart approach of addressing failures.

Amazon Route 53 health checks are not triggered by DNS queries; they are run periodically by AWS, and results are published to all DNS servers. This way, name servers can be aware of an unhealthy endpoint and route differently within approximately 30 seconds of a problem (after three failed tests in a row), and new DNS results will be known to clients a minute later (assuming your TTL is 60 seconds), bringing complete recovery time to about a minute and a half in total in this scenario.

 The 2014 AWS re:Invent session SDD408, "Amazon Route 53 Deep Dive: Delivering Resiliency, Minimizing Latency," introduced a set of best practices for Amazon Route 53. Explore those best practices to help you get started using Amazon Route 53 as a building block to deliver highly-available and resilient applications on AWS.

Amazon Route 53 Enables Resiliency

When pulling these concepts together to build an application that is highly available and resilient to failures, consider these building blocks:

- In every AWS region, an Elastic Load Balancing load balancer is set up with cross-zone load balancing and connection draining. This distributes the load evenly across all instances in all Availability Zones, and it ensures requests in flight are fully served before an Amazon EC2 instance is disconnected from an Elastic Load Balancing load balancer for any reason.
- Each Elastic Load Balancing load balancer delegates requests to Amazon EC2 instances running in multiple Availability Zones in an auto-scaling group. This protects the application from Availability Zone outages, ensures that a minimal amount of instances is always running, and responds to changes in load by properly scaling each group's Amazon EC2 instances.
- Each Elastic Load Balancing load balancer has health checks defined to ensure that it delegates requests only to healthy instances.
- Each Elastic Load Balancing load balancer also has an Amazon Route 53 health check associated with it to ensure that requests are routed only to load balancers that have healthy Amazon EC2 instances.
- The application's production environment (for example, `prod.domain.com`) has Amazon Route 53 alias records that point to Elastic Load Balancing load balancers. The production environment also uses a latency-based routing policy that is associated with Elastic Load Balancing health checks. This ensures that requests are routed to a healthy load balancer, thereby providing minimal latency to a client.

- 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.

Summary

In this chapter, you learned the fundamentals of DNS, which is the methodology that computers use to convert human-friendly domain names (for example, `amazon.com`) into IP addresses (such as `192.0.2.1`).

DNS starts with TLDs (for example, `.com`, `.edu`). The Internet Assigned Numbers Authority (IANA) controls the TLDs in a root zone database, which is essentially a database of all available TLDs.

DNS names are registered with a domain registrar. A registrar is an authority that can assign domain names directly under one or more TLDs. These domains are registered with InterNIC, a service of ICANN, which enforces the uniqueness of domain names across the Internet. Each domain name becomes registered in a central database, known as the WhoIS database.

DNS consists of a number of different record types, including but not limited to the following:

- A
- AAAA
- CNAME
- MX
- NS
- PTR
- SOA
- SPF
- TXT

Amazon Route 53 is a highly available and highly scalable AWS-provided DNS service. Amazon Route 53 connects user requests to infrastructure running on AWS (for example, Amazon EC2 instances and Elastic Load Balancing load balancers). It can also be used to route users to infrastructure outside of AWS.

With Amazon Route 53, your DNS records are organized into hosted zones that you configure with the Amazon Route 53 API. A hosted zone simply stores records for your domain. These records can consist of A, CNAME, MX, and other supported record types.

Amazon Route 53 allows you to have several different routing policies, including the following:

- ***Simple***—Most commonly used when you have a single resource that performs a given function for your domain
- ***Weighted***—Used when you want to route a percentage of your traffic to one particular resource or resources
- ***Latency-Based***—Used to route your traffic based on the lowest latency so that your

users get the fastest response times

- ***Failover***—Used for DR and to route your traffic from your resources in a primary location to a standby location
- ***Geolocation***—Used to route your traffic based on your end user's location

Remember to pull these concepts together to build an application that is highly available and resilient to failures. Use Elastic Load Balancing load balancers across Availability Zones with connection draining enabled, use health checks defined to ensure that the application delegates requests only to healthy Amazon EC2 instances, and use a latency-based routing policy with Elastic Load Balancing health checks to ensure requests are routed with minimal latency to clients. Use Amazon CloudFront edge locations to spread content all over the world with minimal client latency. Deploy the application in multiple AWS regions, protecting it from a regional outage.

Exam Essentials

Understand what DNS is. DNS is the methodology that computers use to convert human-friendly domain names (for example, `amazon.com`) into IP addresses (such as `192.0.2.1`).

Know how DNS registration works. Domains are registered with domain registrars that in turn register the domain name with InterNIC, a service of ICANN. ICANN enforces uniqueness of domain names across the Internet. Each domain name becomes registered in a central database known as the WhoIS database. Domains are defined by their TLDs. TLDs are controlled by IANA in a root zone database, which is essentially a database of all available TLDs.

Remember the steps involved in DNS resolution. Your browser asks the resolving DNS server what the IP address is for `amazon.com`. The resolving server does not know the address, so it asks a root server the same question. There are 13 root servers around the world, and these are managed by ICANN. The root server replies that it does not know the answer to this, but it can give an address to a TLD server that knows about `.com` domain names. The resolving server then contacts the TLD server. The TLD server does not know the address of the domain name either, but it does know the address of the resolving name server. The resolving server then queries the resolving name server. The resolving name server contains the authoritative records and sends these to the resolving server, which then saves these records locally so it does not have to perform these steps again in the near future. The resolving name server returns this information to the user's web browser, which also caches the information.

Remember the different record types. DNS consists of the following different record types: A (address record), AAAA (IPv6 address record), CNAME (canonical name record or alias), MX (mail exchange record), NS (name server record), PTR (pointer record), SOA (start of authority record), SPF (sender policy framework), SRV (service locator), and TXT (text record). You should know the differences among each record type.

Remember the different routing policies. With Amazon Route 53, you can have different routing policies. The simple routing policy is most commonly used when you have a single resource that performs a given function for your domain. Weighted routing is used when you want to route a percentage of your traffic to a particular resource or resources. Latency-based routing is used to route your traffic based on the lowest latency so that your users get the fastest response times. Failover routing is used for DR and to route your traffic from a primary resource to a standby resource. Geolocation routing is used to route your traffic based on your end user's location.

AWS®

Certified Developer

Official Study Guide

Associate (DVA-C01) Exam

Introduction to Serverless Applications

In the previous chapter, you learned about AWS Lambda and how you can write functions that run in a serverless manner. A serverless application is typically a combination of AWS Lambda and other Amazon services. You build serverless applications to allow developers to focus on their core product instead of the need to manage and operate servers or runtimes in the cloud or on-premises. This reduces overhead and lets developers reclaim time and energy that can be better spent developing reliable, scalable products and new features for applications.

Serverless applications have the following three main benefits:

- No server management
- Flexible scaling
- 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 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 instance appropriately.

Finally, 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 in the web application, you do not spend money on Amazon EC2 instances that do not run at their full capacity.

Web Server with Amazon Simple Storage Service (Presentation Tier)

Amazon Simple Storage Service (Amazon S3) can store HTML, CSS, images, and JavaScript files within an Amazon S3 bucket, and can host the website like a traditional web server. Though Amazon S3 hosts static websites, today many websites are dynamic applications,

where you can use JavaScript to create HTTP requests. These HTTP requests are sent to a Representational State Transfer (REST) endpoint service called *Amazon API Gateway*, which allows the application to save and retrieve data dynamically.

Amazon API Gateway opens up a variety of application tier possibilities. An internet-accessible HTTPS API can be consumed by any client capable of HTTPS communication. Some common presentation tier examples that you could use for your application's include the following:

Mobile app Not only can you integrate with custom business logic via Amazon API Gateway and AWS Lambda, you can use Amazon Cognito to create and manage user identities.

Static website content hosted in Amazon S3 You can enable your Amazon API Gateway APIs to be cross-origin resource sharing–compliant. This allows web browsers to invoke your APIs directly from within the static web pages.

Any other HTTPS-enabled client device Many devices can connect and communicate via HTTPS. There is nothing unique or proprietary about how clients communicate with the APIs that you create with the Amazon API Gateway service; it is pure HTTPS. No specific client software or licenses are required.

Additionally, there are several JavaScript frameworks that are widely available today, such as Angular and React, which allow you to benefit from a *Model-View-Controller* (MVC) architecture.

Amazon S3 Static Website

For the remainder of this chapter, the example bucket's name is `examplebucket`. This is for illustration purposes only, as Amazon S3 bucket names must be globally unique.

To create an Amazon S3 static website, you first need to create a bucket. Name the bucket something meaningful, such as `examplebucket`. When you use virtual hosted-style buckets with Secure Sockets Layer (SSL), the SSL wildcard certificate only matches buckets that do *not* contain periods. To work around this, use HTTP or write your own certificate verification logic. AWS recommends that you do not use periods (`.`) in bucket names when using virtual hosted-style buckets with SSL.

After you create your Amazon S3 bucket, you must enable and configure it to use static website hosting, index document, error document, and redirection rules (optional) in the AWS Management Console ➢ Amazon S3 Service. Use this `examplebucket` bucket to host a website.

The Amazon S3 bucket includes the region based on latency, cost, and regulatory requirements. Each object has a unique key. You grant permissions at the object or bucket level.

For the index document, enter the name of your home page's HTML file (typically `index.html`). Additionally, you may load a custom error page such as `error.html`. As with

many of the Amazon services, you can make these changes with the AWS Command Line Interface (AWS CLI) or in an AWS software development kit (AWS SDK). To enable this option with the AWS CLI, run the following command:

```
aws S3 website s3://examplebucket/ --index-document index.html --error-document
error.html
```

After you enable the Amazon S3 static website hosting feature, enter an endpoint that reflects your AWS Region: `examplebucket.s3-website.region.amazonaws.com`.

Configuring Web Traffic Logs

Amazon S3 allows you to log and capture information such as the number of visitors who access your website. To enable logs, create a new Amazon S3 bucket to store your logs. This excludes your log files from the website-hosting bucket. You can create a `logs-examplebucket-com` bucket, and inside of that bucket, you can create a folder you call `logs/` (or any name you choose). Use this folder to store all of your logs.

The term *folder* is used to describe logs; however, the Amazon S3 data model is a flat structure that allows you to create a bucket, and the bucket stores objects. There is no hierarchy of sub-buckets or subfolders; nevertheless, you can infer a logical hierarchy using key name prefixes and delimiters as the Amazon S3 console does. In other words, you should know that, as a developer, there is technically no such thing as an Amazon S3 folder—it is simply a key.

Now you use Amazon S3 to enable the static website–hosted bucket called `examplebucket` to log files. You can configure the target bucket for the log files at `logs-examplebucket-com`, and you can create a target prefix to send log files to a particular prefix key only.

To enable this feature with the AWS CLI, create an access control list that provides access to the log files that you want to create and then apply the `logging` policy. Here's an example:

```
aws s3api put-bucket-acl --bucket examplebucket --grant-write
'URI="http://acs.amazonaws.com.cn/groups/s3/LogDelivery"' --grant-read-acp
'URI="http://acs.amazonaws.com.cn/groups/s3/LogDelivery"'


aws s3api put-bucket-logging --bucket examplebucket --bucket-logging-status
file://logging.json
```

Here's the file `logging.json`:

```
{
  "LoggingEnabled": {
    "TargetBucket": "examplebucket",
```

```
    "TargetPrefix": "examplebucket/",
    "TargetGrants": [
      {
        "Grantee": {
          "Type": "AmazonCustomerByEmail",
          "EmailAddress": "user@example.com"
        },
        "Permission": "FULL_CONTROL"
      },
      {
        "Grantee": {
          "Type": "Group",
          "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
        },
        "Permission": "READ"
      }
    ]
  }
}
```

Creating Custom Domain Name with Amazon Route 53

Amazon Route 53 is a highly available and scalable cloud 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 names like `www.example.com` into the numeric IP addresses like `192.0.2.1` that computers use to connect to each other. Amazon Route 53 is fully compliant with IPv6 as well.

You may not want to use the Amazon S3 endpoint such as `bucket-name.s3-website-region.amazonaws.com`. Instead, you may want a more user-friendly URL such as `myexamplewebsite.com`. To accomplish this, purchase a domain name with Amazon Route 53.

You can purchase your domain from another provider and then update the name servers to use Amazon Route 53.

Amazon Route 53 effectively connects user requests to infrastructure running in AWS—such as Amazon EC2 instances, Elastic Load Balancing (ELB) load balancers, or Amazon S3 buckets—and can route users to infrastructure outside of AWS. You can use Amazon

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.

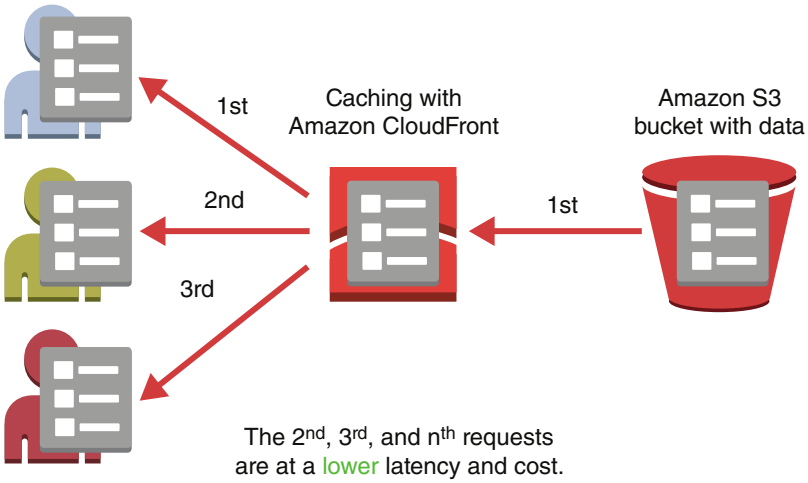
2. Specify the cache behavior. A cache behavior lets you configure a variety of CloudFront functionality for a given URL path pattern for files on your website.

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,

AWS®

Certified Developer

Official Study Guide

Associate (DVA-C01) Exam

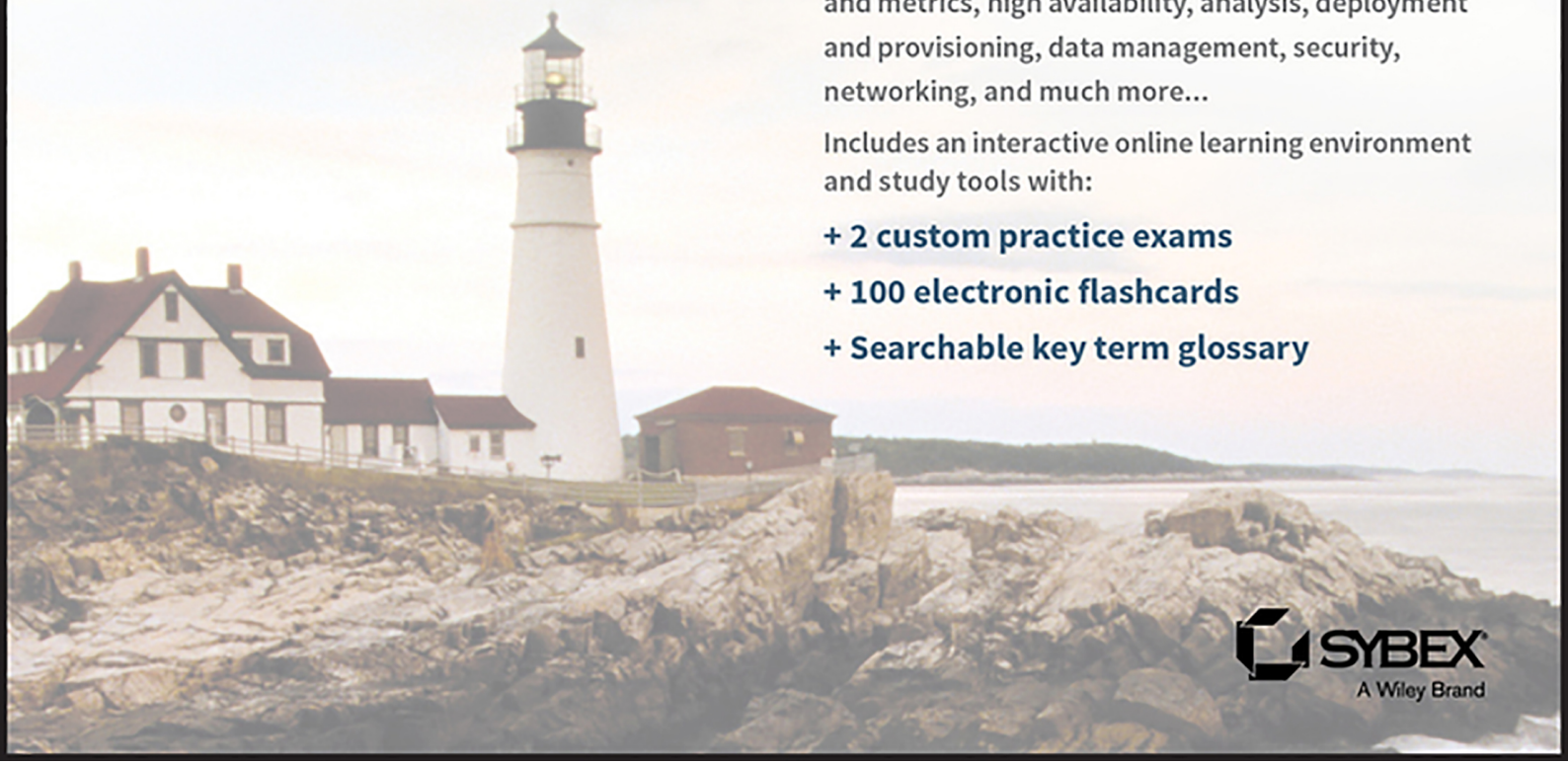
amazon web services

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

Introduction to Networking on AWS

This chapter introduces you to a number of network services that AWS provides. Some of the services described in this chapter, such as Amazon Virtual Private Cloud (Amazon VPC), are fundamental to the operation of services on AWS. Others, like Amazon Route 53, offer services that, while optional, provide tight integration with AWS products and services.

The primary goal of this book is to prepare you for the AWS Certified SysOps Administrator - Associate exam; however, we want to do more for you. The authors of this book want to provide you with as much information as possible to assist you in your everyday journey as a Systems Operator.

The AWS services covered in this chapter include:

Amazon VPC With Amazon VPC, you provision a logically-isolated section of the AWS Cloud where you launch AWS resources in a virtual network that you have defined. You have complete control over your virtual networking environment.

AWS Direct Connect AWS Direct Connect allows you to establish a dedicated network connection from your premises to AWS.

Elastic Load Balancing Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon Elastic Compute Cloud (Amazon EC2) instances in an AWS Region. You can achieve fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to route application traffic.

Virtual Private Network (VPN) connections With VPN connections, you can connect Amazon VPC to remote networks. You can take advantage of AWS infrastructure to build a highly available, highly scalable solution, or you can build your own solution.

Amazon Route 53 Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. You can use Amazon Route 53 for domain management and for DNS service to connect to both AWS and non-AWS resources.

Amazon CloudFront 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.

CloudHub

AWS VPN CloudHub also uses a VGW and, using a hub-and-spoke model, connects multiple customer gateways. AWS VPN CloudHub uses BGP with each customer location having a unique ASN.

Software VPN

Creating a software VPN involves spinning up one or more Amazon EC2 instances in one or more Availability Zones within the region and then loading VPN software onto those Amazon EC2 instances. The VPN software can be acquired directly by the customer, or it can be acquired via AWS Marketplace. AWS Marketplace offers a number of options, including OpenVPN, Cisco, Juniper, and Brocade, among others.

VPN Management

VGW is highly available and scalable. Each VGW comes with two publicly accessible IP addresses. This means that a VGW sets up two separate IPsec tunnels. You need to provision two public IP addresses on your side. These can be on a single customer gateway, two customer gateways at the same location, or two customer gateways in two different locations. You can connect multiple customer gateways to the same VGW.

AWS VPN CloudHub is highly available and scalable. With AWS VPN CloudHub, each location advertises their appropriate routes over their VPN connection. AWS VPN CloudHub receives these advertisements and re-advertises them out to the other customer gateways. This allows each site to send and receive data from the other customer sites.

Creating a software VPN gives you both the greatest level of control and the greatest level of responsibility. You spin up the instance or instances and are responsible for their placement, that they are the correct size (and can increase in size or number to meet increased demand), and that they are monitored and replaced if either is not working or working with reduced functionality.

AWS Direct Connect and VPN are the two most common methods used by large enterprises to connect their existing WAN with AWS. Most use both methods. Understanding the advantages and disadvantages of both methods and how you can use the two methods combined is very important.

Amazon Route 53

Amazon Route 53 is a highly available and scalable cloud DNS web service. DNS routes end users to Internet applications by translating names like `www.example.com` into the numeric IP addresses like `192.0.2.1` that computers use to connect to each other. Amazon Route 53 is fully compliant with IPv6.

You can use Amazon Route 53 to help you get a website or web application up and running. Amazon Route 53 enables you to perform three main functions:

Register domain names. Your website needs a name, such as `example.com`. Amazon Route 53 lets you register a name for your website or web application, known as a *domain name*.

Route Internet traffic to the resources for your domain. When a user opens a web browser and enters your domain name in the address bar, Amazon Route 53 helps the DNS connect the browser with your website or web application.

Check the health of your resources. Amazon Route 53 sends automated requests over the Internet to a resource, such as a web server, to verify that it is reachable, available, and functional. You also can choose to receive notifications when a resource becomes unavailable and choose to route Internet traffic away from unhealthy resources.

You can use any combination of these functions. For example, you can use Amazon Route 53 both to register your domain name and to route Internet traffic for the domain, or you can use Amazon Route 53 to route Internet traffic for a domain that you registered with another domain registrar. If you choose to use Amazon Route 53 for all three functions, you register your domain name, then configure Amazon Route 53 to route Internet traffic for your domain, and finally configure Amazon Route 53 to check the health of your resources. You can use Amazon Route 53 to manage both public and private hosted zones. So, you can use Amazon Route 53 to distribute traffic between multiple AWS Regions and to distribute traffic within an AWS Region.

Amazon Route 53 Implementation

In addition to registering new domains, you can transfer existing domains. When you register a domain with Amazon Route 53, a hosted zone is automatically created for that domain. This makes it easier to use Amazon Route 53 as the DNS service provider for this domain. You are, however, not obligated to use Amazon Route 53 as the DNS service provider. You may route your DNS queries to another DNS provider.

When you're using Amazon Route 53 as the DNS service provider, you need to configure the DNS service. As mentioned, when you use Amazon Route 53 as the domain registrar, a hosted zone is automatically created for you. If you are not using Amazon Route 53 as the domain registrar, then you will need to create a hosted zone.

A *hosted zone* contains information about how you want to route your traffic both for your domain and for any subdomains that you may have. Amazon Route 53 assigns a unique set of nameservers for each hosted zone that you create. You can use this set of nameservers for multiple hosted zones if you want.

A hosted zone is a collection of resource record sets for a specified domain. You create a hosted zone for a domain (such as `example.com`), and then you create resource record sets to tell the Domain Name System how you want traffic to be routed for that domain.

After you have created your hosted zones, you need to create resource record sets. This involves two parts: the record type and the routing policy. Different routing policies can be applied to different record types.

A *routing policy* determines how Amazon Route 53 responds to queries. There are five routing policies available, and each of them is explained in this chapter.

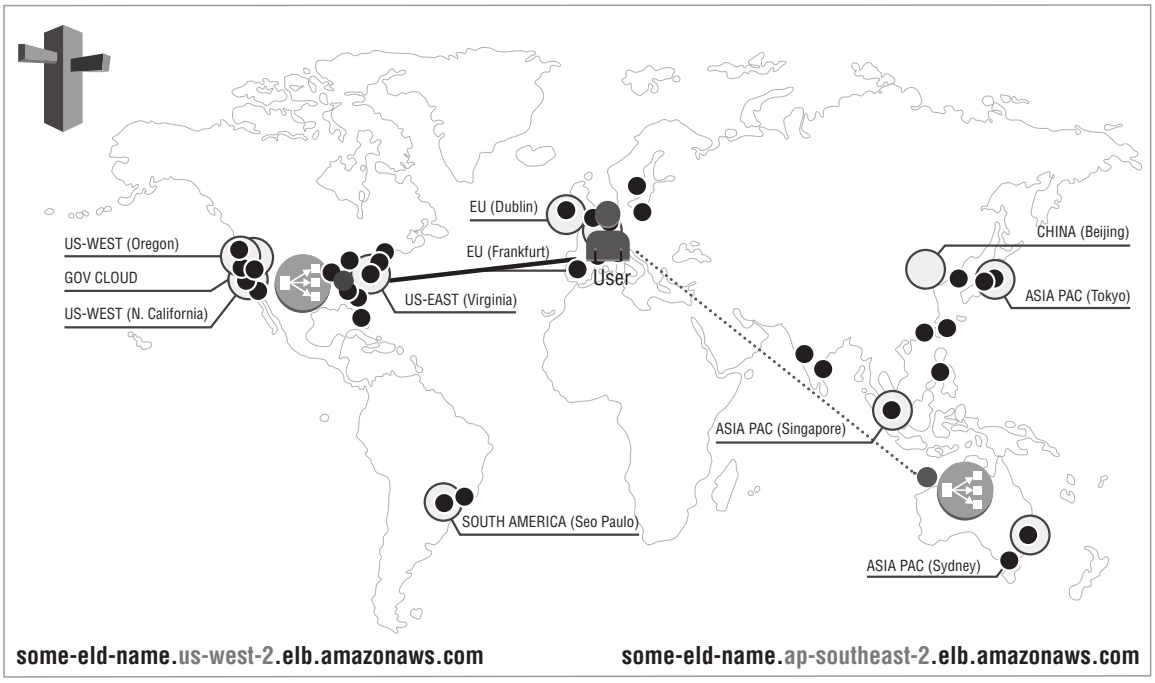
Simple Routing

Use a simple routing policy when you have a single resource that performs a given function for your domain; for example, one web server that serves content for the `example.com` website. In this case, Amazon Route 53 responds to DNS queries based only on the values in the resource record set; for example, the IP address in an A record.

Weighted Routing

Use the weighted routing policy when you have multiple resources that perform the same function (for example, web servers that serve the same website) and you want Amazon Route 53 to route traffic to those resources in proportions that you specify (for example, one quarter to one server and three quarters to the other). Figure 5.8 demonstrates weighted routing.

FIGURE 5.8 Weighted routing

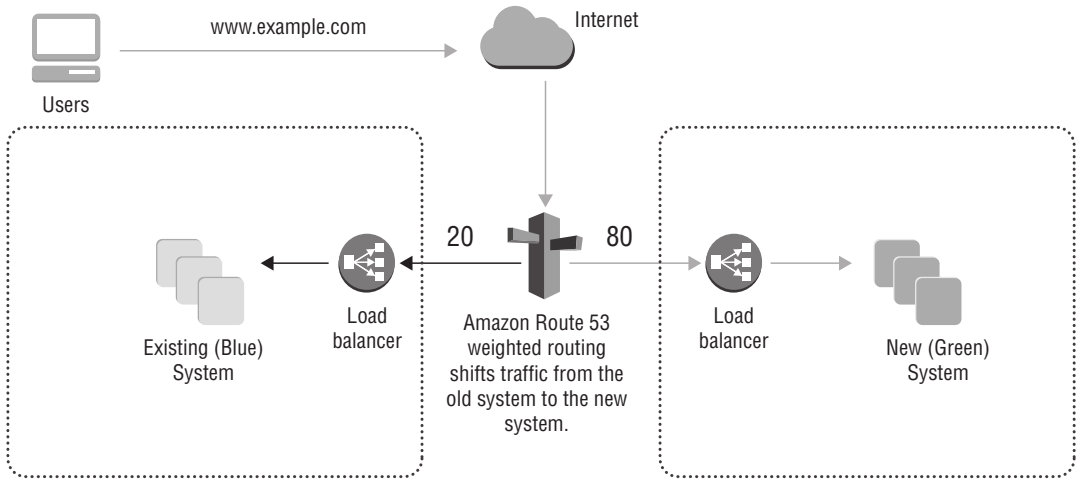


Latency-Based Routing

Use the latency routing policy when you have resources in multiple Amazon EC2 datacenters that perform the same function and you want Amazon Route 53 to respond to DNS queries

with the resources that provide the best latency. For example, you might have web servers for `example.com` in the Amazon EC2 datacenters in Ireland and in Tokyo. When a user browses to `example.com`, Amazon Route 53 chooses to respond to the DNS query based on which datacenter gives your user the lowest latency. Figure 5.9 demonstrates this concept.

FIGURE 5.9 Latency-based routing



Geolocation Routing

Use the geolocation routing policy when you want Amazon Route 53 to respond to DNS queries based on the location of your users. Geolocation routing returns the resource based on the geographic location of the user. You can specify geographic locations by continent, country, or state within the United States.

Some IP addresses aren't mapped to geographic locations, so even if you create geolocation resource record sets that cover all seven continents, Amazon Route 53 will receive some DNS queries from locations that it can't identify. You can create a default resource record set that handles both queries from IP addresses that aren't mapped to any location. If you don't create a default resource record set, Amazon Route 53 returns a "no answer" response for queries from those locations.

Failover Routing

When using a failover routing policy, you designate a primary resource and a secondary resource. The secondary resource takes over in the event of a failure of the primary resource. To accomplish this, you configure a health check for the primary resource record set. If the health check fails, Amazon Route 53 routes the traffic to the secondary resource. It is recommended, but not obligatory, to configure a health check for the secondary

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

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.

Replacement Upgrade

The replacement upgrade method replaces in-place resources with newly provisioned resources. There are advantages and disadvantages between the in-place upgrade method and replacement upgrade method. You can perform a replacement upgrade in a number of ways. You can use an Auto Scaling policy to define how you want to add (scale out) or remove (scale in) instances. By coupling this with your update strategy, you can control the rollout of an application update as part of the scaling event.

For example, you can create a new Auto Scaling Launch Configuration that specifies a new AMI containing the new version of your application. Then you can configure the Auto Scaling group to use the new launch configuration. The Auto Scaling termination policy by default will first terminate the instance with the oldest launch configuration and that is closest to the next billing hour. This in effect provides the most cost-effective method to phase out all instances that use the previous configuration. If you are using Elastic Load Balancing, you can attach an additional Auto Scaling configuration behind the load balancer and use a similar approach to phase in newer instances while removing older instances.

Similarly, you can configure rolling deployments in conjunction with deployment services such as AWS Elastic Beanstalk and AWS CloudFormation. You can use update policies to describe how instances in an Auto Scaling group are replaced or modified as part of your update strategy. With these deployment services, you can configure the number of instances to get updated concurrently or in batches, apply the updates to certain instances while isolating in-service instances, and specify the time to wait between batched updates. In addition, you can cancel or roll back an update if you discover a bug in your application code. These features can help increase the availability of your application during updates.

Blue/Green Deployments

Blue/green is a method where you have two identical stacks of your application running in their own environments. You use various strategies to migrate the traffic from your current application stack (blue) to a new version of the application (green). This method is used for a replacement upgrade. During a blue/green deployment, the latest application revision is installed on replacement instances and traffic is rerouted to these instances either immediately or as soon as you are done testing the new environment.

This is a popular technique for deploying applications with zero downtime. Deployment services like AWS Elastic Beanstalk, AWS CloudFormation, or AWS OpsWorks are particularly useful for blue/green deployments because they provide a simple way to duplicate your existing application stack.

Blue/green deployments offer a number of advantages over in-place deployments. An application can be installed and tested on the new instances ahead of time and deployed to production simply by switching traffic to the new servers. Switching back to the most recent version of an application is faster and more reliable because traffic can be routed back to the original instances as long as they have not been terminated. With an in-place deployment, versions must be rolled back by redeploying the previous version of the application. Because the instances provisioned for a blue/green deployment are new, they reflect

the most up-to-date server configurations, which helps you avoid the types of problems that sometimes occur on long-running instances.

For a stateless web application, the update process is pretty straightforward. Simply upload the new version of your application and let your deployment service deploy a new version of your stack (green). To cut over to the new version, you simply replace the Elastic Load Balancing URLs in your Domain Name Server (DNS) records. AWS Elastic Beanstalk has a Swap Environment URLs feature to facilitate a simpler cutover process. If you use Amazon Route 53 to manage your DNS records, you need to swap Elastic Load Balancing endpoints for AWS CloudFormation or AWS OpsWorks deployment services.

For applications with session states, the cutover process can be complex. When you perform an update, you don't want your end users to experience downtime or lose data. You should consider storing the sessions outside of your deployment service because creating a new stack will re-create the session database with a certain deployment service. In particular, consider storing the sessions separately from your deployment service if you are using an Amazon RDS database.

If you use Amazon Route 53 to host your DNS records, you can consider using the Weighted Round Robin (WRR) feature for migrating from blue to green deployments. The feature helps to drive the traffic gradually rather than instantly. If your application has a bug, this method helps ensure that the blast radius is minimal, as it only affects a small number of users. This method also simplifies rollbacks if they become necessary by redirecting traffic back to the blue stack. In addition, you only use the required number of instances while you scale up in the green deployment and scale down in the blue deployment. For example, you can set WRR to allow 10 percent of the traffic to go to green deployment while keeping 90 percent of traffic on blue. You gradually increase the percentage of green instances until you achieve a full cutover. Keeping the DNS cache to a shorter Time To Live (TTL) on the client side also ensures that the client will connect to the green deployment with a rapid release cycle, thus minimizing bad DNS caching behavior. For more information on Amazon Route 53, see Chapter 5, "Networking."

Hybrid Deployments

You can also use the deployment services in a hybrid fashion for managing your application fleet. For example, you can combine the simplicity of managing AWS infrastructure provided by AWS Elastic Beanstalk and the automation of custom network segmentation provided by AWS CloudFormation. Leveraging a hybrid deployment model also simplifies your architecture because it decouples your deployment method so that you can choose different strategies for updating your application stack.

Deployment Services

AWS deployment services provide easier integration with other AWS Cloud services. Whether you need to load-balance across multiple Availability Zones by using Elastic Load Balancing or by using Amazon RDS as a back end, the deployment services like AWS Elastic

scale your database horizontally. Amazon RDS MySQL, PostgreSQL, and Maria DB can have up to 5 Read Replicas, and Amazon Aurora can have up to 15 Read Replicas.

You can also place your Read Replica in a different AWS Region closer to your users for better performance. Additionally, you can use Read Replicas to increase the availability of your database by promoting a Read Replica to a master for faster recovery in the event of a disaster. Read Replicas are not a replacement for the high availability and automatic failover capabilities that Multi-AZ architectures provide, however. For a refresher on Amazon RDS high availability and Read Replicas, refer to Chapter 7.

Multi-Region High Availability

In addition to building a highly available application that runs in a single region, your application may require regional fault tolerance. This can be delivered by placing and running infrastructure in another region and then using Amazon Route 53 to load balance the traffic between the regions.

Amazon Simple Storage Service

If you need to keep Amazon Simple Storage Service (Amazon S3) data in multiple regions, you can use cross-region replication. *Cross-region replication* is a bucket-level feature that enables automatic, asynchronous copying of objects across buckets in different AWS Regions. More information on Amazon S3 and cross-region replication is available in Chapter 6, "Storage Systems."

Amazon DynamoDB

Amazon DynamoDB uses DynamoDB Streams to replicate data between regions. An application in one AWS Region modifies the data in an Amazon DynamoDB table. A second application in another AWS Region reads these data modifications and writes the data to another table, creating a replica that stays in sync with the original table.

Amazon Route 53

When you have more than one resource performing the same function (for example, more than one HTTP/S server or mail server), you can configure Amazon Route 53 to check the health of your resources and respond to Domain Name System (DNS) queries using only the healthy resources. For example, suppose your website, `Example.com`, is hosted on 10 servers, 2 each in 5 regions around the world. You can configure Amazon Route 53 to

check the health of those servers and to respond to DNS queries for `Example.com` using only the servers that are currently healthy.

You can set up a variety of failover configurations using Amazon Route 53 alias, weighted, latency, geolocation routing, and failover resource record sets.

Active-active failover Use this failover configuration when you want all of your resources to be available the majority of the time. When a resource becomes unavailable, Amazon Route 53 can detect that it is unhealthy and stop including it when responding to queries.

Active-passive failover Use this failover configuration when you want a primary group of resources to be available the majority of the time and a secondary group of resources to be on standby in case all of the primary resources become unavailable. When responding to queries, Amazon Route 53 includes only the healthy primary resources. If all of the primary resources are unhealthy, Amazon Route 53 begins to include only the healthy secondary resources in response to DNS queries.

Active-active-passive and other mixed configurations You can combine alias and non-alias resource record sets to produce a variety of Amazon Route 53 behaviors. More information on record types can be found in Chapter 5.

In order for these failover configurations to work, health checks will need to be configured. There are three types of health checks: health checks that monitor an endpoint, health checks that monitor Amazon CloudWatch Alarms, and health checks that monitor other health checks.

The following sections discuss simple and complex failover configurations.

Health Checks for Simple Failover

The simplest failover configuration of having two or more resources performing the same function can benefit from health checks. For example, you might have multiple Amazon EC2 servers running HTTP server software responding to requests for your `Example.com` website. In Amazon Route 53, you create a group of resource record sets that have the same name and type, such as weighted resource record sets or latency resource record sets of type A. You create one resource record set for each resource, and you configure Amazon Route 53 to check the health of the corresponding resource. In this configuration, Amazon Route 53 chooses which resource record set will respond to a DNS query for `Example.com` and bases the choice in part on the health of your resources.

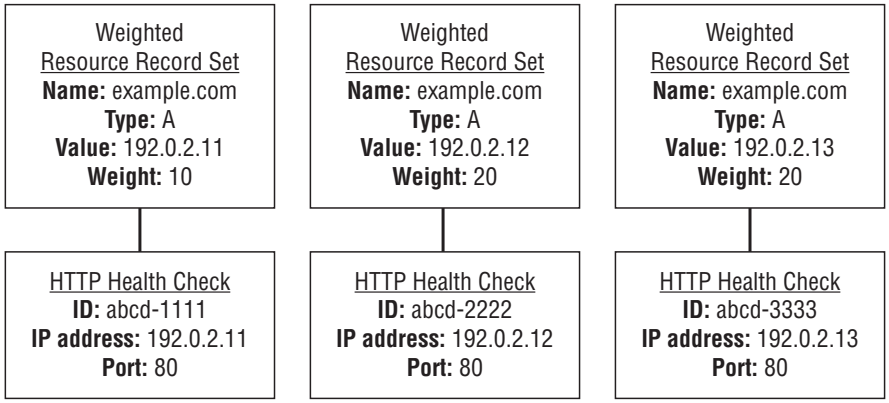
As long as all of the resources are healthy, Amazon Route 53 responds to queries using all of your `Example.com` weighted resource record sets. When a resource becomes unhealthy, Amazon Route 53 responds to queries using only the healthy resource record sets for `Example.com`.

Following are the steps for how you configure Amazon Route 53 to check the health of your resources in this simple configuration and how Amazon Route 53 responds to queries based on the health of your resources.

Configuring Amazon Route 53 to Check the Health of Your Resources

1. You identify the resources whose health you want Amazon Route 53 to monitor. For example, you might want to monitor all of the HTTP servers that respond to requests for `Example.com`.
2. You create health checks for your resources. A health check tells Amazon Route 53 how to send requests to the endpoint whose health you want to check: which protocol (HTTP, HTTPS, or Transmission Control Protocol [TCP]) and which IP address and port to use and also a domain name and path for HTTP/HTTPS health checks.

   A common configuration is to create one health check for each resource and to use the same IP address for the health check endpoint for the resource. If the IP address for your HTTP server is 192.0.2.117, you create a health check for which the IP address is 192.0.2.117.
3. You might need to configure router and firewall rules so that Amazon Route 53 can send regular requests to the endpoints that you specified in your health checks.
4. You create a group of resource record sets for your resources (for example, a group of weighted resource record sets that all have a type of A). You associate the health checks that you created in Step 2 with the corresponding resource record sets. The graphic illustrates how these health checks will operate.



5. Amazon Route 53 periodically sends a request to each endpoint that you specified when you created your health checks; it doesn't perform the health check when it receives a DNS query. Based on the responses, Amazon Route 53 decides whether the endpoints are healthy and uses that information to determine how to respond to queries.
6. When Amazon Route 53 receives a query for `Example.com`:
   a. Amazon Route 53 chooses a resource record set based on the routing policy. In this case, it chooses a resource record set based on weight.

b. It determines the current health of the selected resource record set by checking the status of the health check for that resource record set.

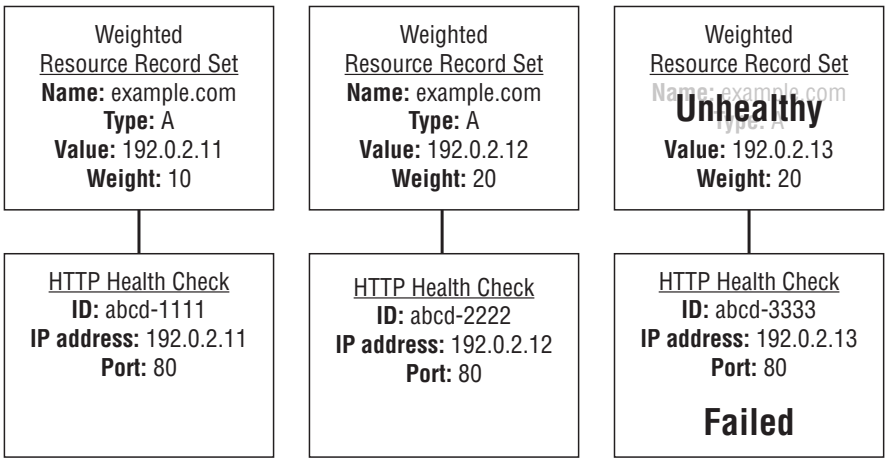
c. If the selected resource record set is unhealthy, it repeats the process of choosing a resource record set based on the routing policy. This time, the unhealthy resource record set isn't considered.

d. It responds to the query with the selected healthy resource record set.

The following example shows a group of weighted resource record sets in which the third resource record set is unhealthy. Initially, Amazon Route 53 selects a resource record set based on the weights of all three resource record sets. If it happens to select the unhealthy resource record set the first time, Amazon Route 53 selects another resource record set, but this time it omits the weight of the third resource record set from the calculation. See Figure 10.7 for an example of an unhealthy health check.

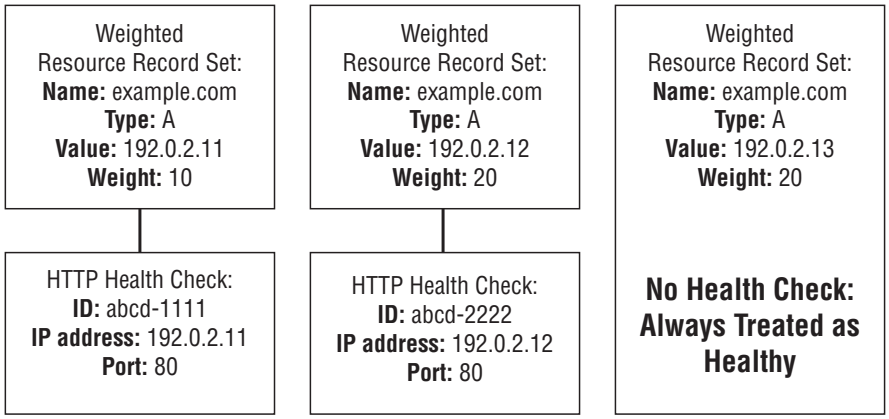
- When Amazon Route 53 initially selects from among all three resource record sets, it responds to requests using the first resource record set about 20 percent of the time: 10/(10 + 20 + 20).
- When Amazon Route 53 determines that the third resource record set is unhealthy, it responds to requests using the first resource record set about 33 percent of the time: 10/(10 + 20).

FIGURE 10.7 Unhealthy health check



If you omit a health check from one or more resource record sets in a group of resource record sets, Amazon Route 53 treats those resource record sets as healthy. Amazon Route 53 has no basis for determining the health of the corresponding resource without an assigned health check and might choose a resource record set for which the resource is unhealthy. See Figure 10.8 for more information.

FIGURE 10.8 No health check enabled

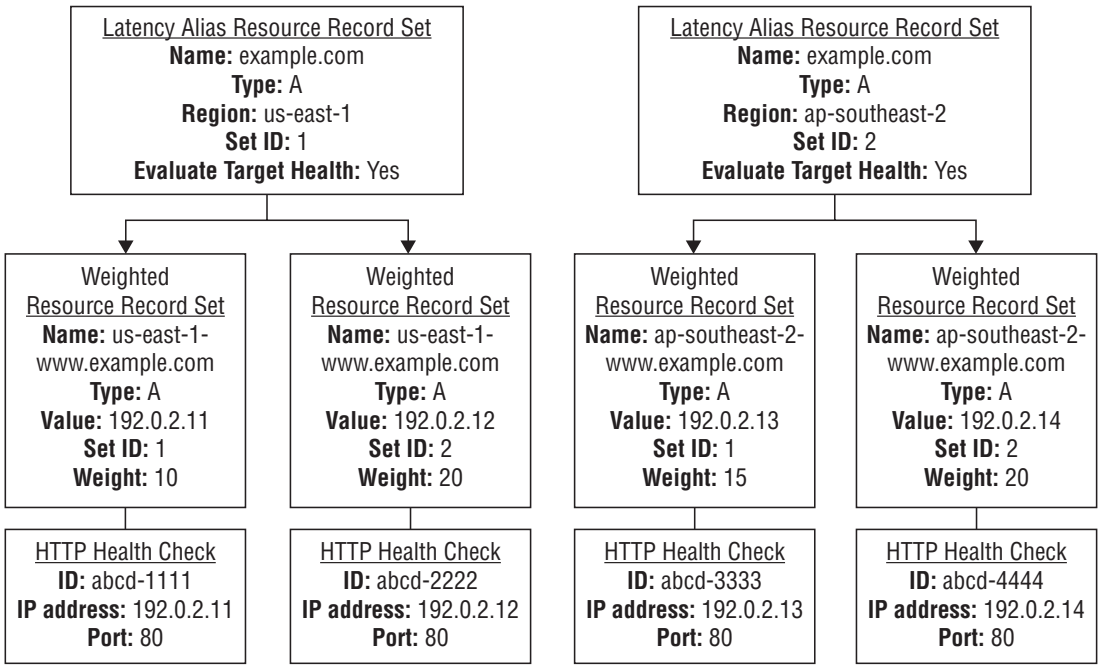


Health Checks for Complex Failover

Checking the health of resources in complex configurations works much the same way as in simple configurations. In complex configurations, however, you use a combination of alias resource record sets (including weighted alias, latency alias, and failover alias) and non-alias resource record sets to build a decision tree that gives you greater control over how Amazon Route 53 responds to requests.

For example, you might use latency alias resource record sets to select a region close to a user and use weighted resource record sets for two or more resources within each region to protect against the failure of a single endpoint or an Availability Zone. Figure 10.9 shows this configuration.

FIGURE 10.9 Health check for a complex failover



An overview of how Amazon EC2 and Amazon Route 53 are configured follows:

- You have Amazon EC2 instances in two regions: us-east-1 and ap-southeast-2. You want Amazon Route 53 to respond to queries by using the resource record sets in the region that provides the lowest latency for your customers, so you create a latency alias resource record set for each region. (You create the latency alias resource record sets after you create resource record sets for the individual Amazon EC2 instances.)
- Within each region, you have two Amazon EC2 instances. You create a weighted resource record set for each instance. The name and the type are the same for both of the weighted resource record sets in each region.

When you have multiple resources in a region, you can create weighted or failover resource record sets for your resources. You can also make even more complex configurations by creating weighted alias or failover alias resource record sets that, in turn, refer to multiple resources.

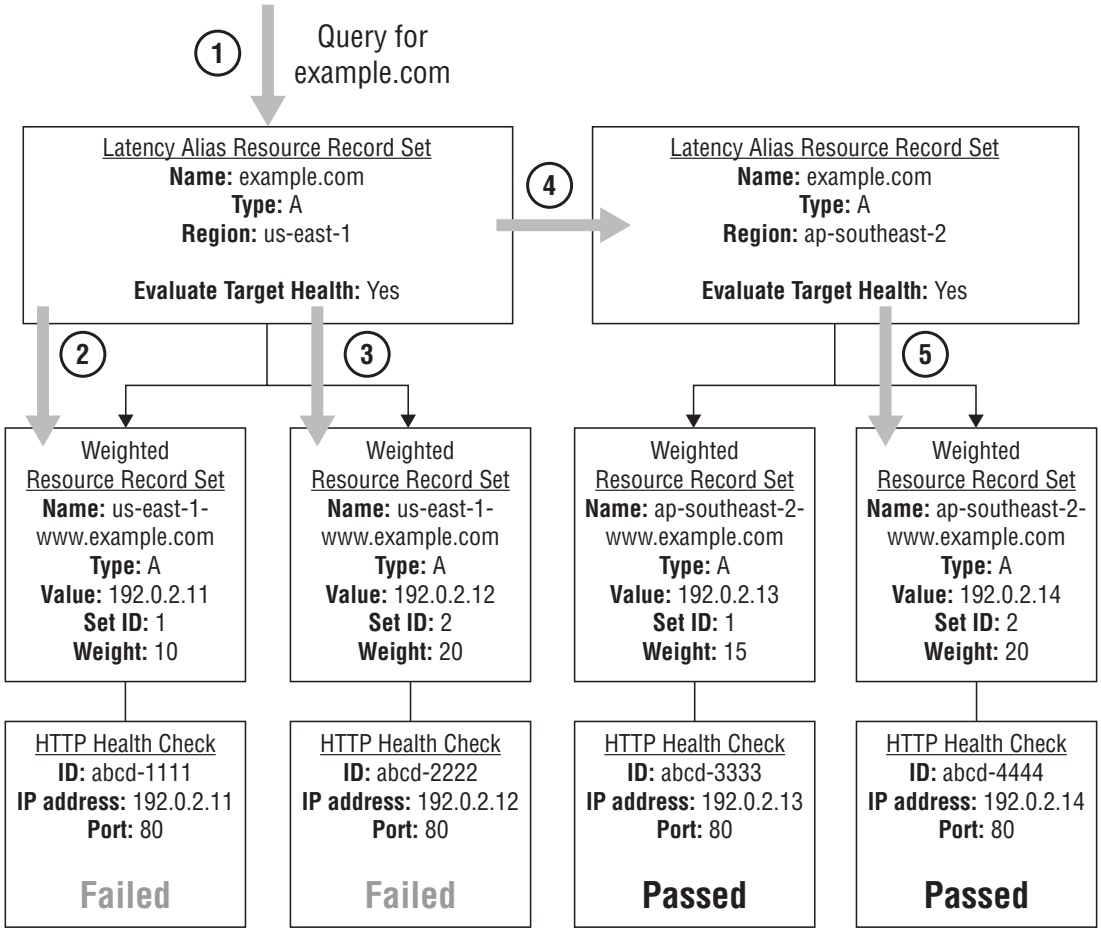
- Each weighted resource record set has an associated health check. The IP address for each health check matches the IP address for the corresponding resource record set. This isn't required, but it is the most common configuration.
- For both latency alias resource record sets, you set the value of Evaluate Target Health to Yes.

You use the Evaluate Target Health setting for each latency alias resource record set to make Amazon Route 53 evaluate the health of the alias targets—the weighted resource record sets—and respond accordingly.

Figure 10.10 demonstrates the sequence of events that follows:

1. Amazon Route 53 receives a query for `Example.com`. Based on the latency for the user making the request, Amazon Route 53 selects the latency alias resource record set for the us-east-1 region.
2. Amazon Route 53 selects a weighted resource record set based on weight. Evaluate Target Health is Yes for the latency alias resource record set, so Amazon Route 53 checks the health of the selected weighted resource record set.
3. The health check failed, so Amazon Route 53 chooses another weighted resource record set based on weight and checks its health. That resource record set also is unhealthy.
4. Amazon Route 53 backs out of that branch of the tree, looks for the latency alias resource record set with the next-best latency, and chooses the resource record set for ap-southeast-2.
5. Amazon Route 53 again selects a resource record set based on weight and then checks the health of the selected resource record set. The health check passed, so Amazon Route 53 returns the applicable value in response to the query.

FIGURE 10.10 Evaluate target health



Highly Available Connectivity Options

In this section of this chapter, we discuss how to make your connections to AWS redundant. In Chapter 5, we discussed the various connectivity options to connect to AWS and, more specifically, your Amazon VPC. We will discuss the Virtual Private Network (VPN) and AWS Direct Connect connectivity options and how to make them highly available.

Redundant Active-Active VPN Connections

To get your connectivity up and running quickly, you can implement VPN connections because they are a quick, easy, and cost-effective way to set up remote connectivity to your Amazon VPC. To enable redundancy, each AWS Virtual Private Gateway (VGW) has two VPN endpoints with capabilities for static and dynamic routing. Although statically routed

JON BONSO AND KENNETH SAMONTE

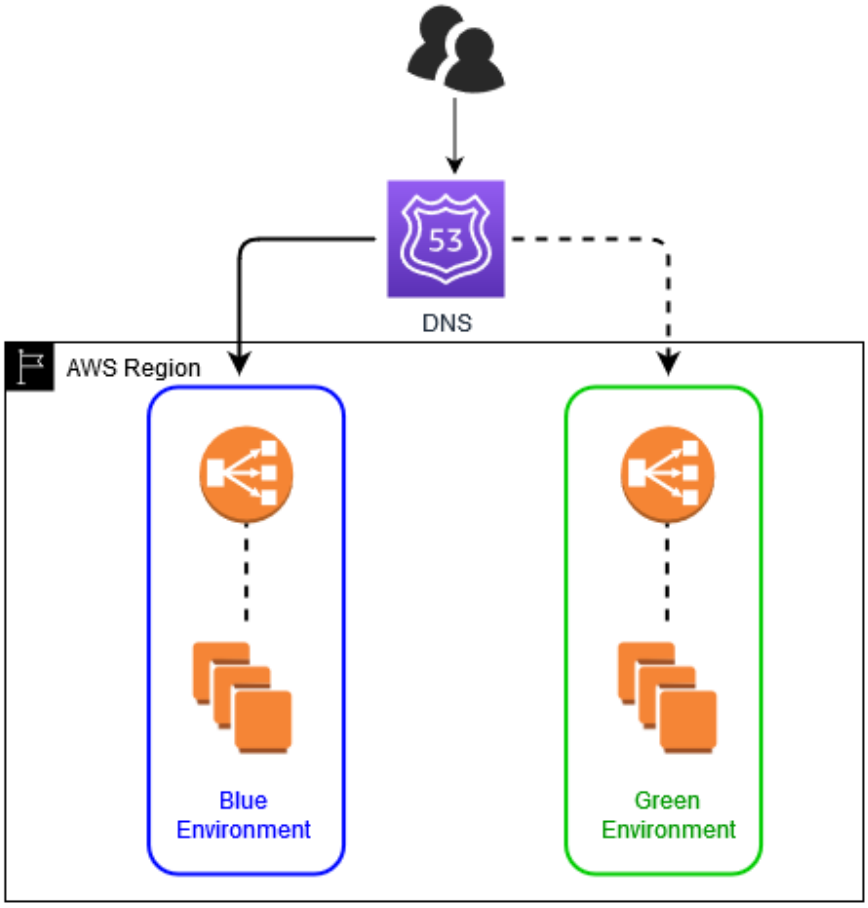
AWS CERTIFIED

DEVOPS ENGINEER PROFESSIONAL

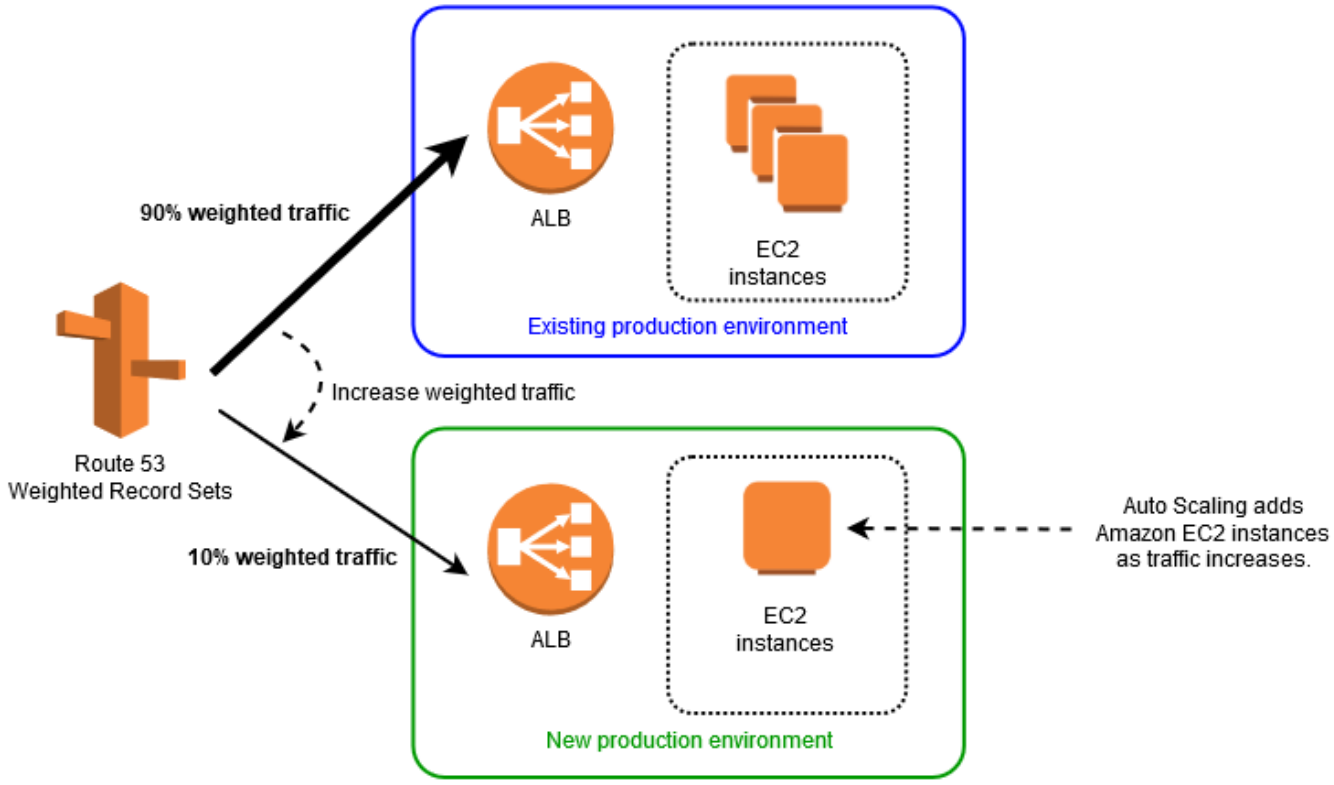
Tutorials Dojo
Study Guide and Cheat Sheets

Basic Blue/Green Deployment using Route 53

Blue/green deployment on the AWS platform provides a safer way to upgrade production software. This deployment usually involves two environments, the production environment (blue) and the new updated environment (green).



Once the new version is deployed on the green environment, you can validate the new software before going live. Then, you start shifting traffic away from the blue environment and sending it to the green one. Normally, you'd use Route 53 weighted routing policy because it gives you an easy way to push incremental traffic to the green environment or revert traffic back to the blue environment in case of issues. If you want to, you can switch the traffic immediately by updating the production Route 53 record to point to the green endpoint. Users will not see that you changed the endpoint since from their perspective, the production URL is the same.



You can also shift a small portion (like 10%) of traffic on the Green environment by using a weighted routing policy on Route 53. This way, you can test live traffic on the new environment, analyze the new logs, and then you can easily revert to the original environment if you find any problems. This process is also called a canary deployment.

Source:
https://aws.amazon.com/blogs/startups/upgrades-without-tears-part-2-bluegreen-deployment-step-by-step-on-aws/

Amazon Route 53 Routing Policies

Most large organizations often have complex network structures to support their hybrid cloud architecture, distributed systems, and global users. They have several on-premises networks integrated with AWS Direct Connect or AWS VPN to connect to their AWS cloud resources across multiple Availability Zones and AWS Regions. To ensure business continuity, companies implement a disaster recovery plan that fails over the production traffic from the primary environment to the disaster recovery (DR) site.

Amazon Route 53 is a global Domain Name System (DNS) service that allows you to route traffic across various AWS Regions and external systems outside of AWS. It provides a variety of routing policies that you can implement to meet your required use cases and automatically monitor the state and performance of your applications, servers, and other resources using health checks. You can combine two or more routing policies to comply with your company's strict RTO and RPO requirements. It helps simplify the process of setting up an active-passive or active-active failover for your disaster recovery plan by intelligently routing traffic from your primary resources to the secondary resources based on the rules you specify.

Your globally distributed resources can either be considered active or passive. It's active if it accepts live production traffic and passive if it is just on standby, which will only be activated during a failover event. You can set up an active-active failover to improve your systems' fault tolerance and performance. By having several active environments, you can ensure the high availability and resiliency of your global applications. To set up an active-active failover, you can use a single or a combination of routing policies such as latency, geolocation, geoproximity, and others to configure Route 53 to respond to a DNS query using any healthy record.

Below are the different types of Amazon Route 53 routing policies that you can use in your architecture:

- **Simple -** This routing policy is commonly used for a single resource that performs a straight-forward function for your domain records. For example, you can use this policy to route traffic from tutorialsdojo.com apex domain to an NGINX web server running on an Amazon EC2 instance.
- **Failover** – As the name implies, you can use this policy to set up an active-passive failover for your network architecture.
- **Geolocation** – Amazon Route 53 can detect the geographical location where the DNS queries originated. This routing policy lets you choose the specific resources that serve incoming traffic based on your users' geographic location. Say, you might want all user traffic from North America routed to an Application Load Balancer in the Singapore region. It works by mapping the IP addresses to geographical areas using the Extension Mechanisms for DNS version 0 (EDNS0).
- **Geoproximity** – This one is similar to the Geolocation routing policy except that it uses the Traffic Flow feature of Route 53 and has an added capability of shifting more or less traffic to your AWS services in one geographical location using a bias. It concentrates on the proximity of the resource in a given geographic area rather than its exact location.

- **Latency** – You can improve the application performance for the benefit of your global users by serving their requests from the AWS Region that provides the lowest latency. This routing policy is suitable for organizations that have resources in multiple AWS Regions.
- **Multivalue Answer** – Unlike the *Simple* routing policy, this type can route traffic to numerous resources in response to DNS queries with up to eight active records selected randomly. This policy is perfect if you are configuring an active-active failover for your network.
- **Weighted** – This policy allows you to route traffic to multiple resources in proportions that you specify. It acts as a load balancer that routes requests to a record based on the relative percentage of traffic or weight that you specified.

To monitor the system status or health, you can use Amazon Route 53 health checks to properly execute automated tasks to ensure the availability of your system. Health checks can also track the status of another health check or an Amazon CloudWatch Alarm.

Sources:
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/disaster-recovery-resiliency.html
https://tools.ietf.org/html/rfc2671

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.