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.

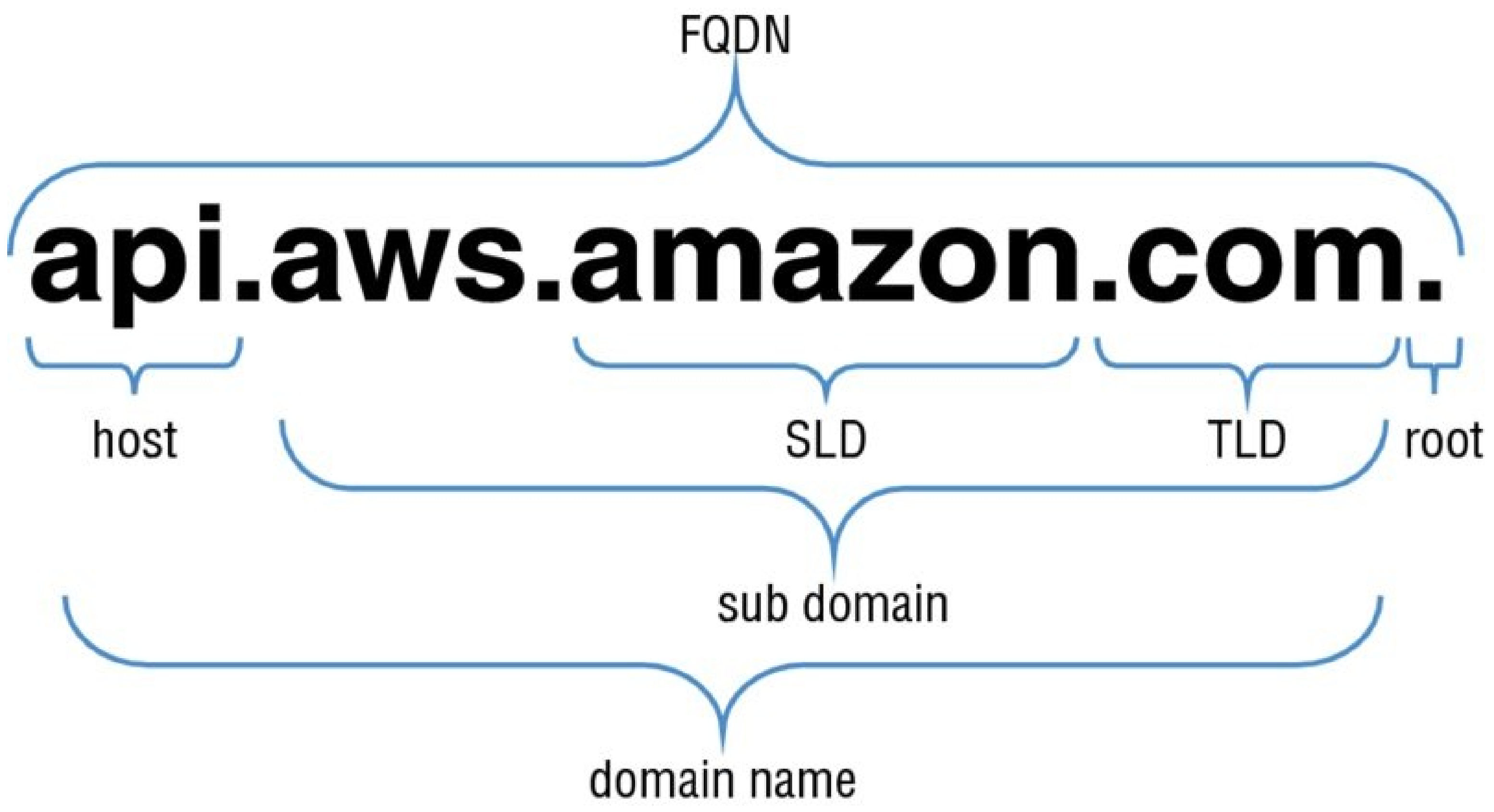

FIGURE 9.1 FQDN components

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)

*Sender 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.

TIP 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.

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

TIP 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.

TIP 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.

Exercises

In this section, you explore the different types of DNS routing policies that you can create using AWS. For specific step-by-step instructions, refer to the Amazon Route 53 information and documentation at `http://aws.amazon.com/route53/`. You will need your own domain name to complete this section, and you should be aware that Amazon Route 53 is not AWS Free Tier eligible. Hosting a zone on Amazon Route 53 should cost you a minimal amount per month per hosted zone, and additional charges will be levied depending on the routing policy you use. For current information on Amazon Route 53 pricing, refer to http://aws.amazon.com/route53/pricing/.

EXERCISE 9.1

Create a New Zone

1. Log in to the AWS Management Console.
2. Navigate to Amazon Route 53, and create a hosted zone.
3. Enter your domain name, and create your new zone file.
4. In the new zone file, you will see the SOA record and name servers. You will need to log in to your domain registrar's website, and update the name servers with your AWS name servers.
5. After you update your name servers with your domain registrars, Amazon Route 53 will be configured to serve DNS requests for your domain.

You have now created your first Amazon Route 53 zone.

EXERCISE 9.2

Create Two Web Servers in Two Different Regions

In this exercise, you will create two new Amazon EC2 web servers in different AWS regions. You will use these in the following exercises when setting up Amazon Route 53 to access the web servers.

Create an Amazon EC2 Instance

1. Log in to the AWS Management Console.
2. Change your region to Asia Pacific (Sydney).
3. In the Compute section, load the Amazon EC2 dashboard. Launch an instance, and select the first Amazon Linux Amazon Machine Image (AMI).
4. Select the instance type, and configure your instance details. Take a close look at the different options available to you, and change your instance's storage device settings as necessary.

5. Name the instance **`Sydney,`** and add a security group that allows HTTP.
6. Launch your new Amazon EC2 instance, and verify that it has launched properly.

Connect to Your Amazon EC2 Instance

7. Navigate to the Amazon EC2 instance in the AWS Management Console, and copy the public IP address to your clipboard.
8. Using a Secure Shell (SSH) client of your choice, connect to your Amazon EC2 instance using the public IP address, the user name `ec2-user`, and your private key.
9. When prompted about the authenticity of the host, type **`Yes`**, and continue.
10. You should now be connected to your Amazon EC2 instance. Elevate your privileges to root by typing **`#sudo su`**.
11. While you're logged in as the root user to your Amazon EC2 instance, run the following command to install Apache `httpd`:

    ```
    #yum install httpd -y
    ```

12. After the installation has completed, run the command `#service httpd start` followed by `#chkconfig httpd on`.
13. Navigate to the EC2 instance, and type: **`cd /var/www/html`**
14. Type **`#nano index.html`** and press Enter.
15. In Nano, type **`This is the Sydney Server`** and then press Ctrl+X.
16. Type **`Y`** to confirm that you want to save the changes, and then press Enter.
17. Type **`#ls`**. You should now see your newly created `index.html` file.
18. In your browser, navigate to `http://yourpublicipaddress/index.html`.

You should now see your "This is the Sydney Server" home page. If you do not see this, check your security group to make sure you allowed access for port 80.

Create an Elastic Load Balancing Load Balancer

19. Return to the AWS Management Console, and navigate to the Amazon EC2 dashboard.
20. Create a load balancer named **`Sydney`**, leaving the settings at their default values.
21. Create your security group, and allow all traffic in on port 80.
22. Configure health check, leaving the settings at their default values.
23. Select your newly added instance. Add tags here if you want to tag your instances.
24. Click Create to provision your load balancer.

Create These Resources in a Second Region

25. Return to the AWS Management Console, and change your region to South America (Sao Paulo).

26. Repeat the three procedures in this section to add a second Amazon EC2 instance and a load balancer in this new region.

You have now created two web servers in different regions of the world and placed these regions behind Elastic Load Balancing load balancers.

EXERCISE 9.3

Create an Alias A Record with a Simple Routing Policy

1. Log in to the AWS Management Console, and navigate to the Amazon Route 53 dashboard.
2. Select your newly-created zone domain name, and create a record set with the name **`A – IPv4 Address`**
3. Create an alias, leaving your routing policy set to Simple.
4. In your web browser, navigate to your domain name. You should now see a welcome screen for the Sydney region. If you do not see this, check that your Amazon EC2 instance is attached to your load balancer and that the instance is in service. If the instance is not in service, this means that it is failing its health check. Check that Apache HTTP Server (HTTPD) is running and that your `index.html` document is accessible.

You have now created your first Alias A record for the zone apex using the simple routing policy.

EXERCISE 9.4

Create a Weighted Routing Policy

1. Return to the AWS Management Console, and navigate to the Amazon Route 53 dashboard.
2. Navigate to hosted zones, and select your newly-created zone domain name.
3. Create a record set with type set to developer. This will create a subdomain of `developer.yourdomainname.com`.
4. Select your Sydney load balancer. Change the routing policy to Weighted with a value of 50 and a type of Sydney. Leave the other values at their defaults. Click Create. You will now see your newly-created DNS entry.
5. Create another record set with type set to developer. This will add a new record with the same name you created earlier. Both records will work together.
6. Select your Sao Paulo load balancer. Change the routing policy to Weighted with a value of 50 and type of Sao Paulo. Leave the other values at their defaults. Click Create. You will now see your newly-created DNS entry.
7. Test your DNS by visiting `http://developer.yourdomainname.com` and refreshing the page. You should be accessing the Sydney server 50 percent of the time and the Sao Paulo server the other 50 percent of the time.

You have now created a weighted DNS routing policy. You can continue to experiment with other routing policies by following the documentation at `http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html`.

EXERCISE 9.5

Create a Hosted Zone for Amazon Virtual Private Cloud (Amazon VPC)

Amazon VPC details are covered in Chapter 4, “Amazon Virtual Private Cloud (Amazon VPC).”

Create a Private Hosted Zone

1. Return to the AWS Management Console, and navigate to the Amazon Route 53 dashboard.
2. Create a hosted zone, and enter your private domain name.
3. Select the default Amazon VPC that you used in Exercise 9.2 to deploy the first server in the Asia Pacific (Sydney) region. Click Create. This will create a new zone file.

Verify Amazon VPC Configuration

4. Return to the AWS Management Console, and change your region to Asia Pacific

(Sydney).

5. In the Amazon VPC dashboard, choose your Amazon VPC.
6. Click on the default Amazon VPC from the list. Ensure that both DNS resolution and DNS host names are enabled. These settings need to use private hosted zones.

Create Resource Record Sets

7. Return to the AWS Management Console, and navigate to the Amazon Route 53 dashboard.
8. Select your newly-created private zone domain name, and create a record set.
9. Enter the name you want to give to your Amazon EC2 instance (for example, **`webserver1`**), and select IPv4 address with no alias.
10. Enter the internal IP address of your Amazon EC2 instance that you noted in Exercise 9.2.
11. Leave your routing policy set to Simple, and click Create.

Connect to Your Amazon EC2 Instance

12. On the Amazon EC2 instances screen, wait until you see your virtual machine's instance state as running. Copy the public IP address to your clipboard.
13. Using an SSH client of your choice, connect to your Amazon EC2 instance using the public IP address, the user name `ec2-user`, and your private key. For example, if you're using Terminal in OSX, you would type the following command:

    ```
    ssh ec2-user@publicipaddresshere -i MyPrivateKey.pem
    ```

14. When prompted about the authenticity of the host, type **`Yes`** and continue. You should now be connected to your Amazon EC2 instance.
15. While you're logged in to your Amazon EC2 instance, run the following command to check if the host names in Amazon Route 53 are resolving:

    ```
    nslookup webserver1.yourprivatehostedzone.com
    ```

16. You should receive a non-authoritative answer with the host name and IP address for the record set that you created in Amazon Route 53.

You have now created a private hosted zone in Amazon Route 53 and associated it with an Amazon VPC. You can continue to add instances in Amazon VPC and create resource record sets for them in Amazon Route 53. These new instances would be able to inter-communicate with the instances in the same Amazon VPC using the domain name that you created.

Remember to delete your Amazon EC2 instances and Elastic Load Balancing load balancers after you've finished experimenting with your different routing policies. You may also want to delete the zone if you are no longer using it.

Review Questions

1. Which type of record is commonly used to route traffic to an IPv6 address?
   A. An A record
   B. A CNAME
   C. An AAAA record
   D. An MX record
2. Where do you register a domain name?
   A. With your local government authority
   B. With a domain registrar
   C. With InterNIC directly
   D. With the Internet Assigned Numbers Authority (IANA)
3. You have an application that for legal reasons must be hosted in the United States when U.S. citizens access it. The application must be hosted in the European Union when citizens of the EU access it. For all other citizens of the world, the application must be hosted in Sydney. Which routing policy should you choose in order to achieve this?
   A. Latency-based routing
   B. Simple routing
   C. Geolocation routing
   D. Failover routing
4. Which type of DNS record should you use to resolve an IP address to a domain name?
   A. An A record
   B. A C Name
   C. An SPF record
   D. A PTR record
5. You host a web application across multiple AWS regions in the world, and you need to configure your DNS so that your end users will get the fastest network performance possible. Which routing policy should you apply?
   A. Geolocation routing
   B. Latency-based routing
   C. Simple routing
   D. Weighted routing
6. Which DNS record should you use to configure the transmission of email to your intended mail server?

   A. SPF records
   B. A records
   C. MX records
   D. SOA record

7. Which DNS records are commonly used to stop email spoofing and spam?
   A. MX records
   B. SPF records
   C. A records
   D. C names

8. You are rolling out A and B test versions of a web application to see which version results in the most sales. You need 10 percent of your traffic to go to version A, 10 percent to go to version B, and the rest to go to your current production version. Which routing policy should you choose to achieve this?
   A. Simple routing
   B. Weighted routing
   C. Geolocation routing
   D. Failover routing

9. Which DNS record must all zones have by default?
   A. SPF
   B. TXT
   C. MX
   D. SOA

10. Your company has its primary production site in Western Europe and its DR site in the Asia Pacific. You need to configure DNS so that if your primary site becomes unavailable, you can fail DNS over to the secondary site. Which DNS routing policy would best achieve this?
    A. Weighted routing
    B. Geolocation routing
    C. Simple routing
    D. Failover routing

11. Which type of DNS record should you use to resolve a domain name to another domain name?
    A. An A record
    B. A CNAME record
    C. An SPF record

D. A PTR record

12. Which is a function that Amazon Route 53 does not perform?

    A. Domain registration

    B. DNS service

    C. Load balancing

    D. Health checks

13. Which DNS record can be used to store human-readable information about a server, network, and other accounting data with a host?

    A. A TXT record

    B. An MX record

    C. An SPF record

    D. A PTR record

14. Which resource record set would not be allowed for the hosted zone `example.com`?

    A. `www.example.com`

    B. `www.aws.example.com`

    C. `www.example.ca`

    D. `www.beta.example.com`

15. Which port number is used to serve requests by DNS?

    A. 22

    B. 53

    C. 161

    D. 389

16. Which protocol is primarily used by DNS to serve requests?

    A. Transmission Control Protocol (TCP)

    B. Hyper Text Transfer Protocol (HTTP)

    C. File Transfer Protocol (FTP)

    D. User Datagram Protocol (UDP)

17. Which protocol is used by DNS when response data size exceeds 512 bytes?

    A. Transmission Control Protocol (TCP)

    B. Hyper Text Transfer Protocol (HTTP)

    C. File Transfer Protocol (FTP)

    D. User Datagram Protocol (UDP)

18. What are the different hosted zones that can be created in Amazon Route 53?

1. Public hosted zone
2. Global hosted zone
3. Private hosted zone

A. 1 and 2

B. 1 and 3

C. 2 and 3

D. 1, 2, and 3

19. Amazon Route 53 cannot route queries to which AWS resource?

A. Amazon CloudFront distribution

B. Elastic Load Balancing load balancer

C. Amazon EC2

D. AWS OpsWorks

20. When configuring Amazon Route 53 as your DNS service for an existing domain, which is the first step that needs to be performed?

A. Create hosted zones.

B. Create resource record sets.

C. Register a domain with Amazon Route 53.

D. Transfer domain registration from current registrar to Amazon Route 53.