SWINBURNE UNIVERSITY OF TECHNOLOGY

Cloud Computing Architect

Lecture 03 Networks

- Caching with CloudFront
- Routing with Route53

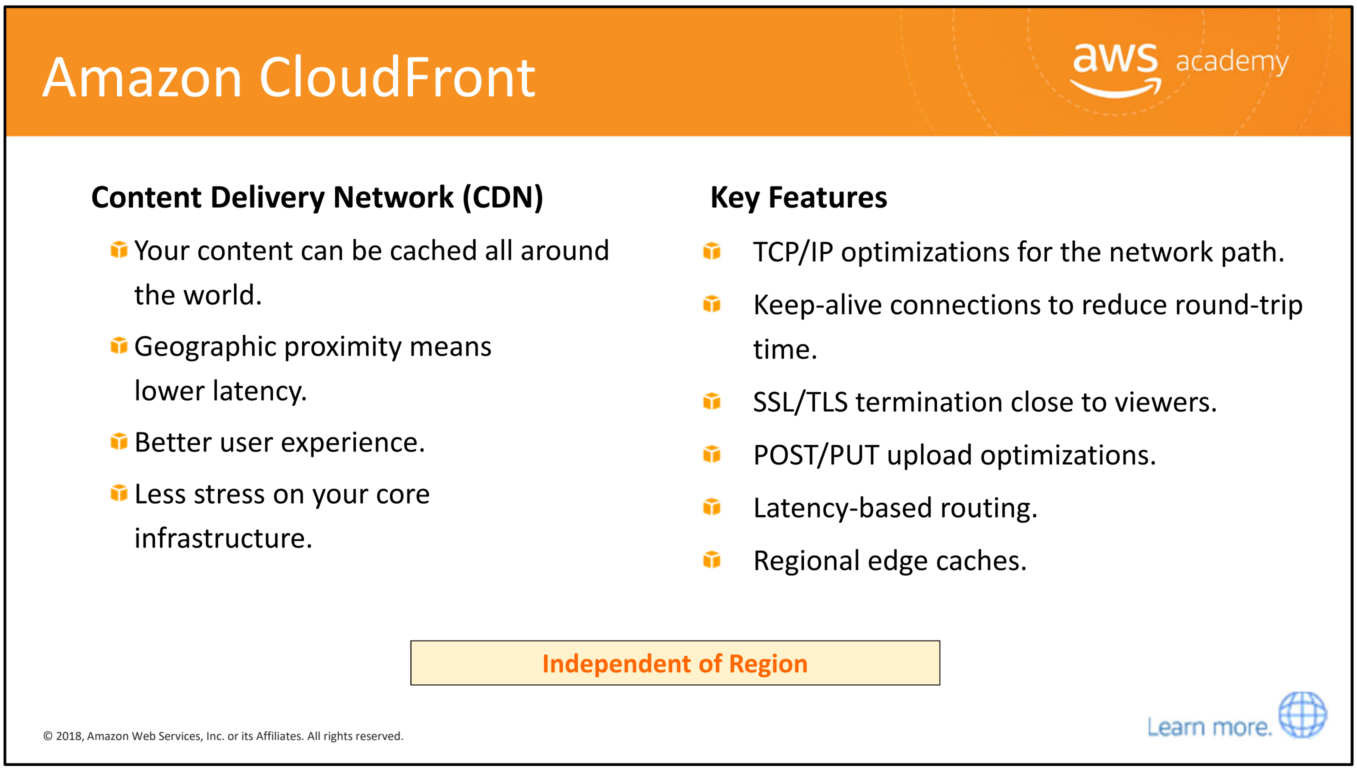
includes material from
ACA Modules 3 and 7

This week – Networks outside the VPC

- Web Caching
  - CloudFront
- Routing across Regions
  - Route53

Caching with Amazon CloudFront

Now, let’s discuss caching with Amazon CloudFront.



Amazon CloudFront

Content Delivery Network (CDN)

- Your content can be cached all around the world.
- Geographic proximity means lower latency.
- Better user experience.
- Less stress on your core infrastructure.

Key Features

- TCP/IP optimizations for the network path.
- Keep-alive connections to reduce round-trip time.
- SSL/TLS termination close to viewers.
- POST/PUT upload optimizations.
- Latency-based routing.
- Regional edge caches.

Independent of Region

Learn more.

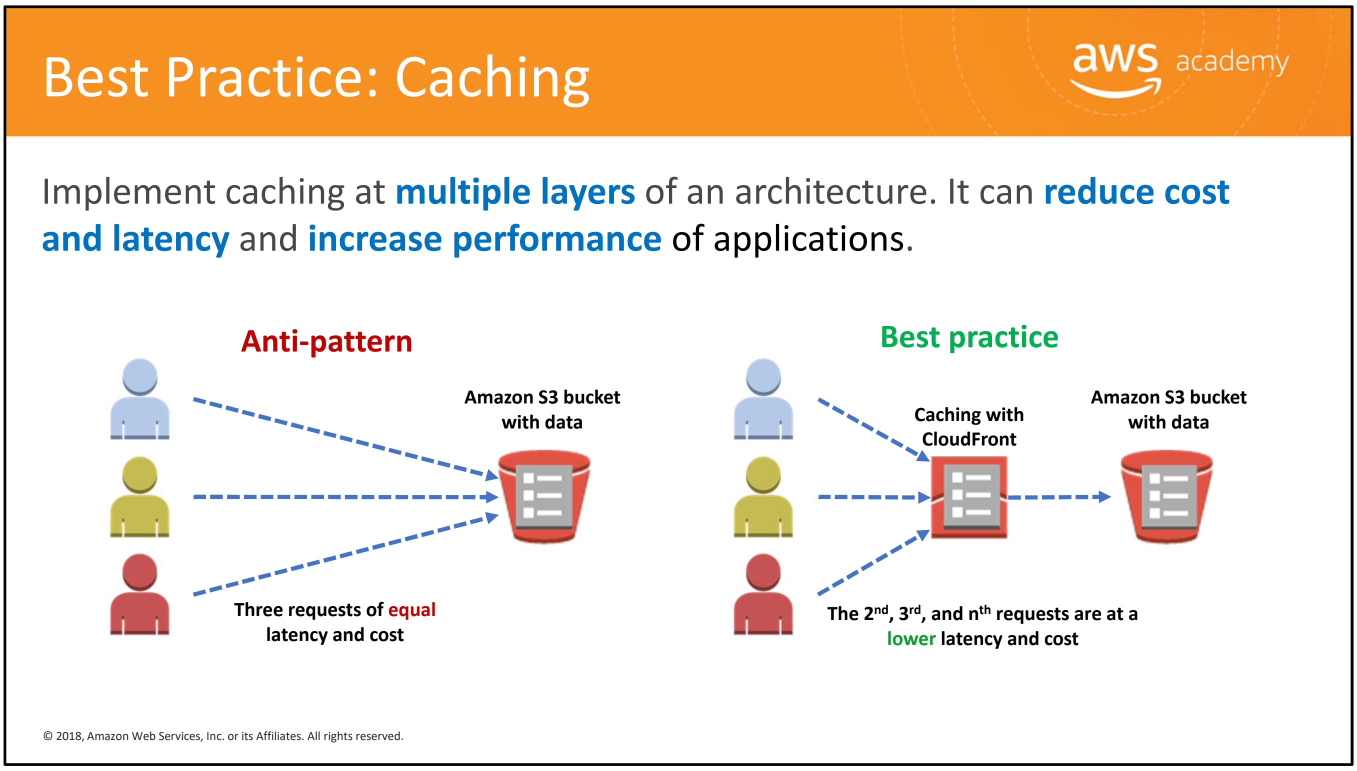
Amazon CloudFront is a web service that speeds the distribution of static and dynamic web content to users,—such as Hypertext Markup Language, or HTML, files; CSS files; JavaScript files; and image files. CloudFront is a content delivery network—or CDN—that delivers cached content through a worldwide network of data centers that are called edge locations. This geographic proximity means lower latency when you serve content to users. This makes it more cost effective and faster for a better user experience, as well as putting less stress on your core infrastructure.

Key features for CloudFront include:
- TCP/IP optimizations for the network path.
- Keep-alive connections to reduce round-trip time.
- SSL/TLS termination that is close to viewers.
- POST and PUT upload optimizations.
- Latency-based routing.
- And regional edge caches.

When a user requests content that you serve with CloudFront, the user is routed to the edge location that provides the lowest latency, or time delay. This process means that content is delivered to the user with the best possible performance. If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately. If the content is not currently in that edge location, CloudFront retrieves it from an Amazon S3 bucket or an HTTP server—for example, a web server—hat you have identified as the source for the definitive version of your content.

In addition, Amazon CloudFront has another type of edge location, which is called the regional edge cache. Regional edge caches further improves performance for your viewers. They can also help reduce the load on your origin resources, which minimizes the operational burden and the costs that are associated with scaling your origin resources. Regional edge caches are turned on by default for your CloudFront distributions, and you do not need to make any changes to your distributions to take advantage of this feature. There are also no additional charges to use this feature.

To learn more, go to the following page about CloudFront:
https://aws.amazon.com/cloudfront/details/



Best Practice: Caching

Implement caching at **multiple layers** of an architecture. It can **reduce cost and latency** and **increase performance** of applications.

Anti-pattern
Amazon S3 bucket with data
Three requests of equal latency and cost
Best practice
Caching with CloudFront
Amazon S3 bucket with data
The 2nd, 3rd, and nth requests are at a lower latency and cost

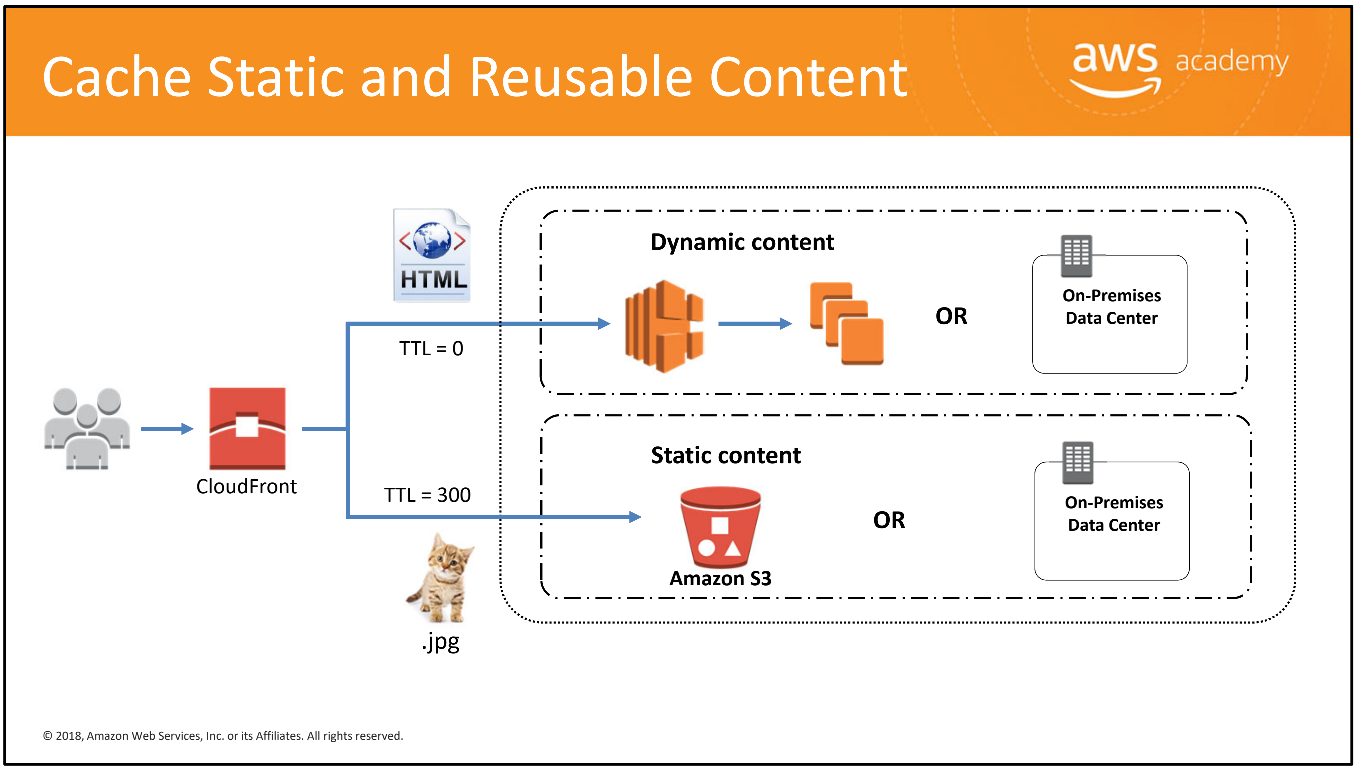
It is an architectural best practice is to implement caching at multiple layers of an architecture to reduce cost and latency, and to increase the performance of applications. It is more cost-effective to distribute files from CloudFront than from an Amazon S3 bucket.

Caching is the process of temporarily storing data or files in an intermediary location between the requester and the permanent storage. The purpose of caching is to make responding to future requests faster and reduce network throughput.

For instance, in the anti-pattern that is shown on the slide, your Amazon S3 bucket does not use a caching service. Three users request a file from one of your Amazon S3 buckets, one at a time. The file is delivered to each user in the same way. The result is that each request takes the same amount of time to complete. This process also incurs costs for the three separate times that the file is delivered for each request.

Let’s compare the process in the anti-pattern with a better pattern. In the best practice pattern, your infrastructure places Amazon CloudFront—which offers caching—in front of Amazon S3. In this scenario, the first request checks for the file in CloudFront. If the request does not find the file in CloudFront, it pulls the file from Amazon S3, and stores a copy of the file in CloudFront at the edge location that is closest to the user. The request then sends a copy of the file to the user who made the request. Now, when any other users request that file, it's retrieved from the closer edge location in CloudFront. The request does not have to go to Amazon S3 to get the file.

The best practice pattern reduces both latency and cost. After the first request is complete, you no longer pay for the file to be transferred out of Amazon S3.

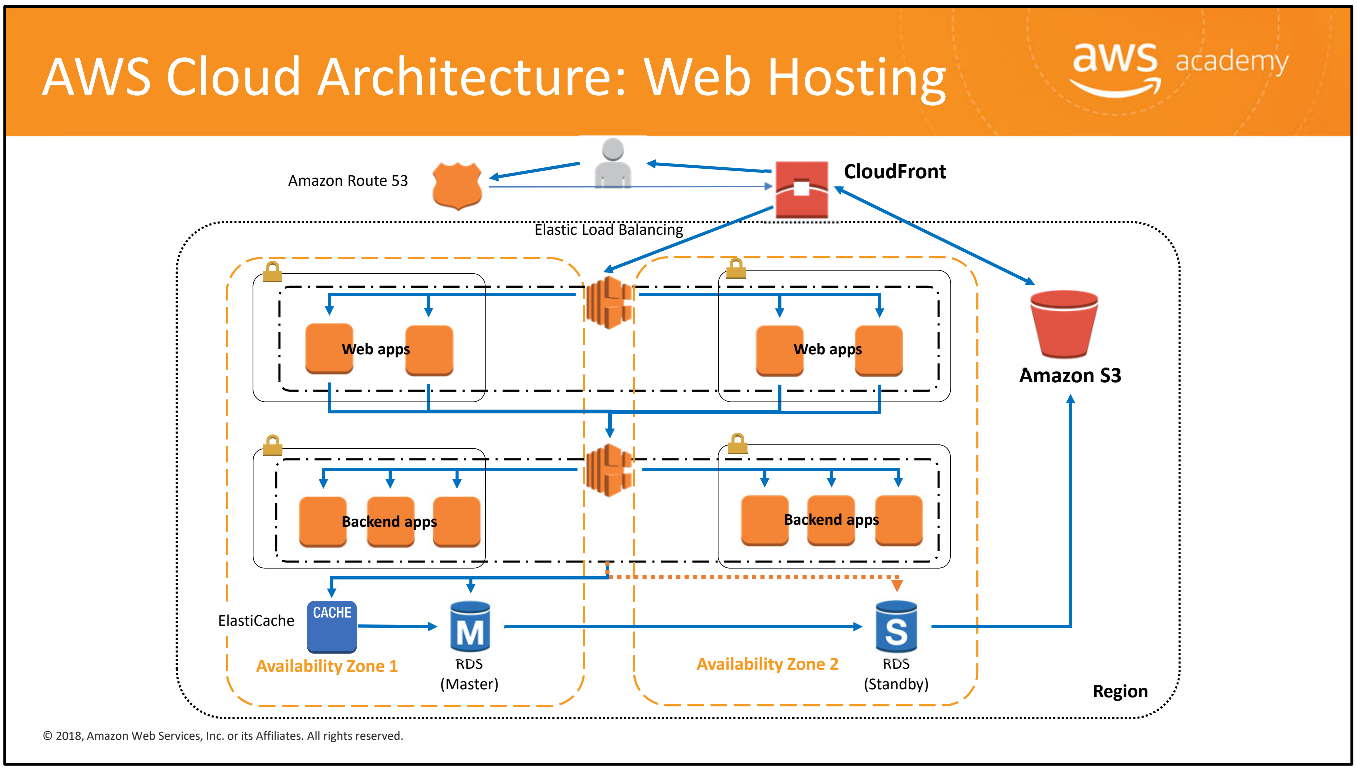


Cache Static and Reusable Content

This slide shows another way to use Amazon CloudFront. In general, you only cache static content. However, dynamic or unique content affects the performance of your application. Depending on the demand, you might still get some performance gain by caching the dynamic or unique content in Amazon S3.

For example, if there's something that needs to be personalized, the time to live is zero. A time to live of zero tells CloudFront that each and every time it sees the dynamic content, it needs to go back to its origin because it will change a lot.

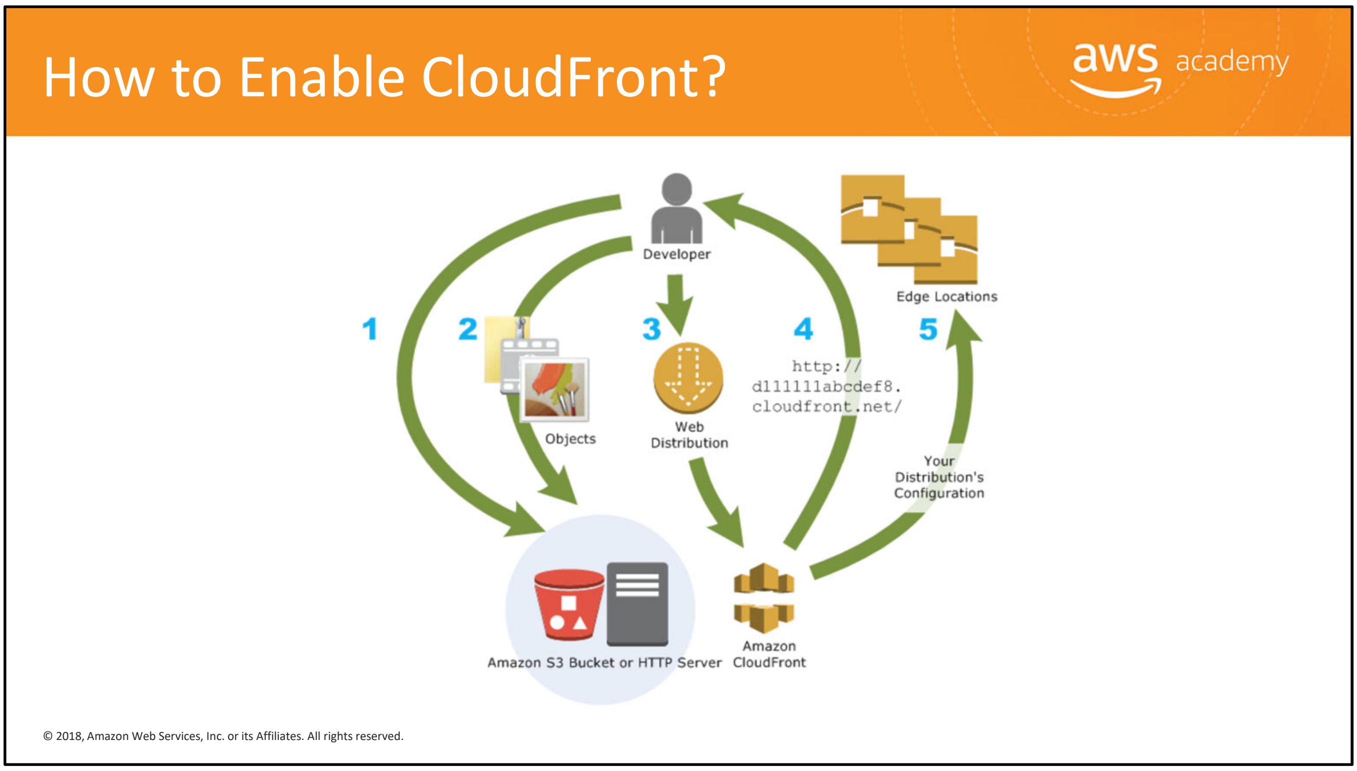
At the same time, the time to live can be set to 5 minutes or 24 hours, depending on how long the content is good for. CloudFront can pull content from Amazon S3 so that it can save the load back to an on-premises data center. When you save a load, you can have smaller instances and save money, and resources can perform more efficiently.



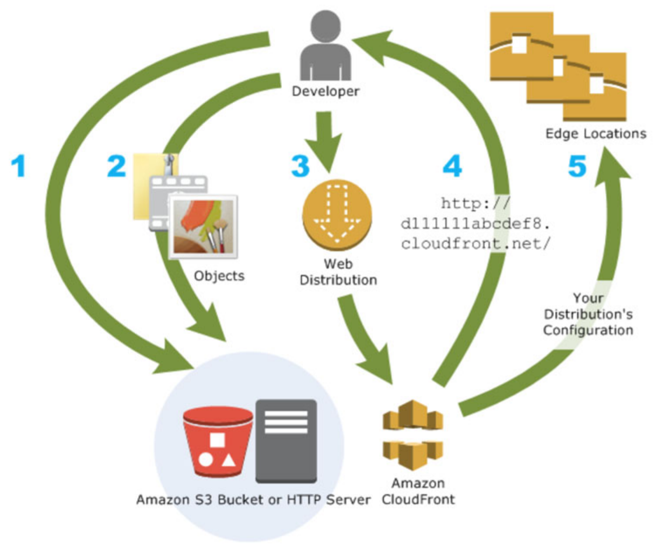
AWS Cloud Architecture: Web Hosting

Let’s take a look at cloud architecture for web hosting. We have our Elastic Load Balancing load balancers in place with CloudFront. End users are directed to CloudFront via Amazon Route 53. The load balancers pull content and data from the Amazon S3 bucket, Amazon RDS, or Amazon ElastiCache. This can serve as a read replica if content is cached there.

This architecture diagram shows how you can use CloudFront in front of your hosting architecture to decrease the number of times CloudFront must redirect requests to the load balancer, and pull content.



How to Enable CloudFront?


Developer
Edge Locations
1
2
3
4
5
Objects
Web Distribution
http://d111111abcdef8.cloudfront.net/
Your Distribution's Configuration
Amazon S3 Bucket or HTTP Server
Amazon CloudFront

How can you enable Amazon CloudFront?

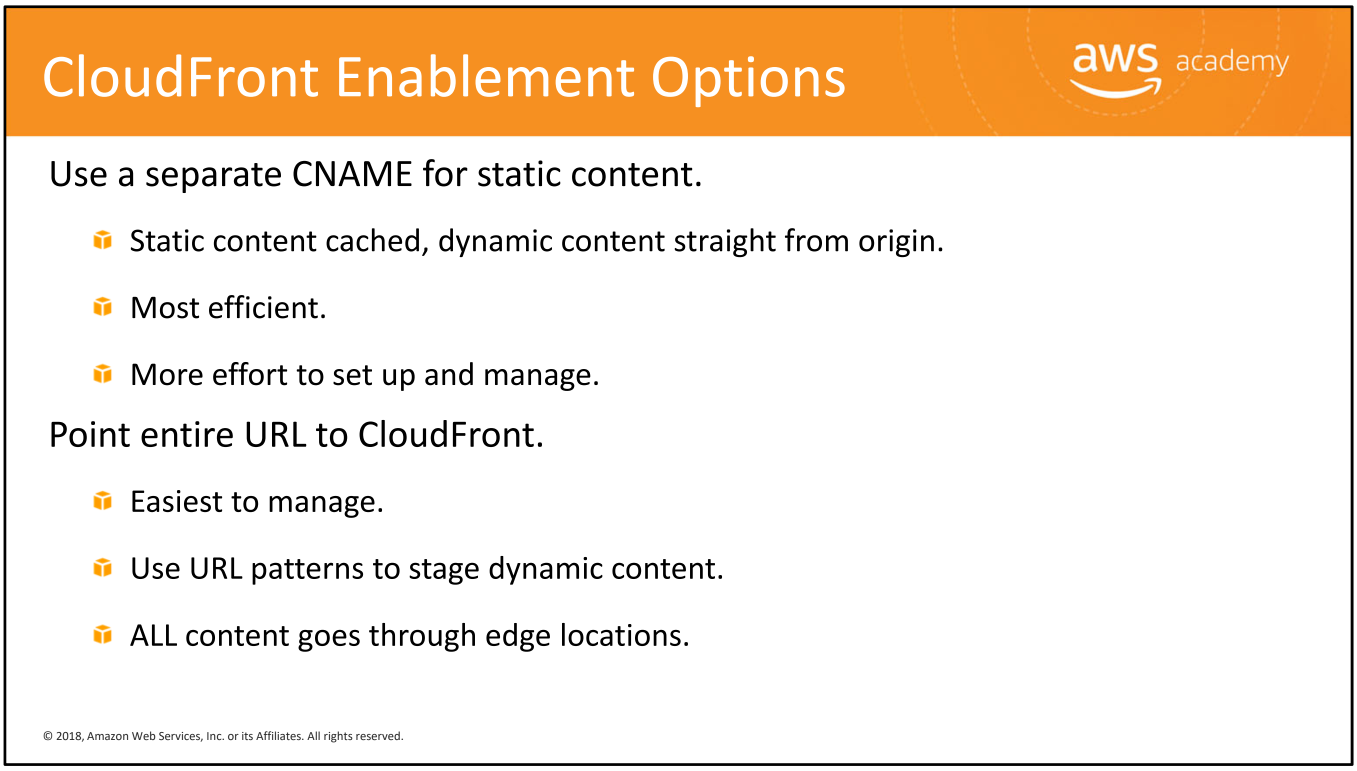
First, you specify origin servers, like an Amazon S3 bucket or your own HTTP server. CloudFront gets your files from the origin servers, and the files will then be distributed from CloudFront edge locations all over the world.

Second, you upload your files to your origin servers. Your files, which are also known as objects, typically include webpages, images, and media files.

Third, you create a CloudFront distribution, which tells CloudFront which origin servers to get your files from when users request the files through your website or application. At the same time, you specify details, such as whether you want CloudFront to log all requests, and whether you want the distribution to be enabled as soon as it's created.

Fourth, CloudFront assigns a domain name to your new distribution. You can see the domain name in the CloudFront console. The domain name can also be returned in the response to a programmatic request, like a request from an application programming interface, or API.

Fifth, CloudFront sends your distribution's configuration—but not your content—to all of its edge locations. -collections of servers in geographically dispersed data centers where CloudFront caches copies of your objects.



CloudFront Enablement Options

Use a separate CNAME for static content.

- Static content cached, dynamic content straight from origin.
- Most efficient.
- More effort to set up and manage.

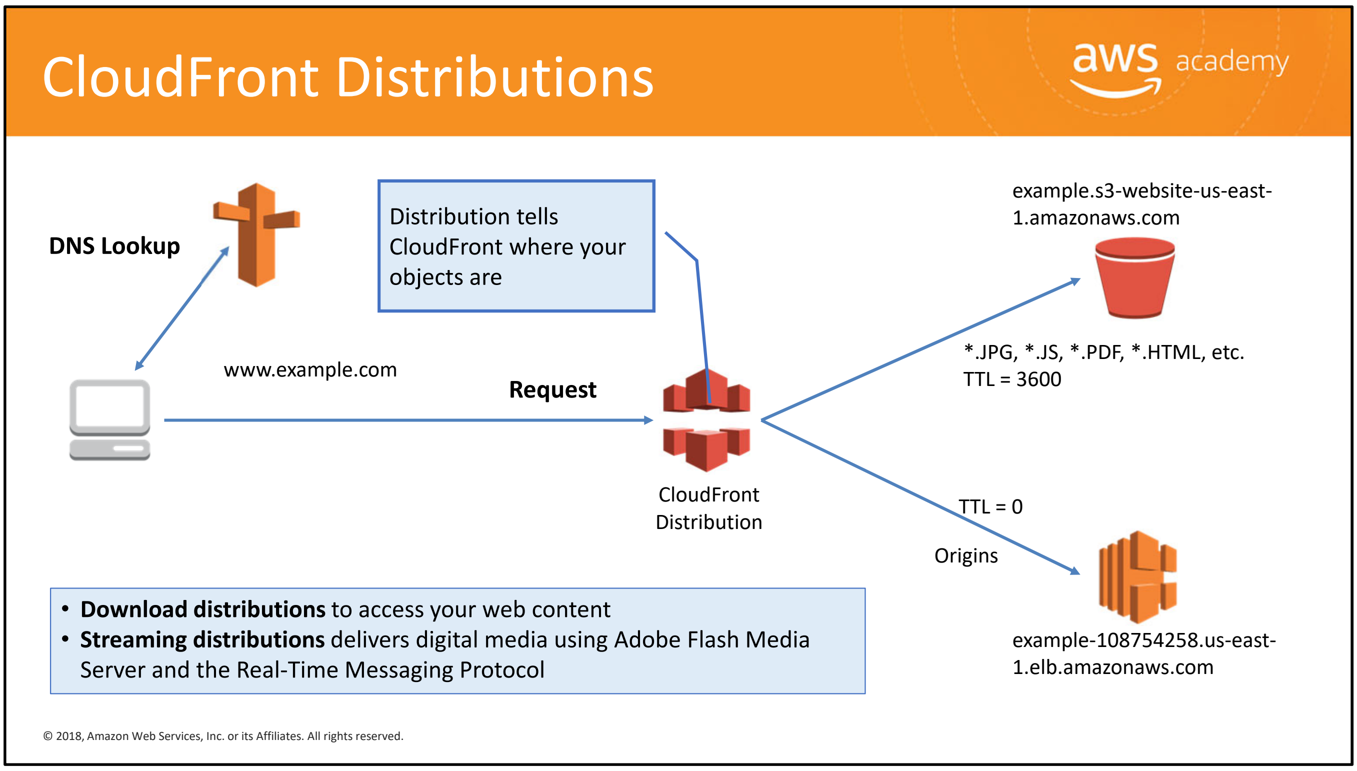
Point entire URL to CloudFront.

- Easiest to manage.
- Use URL patterns to stage dynamic content.
- ALL content goes through edge locations.

Amazon CloudFront has several options for enablement.

If you choose to use a separate Canonical Name Record—or CNAME—for static content, the static content is cached straight from the origin server. This process is the most efficient, but it takes more effort to set up and manage.

If you point the entire uniform resource locator—or URL—to CloudFront, it's easier to manage. You can use URL patterns to stage dynamic content. All of the content goes through edge locations.



CloudFront Distributions

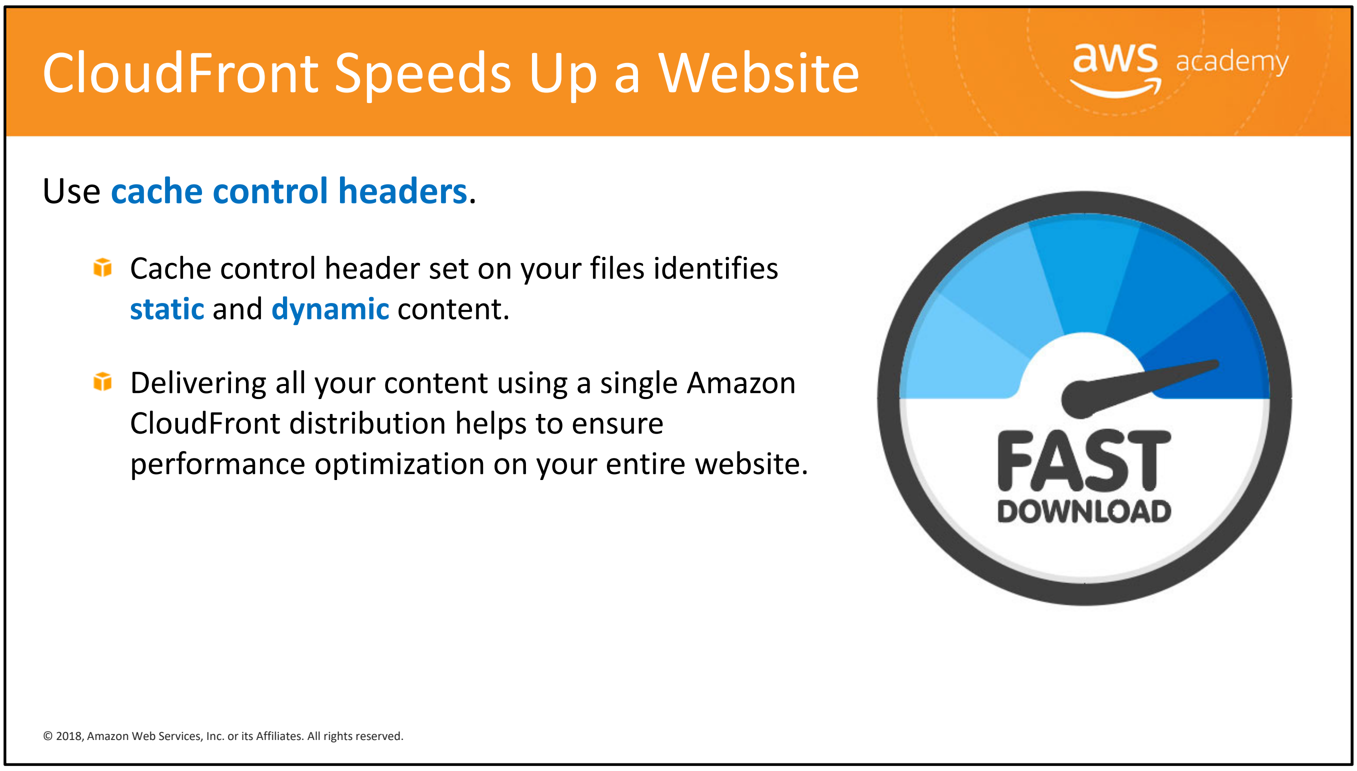
aws academy
DNS Lookup
Distribution tells CloudFront where your objects are
www.example.com
Request
CloudFront Distribution
example.s3-website-us-east-1.amazonaws.com
*.JPG, *.JS, *.PDF, *.HTML, etc.
TTL = 3600
TTL = 0
Origins
example-108754258.us-east-1.elb.amazonaws.com

- **Download distributions** to access your web content
- **Streaming distributions** delivers digital media using Adobe Flash Media Server and the Real-Time Messaging Protocol

Now let's look at how the CloudFront distributions work. In this example, Amazon Route 53 resolves example.com on behalf of the client. A request is made to the CloudFront distribution. CloudFront looks at the Amazon S3 bucket for the content that it identifies as being stored statically. For any content that has a time to live of zero, CloudFront will go directly to the Elastic Load Balancing load balancer because it has to go back to the origin server to pull the content. In this way, CloudFront delivers both static and dynamic content.

There are two distribution types:

- Web distribution, which lets you access your web content in any combination of up to 10 Amazon S3 buckets or custom origin servers.
- And Real Time Messaging Protocol—or RTMP—distribution, which is always an Amazon S3 bucket.



CloudFront Speeds Up a Website

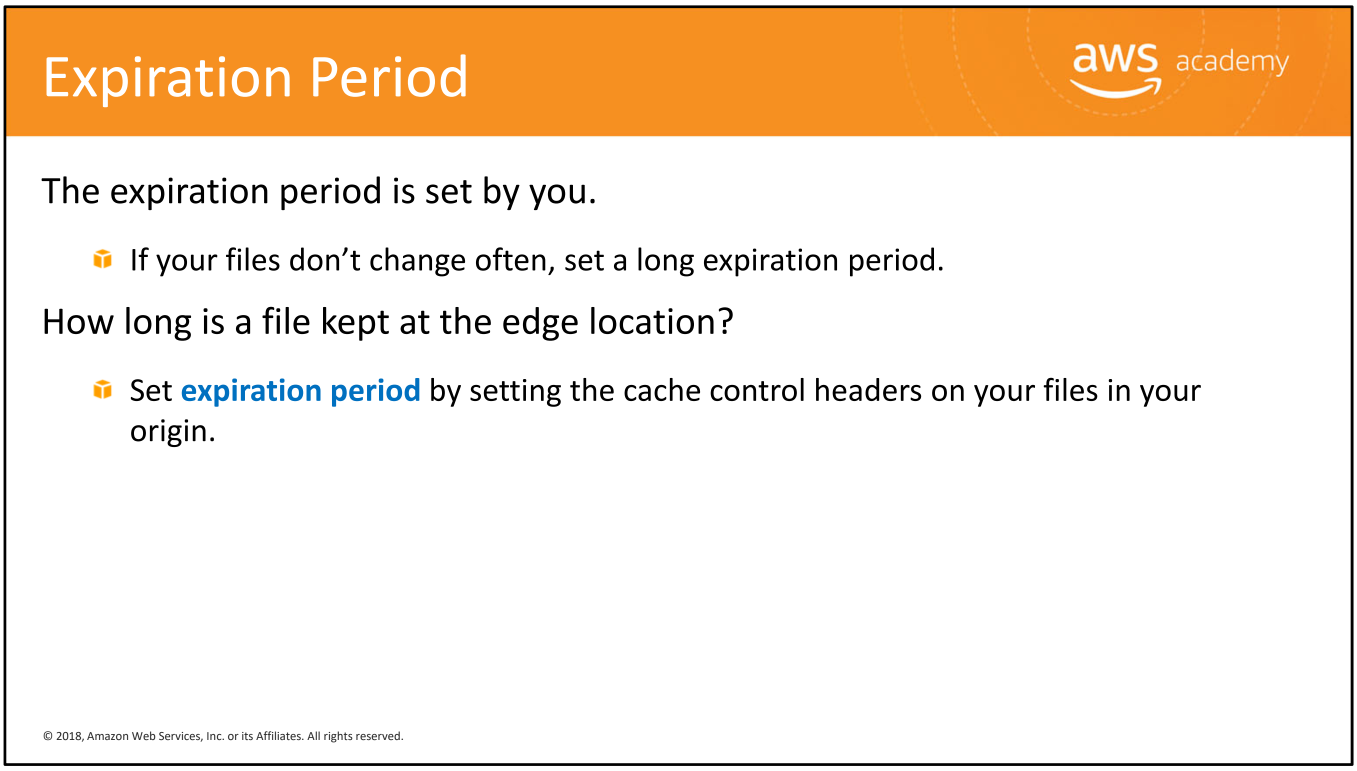
Use **cache control headers**.

- Cache control header set on your files identifies **static** and **dynamic** content.
- Delivering all your content using a single Amazon CloudFront distribution helps to ensure performance optimization on your entire website.

How does CloudFront speed up a website? Amazon CloudFront reads cache control headers to determine how frequently it needs to check the origin server for an updated version of that file. For an expiration period set to 0 seconds, Amazon CloudFront will revalidate every request with the origin server.

The cache control header that is set on your files identifies both static and dynamic content. The cache control header can even have custom headers within the CloudFront distribution graphical user interface—or GUI— within the console.

Delivering all your content by using a single Amazon CloudFront distribution helps to ensure performance optimization on your entire website.



Expiration Period

The expiration period is set by you.

- If your files don’t change often, set a long expiration period.

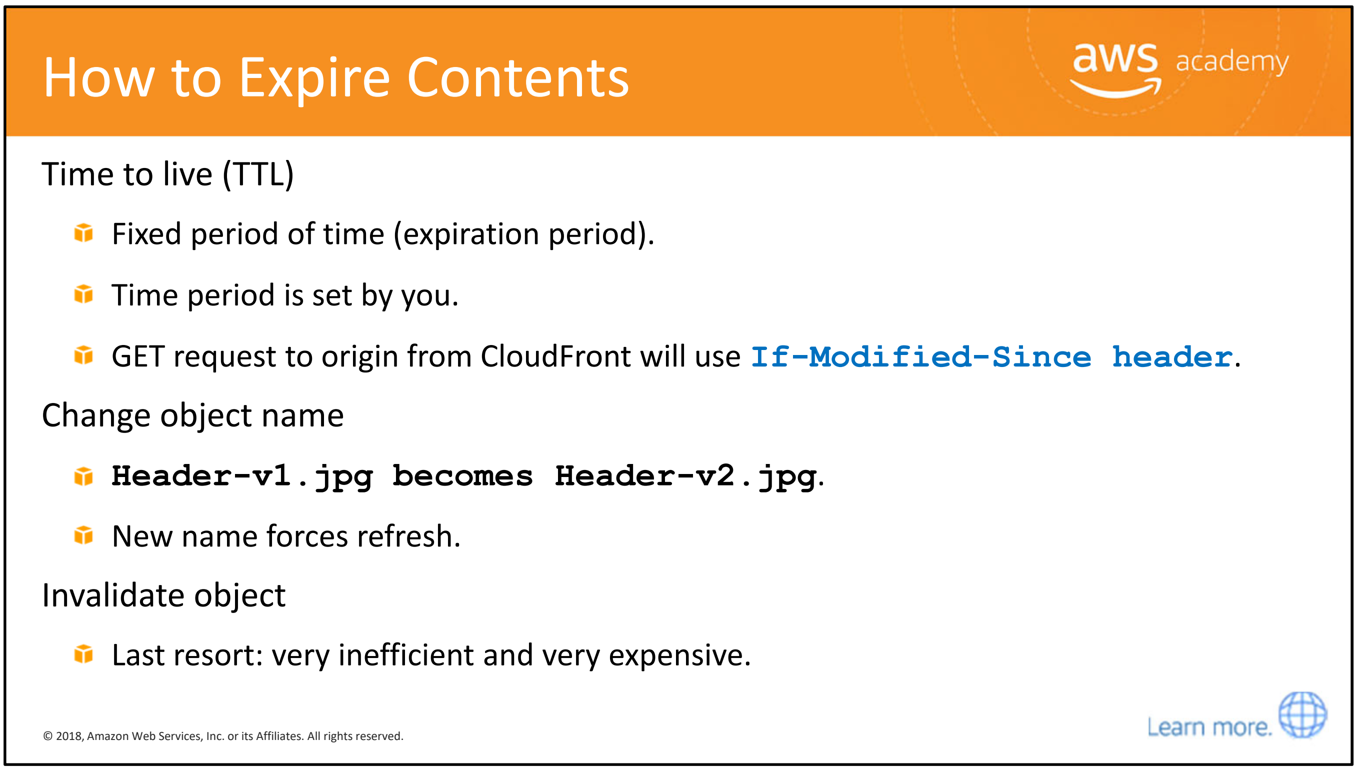
How long is a file kept at the edge location?

- Set **expiration period** by setting the cache control headers on your files in your origin.

What about an expiration period? You can set one. If your files don’t change very often, the best practice is to set a long expiration period, and implement a versioning system to manage updates to your files.

By default, if no cache control header is set, each edge location checks for an updated version of your file when it receives a request more than 24 hours after the previous time it checked the origin server for changes to that file.

How long is a file kept at the edge location?
You can set the expiration period by setting the cache control headers on your files in your origin server.



How to Expire Contents

Time to live (TTL)

- Fixed period of time (expiration period).
- Time period is set by you.
- GET request to origin from CloudFront will use `If-Modified-Since header`.

Change object name

- **`Header-v1.jpg becomes Header-v2.jpg`**.
- New name forces refresh.

Invalidate object

- Last resort: very inefficient and very expensive.

Learn more.

How can you set content to expire? There are three ways to retire cached content:
- Use time to live, or TTL
- Change the object name
- Or invalidate the object

The most preferred options are to use time to live and to change the object name. Using time to live is the easiest option if the replacement does not need to be immediate.

If you set the time to live for a particular origin server to 0, CloudFront will still cache the content from that origin server. It will then make a GET request with an If-Modified-Since header. This header allows the origin server to signal that CloudFront can continue to use the cached content if the content has not changed at the origin server.

Changing the object name requires more effort, but its effects are immediate. There might be some support for this option in some content management systems, or CMSs. Although you can update existing objects in a CloudFront distribution and use the same object names, it is not recommended. CloudFront distributes objects to edge locations only when the objects are requested, not when you put new or updated objects in your origin server. If you update an existing object in your origin server with a newer version that has the same name, an edge location won’t get that new version from your origin server until the object is updated and requested.

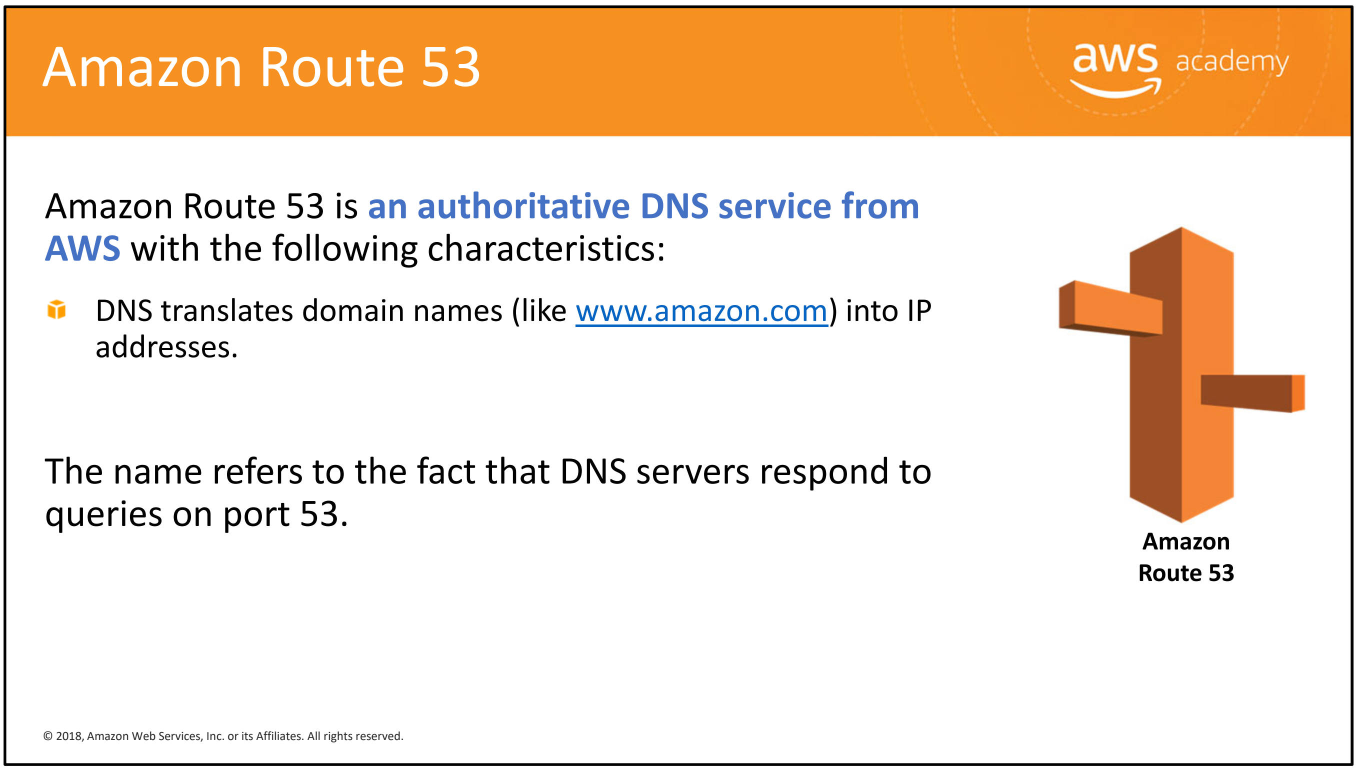
Invalidating an object should be used sparingly for individual objects. It is a bad solution because the system must forcibly interact with all edge locations.

To learn more, go to the following page:
http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/RequestAndResponseBehaviorS3Origin.html

This week – Networks outside the VPC

- Web Caching
  - CloudFront
- **Routing across Regions**
  - **Route53**

14



Amazon Route 53

Amazon Route 53 is **an authoritative DNS service from AWS** with the following characteristics:

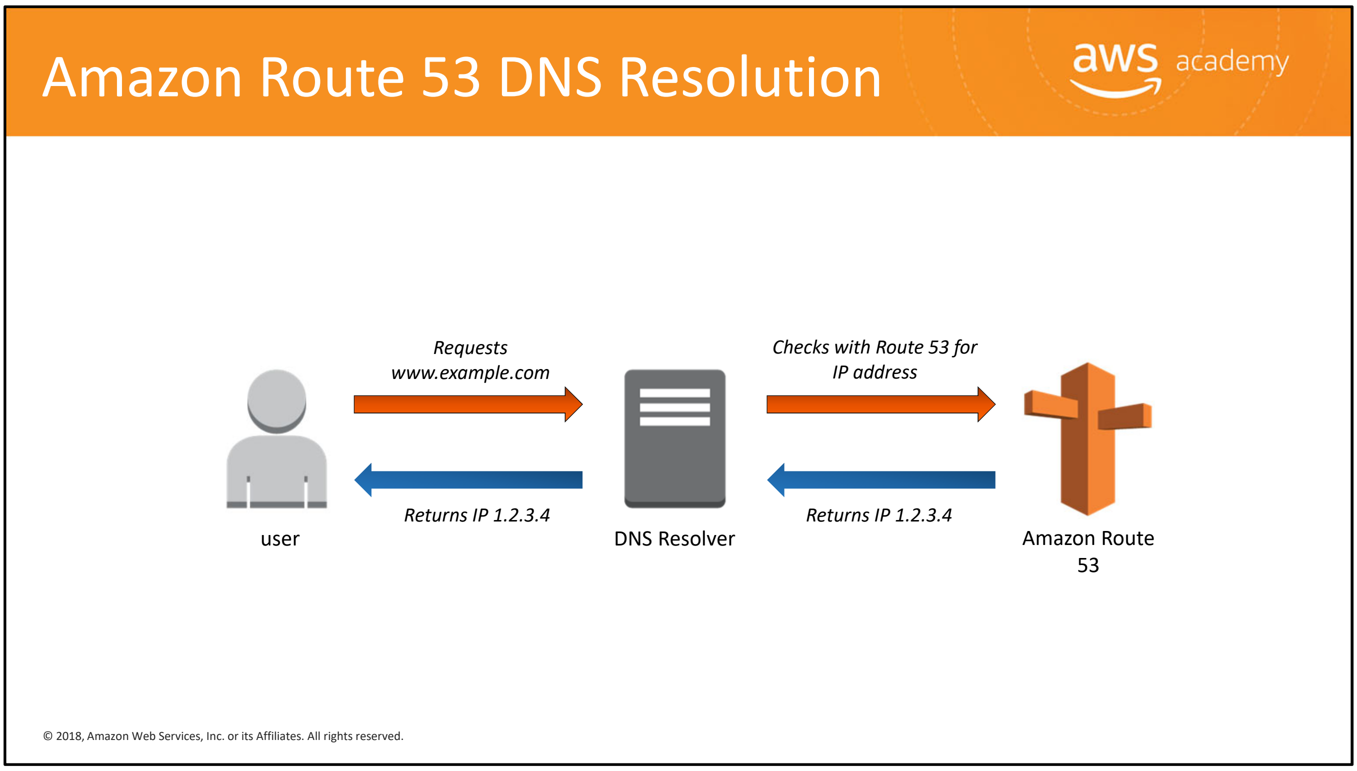
- DNS translates domain names (like www.amazon.com) into IP addresses.

The name refers to the fact that DNS servers respond to queries on port 53.

Amazon Route 53

Elastic load balancers will distribute traffic within a Region. If you want to go outside a Region, you have to use Amazon Route 53. Route 53 is a high availability, authoritative DNS service that translates domain names into IP addresses.

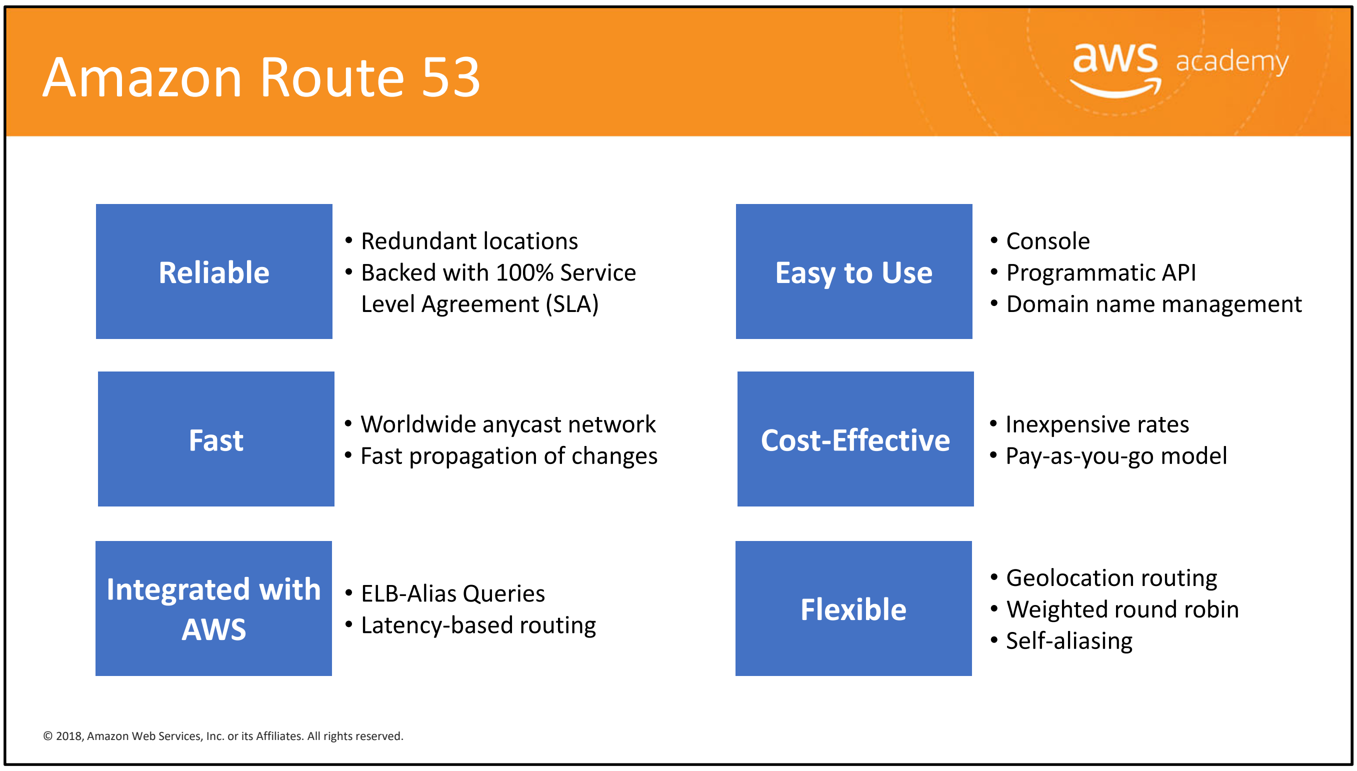
The service is called Route 53 because DNS servers respond to queries on the User Datagram Protocol—or UDP—port 53. Route 53 is one of our only services that has a 100% availability service-level agreement.



Amazon Route 53 DNS Resolution

DNS is how we look up things on the internet.

Here, we see the basic pattern that Route 53 follows when a user initiates a DNS request. When http://example.com is typed in, it goes to a DNS resolver. The DNS resolver checks with your domain in Route 53, gets the IP address of 1.2.3.4, and returns it to the user. That allows the user to get to the website without having to memorize the IP address.



Amazon Route 53

Reliable
- Redundant locations
- Backed with 100% Service Level Agreement (SLA)

Easy to Use
- Console
- Programmatic API
- Domain name management

Fast
- Worldwide anycast network
- Fast propagation of changes

Cost-Effective
- Inexpensive rates
- Pay-as-you-go model

Integrated with AWS
- ELB-Alias Queries
- Latency-based routing

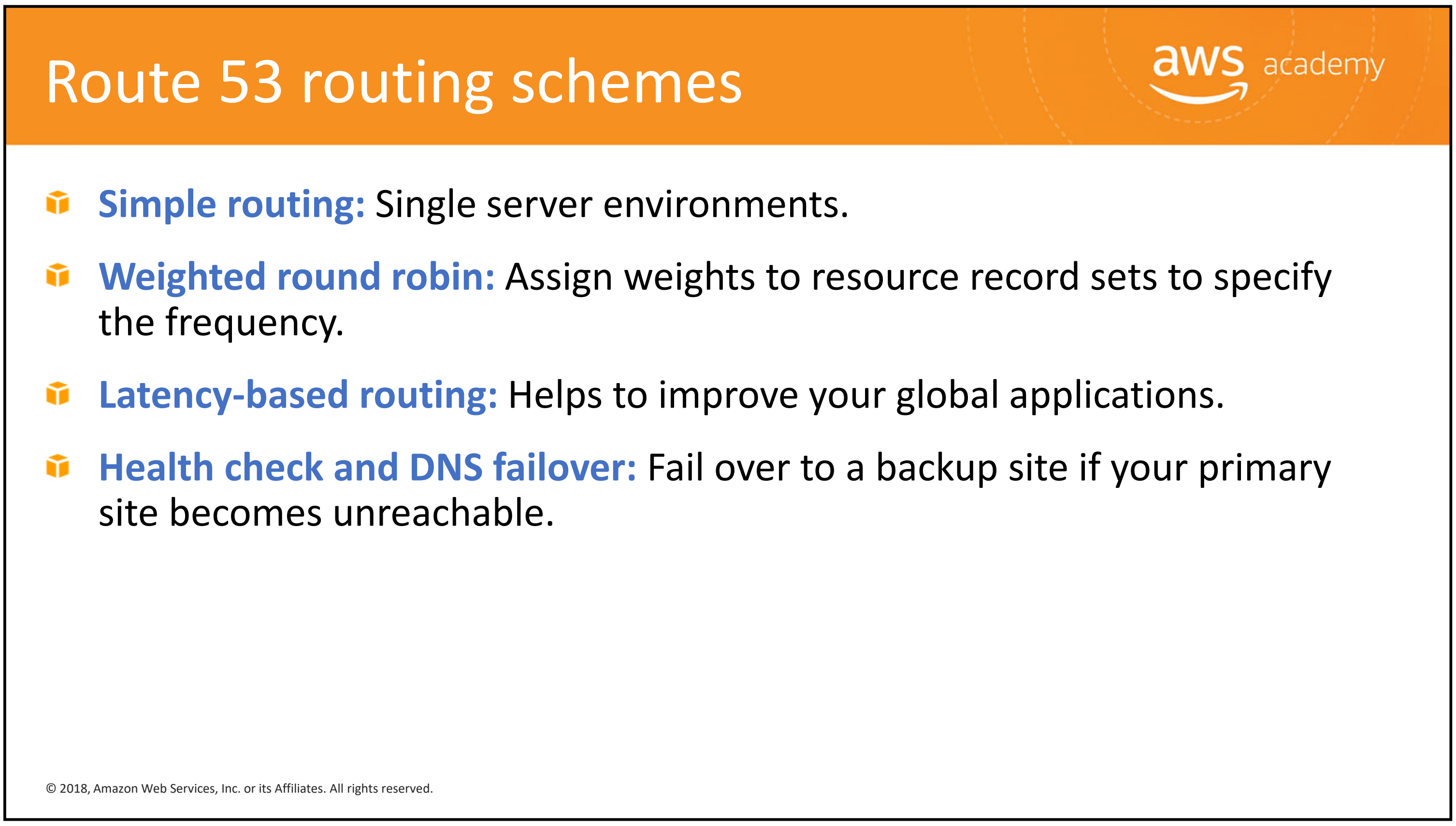
Flexible
- Geolocation routing
- Weighted round robin
- Self-aliasing

Amazon Route 53 is a Tier 0 service, meaning that it is designed and maintained with the highest level of availability in mind.

Route 53 is more than just a DNS service. You can use it to distribute loads across Regions, it has redundant locations backed by 100% SLA, it is fast and utilizes worldwide anycast (which is network addressing and routing methodology in which a single destination address has multiple routing paths), changes are quickly propagated, and it is accessible via the AWS console or programmatic APIs. It is inexpensive with a pay-as-you-go model. You can set up latency-based routing, which means that your customers will be directed to the fastest server based on their latency.

You can also use geolocation rounding, which is helpful if you are in a business where data has to stay in the country of origin. For example, if a customer originates in Germany, their network traffic never leaves Germany.

Weighted round robin—or WRR—is a network scheduling discipline where the number of packets served is in proportion to the assigned weight, and in inverse proportion to the size of the packets. For example, if you're testing a new application, and you only want to send a small amount of that traffic to your new application to see if everything is working properly, you can use WWR.



Route 53 routing schemes

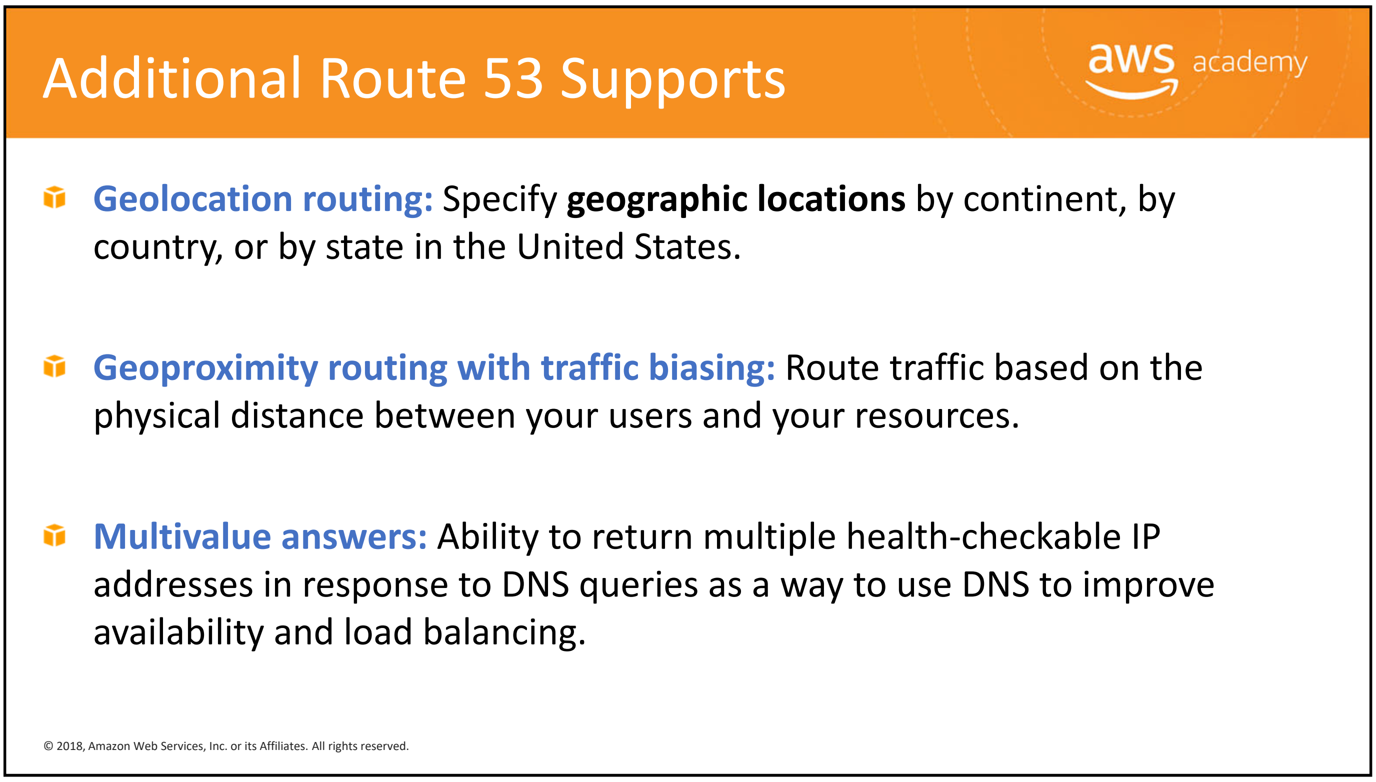
- **Simple routing:** Single server environments.
- **Weighted round robin:** Assign weights to resource record sets to specify the frequency.
- **Latency-based routing:** Helps to improve your global applications.
- **Health check and DNS failover:** Fail over to a backup site if your primary site becomes unreachable.

What kinds of routing does Amazon Route 53 support? Route 53 supports simple routing in a single server environment, and it supports weighted round robin, which assigns the weights according to the desired access frequency. Route 53 also supports latency-based routing, which provides the fastest connection to the application. Finally, it also offers health checks and DNS failover if the primary site becomes unreachable.

Simple routing—or round robin—uses a simple routing policy for when you have a single resource that performs a given function for your domain, such as 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, like the IP address in an A record.

Weighted round robin allows you to assign weights to resource record sets in order to specify the frequency with which different responses are served. You might want to use this capability to do A/B testing, where you send a small portion of traffic to a server where you made a software change. For instance, suppose you have two record sets that are associated with one DNS name: one with weight 3, and one with weight 1. In this case, 75% of the time, Amazon Route 53 will return the record set with weight 3 and 25% of the time Amazon Route 53 will return the record set with weight 1. Weights can be any number between 0 and 255.

Latency-based routing (or LBR) helps you improve your application's performance for a global audience. LBR works by routing your customers to the AWS endpoint (for example Amazon EC2 instances, Elastic IP addresses or load balancers) that provides the fastest experience based on actual performance measurements of the different AWS Regions where your application is running.



Additional Route 53 Supports

- **Geolocation routing:** Specify **geographic locations** by continent, by country, or by state in the United States.
- **Geoproximity routing with traffic biasing:** Route traffic based on the physical distance between your users and your resources.
- **Multivalue answers:** Ability to return multiple health-checkable IP addresses in response to DNS queries as a way to use DNS to improve availability and load balancing.

Additionally, Route 53 supports geolocation and geoproximity routing, which both enable location-based routing. Multivalue answer routing lets you configure Route 53 to return multiple health-checkable IP addresses so you can use DNS to improve availability and load balancing.

Geolocation routing lets you choose the resources that serve your traffic based on the geographic location of your users, or the origin of DNS queries. When you use geolocation routing, you can localize your content and present some or all of your website in the language of your users. You can also use geolocation routing to restrict distribution of content to only the locations where you have distribution rights. Another possible use is for balancing loads across endpoints in a predictable, easy-to-manage way, so that each end-user location is consistently routed to the same endpoint.

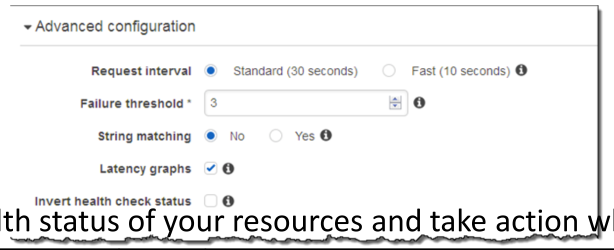
With DNS failover, Amazon Route 53 can help detect an outage of your website and redirect your end users to alternate locations where your application is operating properly. When you enable this feature, Amazon Route 53 health-checking agents will monitor each location or endpoint of your application to determine its availability. You can take advantage of this feature to increase the availability of your customer-facing application.

Geoproximity routing lets you route traffic based on the physical distance between your users and your resources if you're using Route 53 traffic flow. You can also route more or less traffic to each resource by specifying a positive or negative bias. When you create a traffic flow policy, you can specify either an AWS Region, if you're using AWS resources, or the latitude and longitude for each endpoint.

With multiple answers, if you want to route traffic approximately randomly to multiple resources, such as web servers, you can create one multivalue answer record for each resource and, optionally, associate an Amazon Route 53 health check with each record. For example, suppose you manage an HTTP web service with a dozen web servers that each have their own IP address. No one web server could handle all of the traffic, but if you create a dozen multivalue answer records, Amazon Route 53 responds to DNS queries with up to eight healthy records in response to each DNS query. Amazon Route 53 gives different answers to different DNS resolvers. If a web server becomes unavailable after a resolver caches a response, client software can try another IP address in the response.

Amazon Route 53 DNS Failover

- Route 53 tracks health status of resources and take action when error occurs. Configure failover with Amazon Route 53.
- Configure backup and failover scenarios for your own applications.
- Improve health checks by combining multiple health checks, domain name–based health checks, string matching, specifying the request interval, and more.



Advanced configuration

Request interval: [✓ Standard (30 seconds)] Fast (10 seconds)

Failure threshold *: 3

String matching: [✓ No] Yes

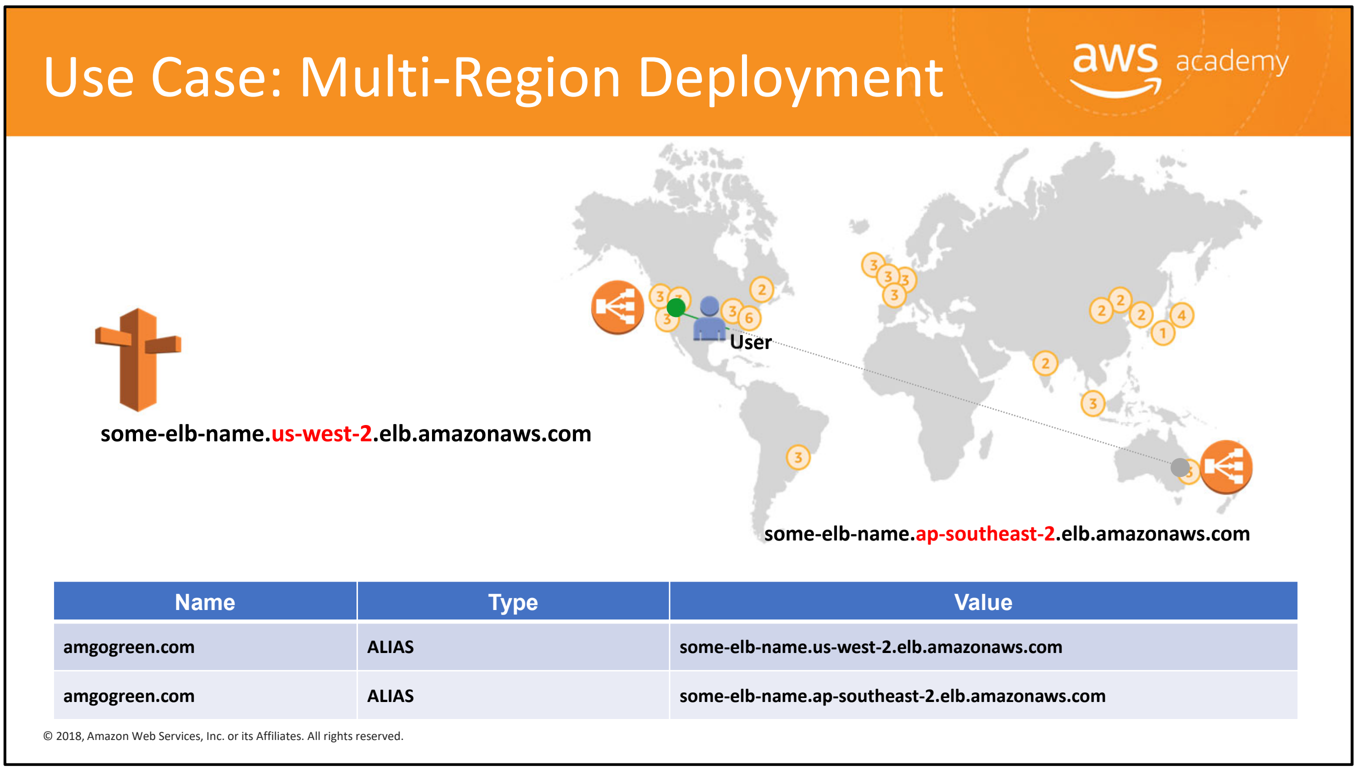
Latency graphs: ✓

Invert health check status: ☐

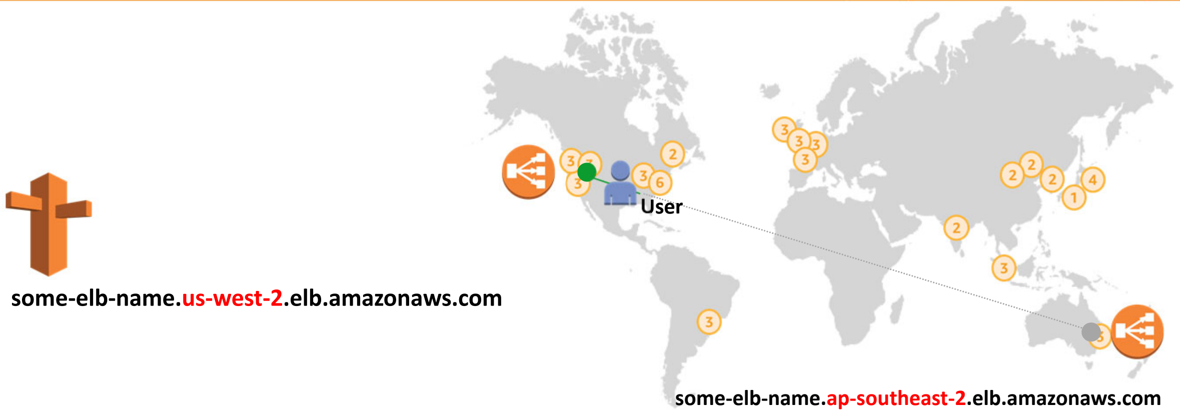
Route 53 lets you track the health status of your resources and take action when an error occurs. It is straightforward to configure failover with Amazon Route 53.

This is a snapshot of what the console would look like if you use Route 53. You can configure backup and failover scenarios for your applications. You can improve health checks by combining multiple health checks, using domain name-based health checks, string matching, and specifying the request interval. The process of configuring these Route 53 settings is minimal.

Note that for string matching, the health check looks for a particular string in the page body, within the first 5 KB of content. You could use string matching to confirm whether the DB instance is working by matching on a string that the page would contain only if it successfully connects to the DB. Failover to the static backup site would then be triggered in the following situations: the web server goes down, the DB instance goes down, if the web server hangs, or if the web server returns invalid content.



Use Case: Multi-Region Deployment


User
some-elb-name.us-west-2.elb.amazonaws.com
some-elb-name.ap-southeast-2.elb.amazonaws.com

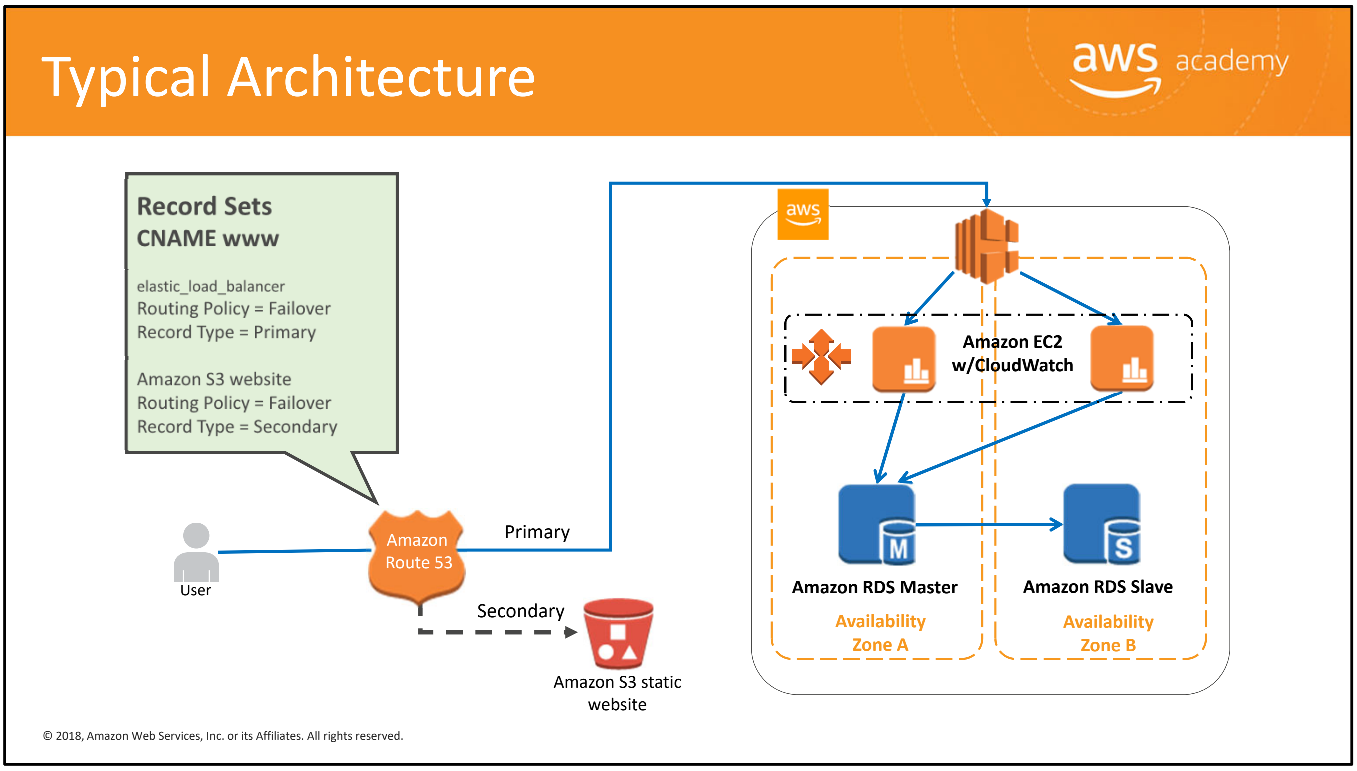
Name	Type	Value
amgogreen.com	ALIAS	some-elb-name.us-west-2.elb.amazonaws.com
amgogreen.com	ALIAS	some-elb-name.ap-southeast-2.elb.amazonaws.com

Here is an example of using a multi-Region deployment. There are two deployments, one in the us-west-2 Region on the west coast of the United States, and one in the Asia Pacific Southeast Region. Both load balancers will respond to http://amgogreen.com . The load balancers will direct the user to the closest http://amgogreen.com.

In this example, the user is automatically directed to the Elastic Load Balancing load balancer that's also in the United States, because it's geographically closer to the user. This situation offers a better user experience. If the west coast site went down, all traffic would automatically be redirected to the Asia Pacific Southeast domain.

Benefits include:

- Latency-based routing to the Region
- And load balancing routing to the Availability Zone



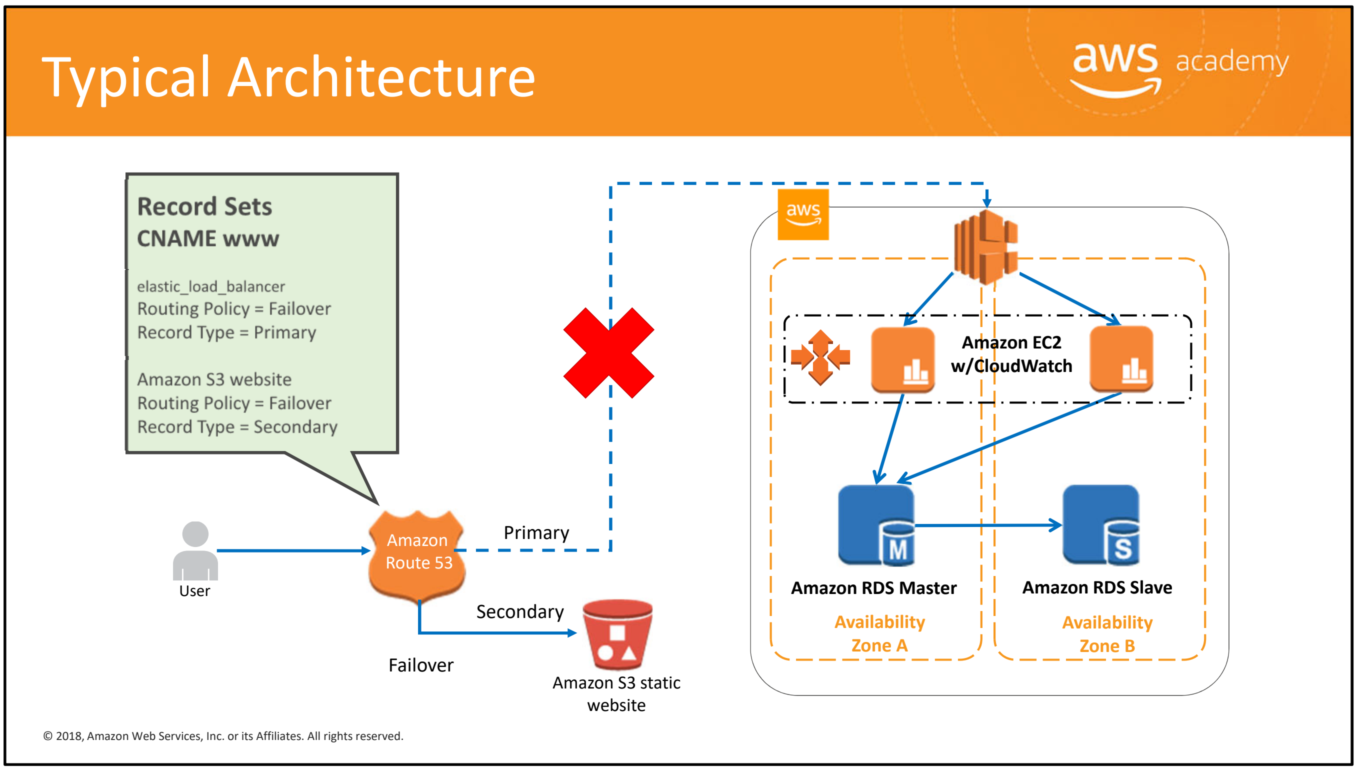
Typical Architecture

This slide shows an example of how DNS failover works in a typical architecture for a multi-tiered web application. Route 53 passes traffic to an ELB load balancer, which then passes the traffic to a fleet of Amazon EC2 instances that use Auto Scaling.

We can do the following tasks with Route 53 to ensure high availability:

First, create two DNS records for the CNAME www with a routing policy of Failover Routing. The first record is the primary, and it points to the ELB load balancer for your web application. The second record is the “Secondary” route policy, and it points to your static Amazon S3 website. If something happened to the primary link, traffic would automatically be redirected to the static website that’s hosted in the Amazon S3 bucket.

Use Route 53 health checks to make sure the primary is up. If the primary is up, all traffic will default to your web application stack.



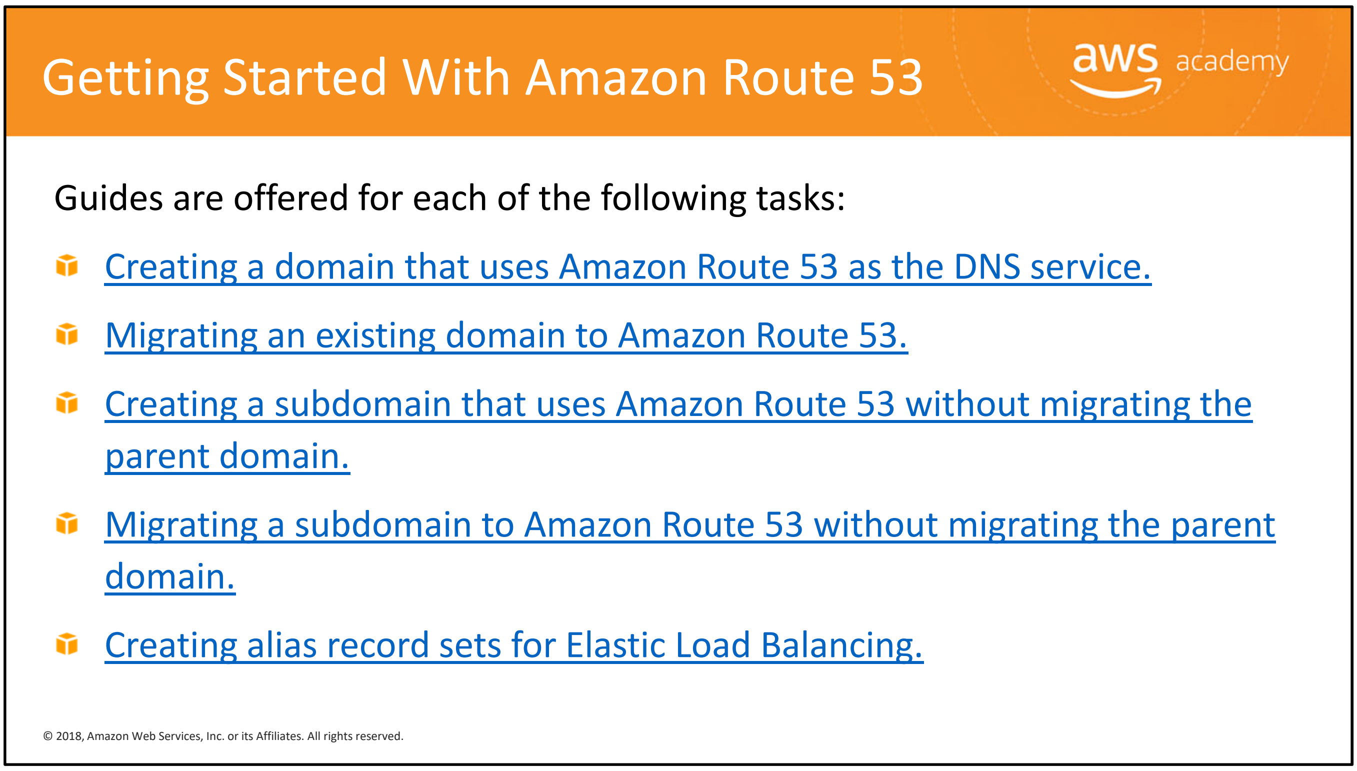
Typical Architecture

This slide shows an example of how DNS failover works in a typical architecture for a multi-tiered web application. Route 53 passes traffic to an ELB load balancer, which then passes the traffic to a fleet of Amazon EC2 instances that use Auto Scaling.

We can do the following tasks with Route 53 to ensure high availability:

First, create two DNS records for the CNAME www with a routing policy of Failover Routing. The first record is the primary, and it points to the ELB load balancer for your web application. The second record is the "Secondary" route policy, and it points to your static Amazon S3 website. If something happened to the primary link, traffic would automatically be redirected to the static website that's hosted in the Amazon S3 bucket.

Use Route 53 health checks to make sure the primary is up. If is the primary is up, all traffic will default to your web application stack.



Getting Started With Amazon Route 53

Guides are offered for each of the following tasks:

- Creating a domain that uses Amazon Route 53 as the DNS service.
- Migrating an existing domain to Amazon Route 53.
- Creating a subdomain that uses Amazon Route 53 without migrating the parent domain.
- Migrating a subdomain to Amazon Route 53 without migrating the parent domain.
- Creating alias record sets for Elastic Load Balancing.

This slide shows some guides on getting started with Amazon Route 53.
Go to each link to learn more. To create a domain that uses Amazon Route 53 as the DNS service select the link. http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/

- To migrate an existing domain to Amazon Route 53 select the link. http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/MigratingDNS.html
- To create a subdomain that uses Amazon Route 53 without migrating the parent domain select the link. http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingNewSubdomain.html
- To migrate a subdomain to Amazon Route 53 without migrating the parent domain select the link. http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/MigratingSubdomain.html
- To create alias record sets for Elastic Load Balancing select the link. http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-choosing-alias-non-alias.html