Route 53

Domain name system

It is a lookup server that help you translate domain names to IP addresses. It is the backbone of the internet.

It uses hierarchical naming structure, start from the root DNS server which contain information about Top Level Domains, which contain information about the Authoritative level domain.

Terminologies

Domain registrar: Amazon Route 53, GoDaddy

DNS Records: A, AAAA, CNAME, NS

Zone file: Contain DNS records

Name server: Resolves DNS queries. They are the server that answers your DNS requests

Top Level Domain: .com, .us, .gov

Authoritative level domain: amazon.com, google.com

How DNS work

When you visit a website say example.com, your web browser will ask your local DNS server managed by your ISP, "do you know what example.com" is, local DNS server say no I don't then it will go out and ask the root DNS server whether it knows example.com.

The root DNS server respond back saying I don't but I do know who manages the .com TLD server and here is the address.
Then local DNS server will use the .com nameserver and ask do you know what "example.com" is, the .com nameserver don't know but it knows the authoritative server example.com's address.
Finally the DNS server will query the example.com authoritative server and get the A record which contains the IP address of the query.

The result will be cached into local DNS server.

Route 53

Highly available, scalable, fully managed and authoritative DNS. You can update the DNS records.

It is also a domain registrar, let you buy domain name.

Records

Define how you want to route traffic for a domain. Each record have a record type, A or CNAME, the value it is mapped to, Time to live, Routing policy.

Route 53 support A, AAAA, CNAME, NS, these are the most important ones

1. A: Maps the domain name to IPv4, example.com = 1.1.1.1
2. AAAA: Maps the domain name to IPv6
3. CNAME: Map a hostname to another hostname, the hostname it is mapped to must have an A or AAAA record.
   Think of CNAME record as alias.
   Can't create CNAME for example.com but for www.example.com
4. NS: Name servers, they are the name servers that can respond to your domain query

Hosted zones

A container for records that define how to route traffic to a domain and subdomain. Basically one hosted zone for your domain and the subdomains that you have.

1. Public hosted zone: Have records that specify how to route traffic on the internet (this is for public domain names)
2. Private hosted zone: Have records tells you how to route traffic in VPC. This is used if you worked for a company, internally they have some reserved URLs that is used to redirect to some internal sites. Public cannot access this.

You pay 0.50 per month per hosted zone

For private hosted name, you can name your EC2 instances an internal domain name like example.com, it will be resolved internally as its name implied and is not for public internet.

Your EC2 instances within a VPC can use route 53 to access the IP address of other EC2 instances in the same hosted name. Remember it is only for those private instances within the hosted zone not for public usage.

TTL

Time to live, the amount of time for client to cache the result of the DNS query. If the TTL is long then less traffic on Route 53 but at the con of having outdated records. if the TTL is short then you will have more traffic on Route 53 because every expired DNS query will need you to fetch it again, but you can change your records easily since it expires often.

Every record have mandatory TTL except for Alias records which you cannot set.

CNAME

CNAME or canonical name is a record to allow a domain name to be used as an alias for another true domain. When a DNS return a CNAME record it will not return that to the client, rather it will look up the true domain name that is returned again and return the A record's IP address.

The chain of CNAME can continue as many as you want, but it is discouraged because it is inefficient.

Simple example would be www.example.com being an alias for example.com. So you would set the CNAME of www.example.com to be example.com, as example.com is the canonical name, the true name for www.example.com.

Limitation: The limitation for CNAME record is that it cannot be used for making a CNAME for root domain. This is because SOA and NS record must be present at the root domain, but CNAME requires that there be no other record be combined with CNAME. Which means that the root domain cannot be used for CNAME. It can be the value of CNAME but it cannot be used as the alias.

Alias

This is specific for Route 53. It allows you to point a hostname to any AWS resources. This will work for root domain and non root domain, doesn't have the limitation of CNAME.

It is free of charge and have native health check.

Example would be setting example.com, having A record to a load balancer's DNS name. The actual IP address will change which is why you would use the DNS name of the load balancer.

You can use example.com to point to the load balancer's DNS name. With CNAME you cannot do that because you are not allowed to use root domain for CNAME records!

Alias type record is always A/AAAA. You cannot set the TTL.

Alias target can be:

- Elastic load balancer
- Amazon CloudFront
- API Gateway
- Elastic Beanstalk
- S3 Website
- VPC Interface endpoints
- Global Accelerator accelerator
- Route 53 record in the same hosted zone

You cannot set alias for an EC2 DNS name!

Routing policies

Define how Route 53 responds to DNS queries. It is only responding to DNS requests! That's all, not load balancer routing!

Simple policy

Route traffic to a single resource.

The value can specify a single value or multiple value, Route 53 will just pick one randomly and return to the client. The example is just a simple A record for a hostname. You can specify one value associated to that hostname, or multiple which Route 53 will pick randomly and return it to the client.

When Alias enabled then you can only specify one AWS resource, you can't do multiple like for values.

Simple policy have no Health checks.

Weighted policy

Control the percentage of requests that go to each specific resource.

Say you can route 30% of the traffic to one EC2 instance, and 70% to another. The DNS records must have the same name and type. Can use health checks with this policy.

The use cases would be load balancing between regions at the DNS level, and test new application versions by routing a small percentage of the traffic to the new application.

Giving a weight of 0 will stop sending traffic to that resource, and if all records have weight of 0 then it will be equally distributed.

Latency policy

Redirect to the resource that has the least latency. Latency is based on traffic between users and AWS regions. People in North America will be redirected to resources in us-east-1 than resources deployedi n ap-southeast-1 since it has lower latency.

You can set records to point to an EC2 instance with latency policy, and you specify the region that the resource is located in, it will then redirect you to the lowest latency resource based on the region that the user is closest to.

Failover policy (active-passive)

We associate the primary health check to the main EC2 instance the health check is a must, and then there is a failover EC2 instance. When the health check detects the first EC2 instance is unhealthy it will route the traffic to the second record to the failover EC2 instance.

There can only be one primary and one secondary resource.

Geolocation policy

Routing is done based on user location not based on latency!

Specify location by continent, country, or by US state. This is used for website localization, restrict content distribution for those country with strict regulations.

Can be associated with health checks.

Ex: German users should go to this IP, France go to this IP, and rest go to this IP.

Geoproximity policy

You can shift more traffic to resources based on the defined bias. More traffic to one resource you increase the bias, less traffic then you decrease the shrink.

The resources can be AWS resources, or Non-AWS resources.

Geoproximity policy is useful when you need to shift traffic from one region to another by increasing the region.

The visual for this is that if you have bias of 0 for two regions, then traffic will be split equally by putting a line in between them. Users on their corresponding side will be routed to that region. However, if you change the bias of that setting with the same setup to say 50 for one region, then the line will be shifted to the other side, incorporating more traffic from the other side because the bias is increased. **It is favored more.**

Multi-value policy

Used when routing traffic to multiple resources, Route 53 will return multiple values/resources, this is different that simple policy multi-value, simple policy doesn't allow health checks while multi-value policy does. It ensures that the resource/value returned is healthy!

You can associate health checks. Up to 8 healthy records can be returned from multi-value query.

It is not a substitute for having an ELB, it is a client side load balancer, with those multi-value query returned, it will be up to the client to pick one healthy query result from it.

Health Check

A way for checking the health only for public resources. This is for when you are deploying into multi-region for high availability and then you let Route 53 route the traffic based on latency or geoproximity. **If one of the region goes down, then your traffic will be rerouted to other region because there is a health check built in (automatic DNS failover)**.

This gives you automated DNS failover. Health check have three types:

1. Health checks that monitor an endpoint (application, server, other AWS resource)
2. Health check that monitor other health checks
3. Health checks that monitor CloudWatch alarms

Health check is an entity that you will be creating. Each health check is backed back about 15 global health checker that will actually be doing the check for healthiness.

Health Check - Monitoring endpoint

There are about 15 global health checkers who will be checking the endpoint for healthiness.

Health cehcker in us-east-1, us-west-1, sa-east-1 and so on. If you set up to monitor an endpoint in eu-west-1, then all of these 15 health checker will ping the endpoint in eu-west-1, and if it receives 200 status then it is healthy.

You can decide the threshold for what is healthy and unhealthy. Default interval is 30 seconds. Supports HTTP, HTTPS, TCP protocol health checks.

Health check wil only with 2xx or 3xx status code. Health check can also be setup based on text in the first 5120 bytes of the response.

You have to configure it to allow Route 53 health checker's traffic in your resource's security group.

Health Check - Calculated Health Check

Combine results of multiple health checks into single health check.

Basically you can set up child health check that monitors their own individual instances. Then you have a master health check that will pool all those child health check's result together and then specify how many of them need to pass to make the parent pass.

This is good for when you are maintaining your website without causing all health checks to fail in say one of the instances.

Health Check - Private resources

Route 53 health checkers live outside of VPC, so they cannot access private endpoints that lives inside VPC.

To do it you have to create a CloudWatch Metric and associate the CloudWatch alarm to the health checker. This is how you can monitor private resources.

Domain registrar vs DNS Service

Domain registrar is the place where you would buy a domain from. They also provide you with DNS service. Namecheap after you purchase the domain name, will let you set DNS record.

However, you do not need to use their domain registrar. It is perfectly valid to purchase your domain name from GoDaddy and then use Route 53 DNS service. You just need to change the NS record of your domain to point to Route 53's DNS server IP. It will then push all the DNS query to Route 53, then you can still use Route 53 for DNS service even though you didn't buy domain from them.

Revision #5
Created 2023-02-14 03:57:35 UTC by Tamarine
Updated 2023-02-27 18:26:49 UTC by Tamarine