Amazon Route 53

What is DNS?

- Domain Name System which translates the human friendly hostnames into the machine IP addresses
- Example: `www.google.com` => `172.217.18.36`
- DNS is the backbone of the Internet
- DNS uses hierarchical naming structure

DNS Hierarchical Structure (example)

- `.com`
- `example.com`
- `www.example.com`
- `api.example.com`

DNS Terminologies

- **Domain Registrar**: Amazon Route 53, GoDaddy, ...
- **DNS Records**: A, AAAA, CNAME, NS, ...
- **Zone File**: contains DNS records
- **Name Server**: resolves DNS queries (Authoritative or Non-Authoritative)
- **Top Level Domain (TLD)**: `.com` , `.us` , `.in` , `.gov` , `.org` , ...
- **Second Level Domain (SLD)**: `amazon.com` , `google.com` , ...

DNS Structure Example

Given the URL `http://api.www.example.com` , here's the breakdown:

- **Protocol**: `http`
- **Subdomain**: `api.www`
- **Second Level Domain (SLD)**: `example`
- **Top Level Domain (TLD)**: `.com`
- **Root**: the invisible `.` at the end of the domain
- **FQDN (Fully Qualified Domain Name)**: `api.www.example.com.`

How DNS Works

1. **Web Browser**: The user tries to access `example.com` .
    - The browser first checks its own DNS cache (based on TTL - Time To Live).
    - If not cached, it queries the **Local DNS Server**.
2. **Local DNS Server**:
    - This server is managed by the user's company or dynamically assigned by their ISP.
    - If it doesn't have the answer cached, it starts a recursive query:
3. **Root DNS Server**:
    - Responds with the name servers for the `.com` TLD.
    - Managed by **ICANN**.
4. **TLD DNS Server** (`.com`):

    - Responds with the name servers for `example.com`.
    - Managed by **IANA** (a branch of ICANN).
5. **SLD DNS Server** (`example.com`):
    - Responds with the IP address for `example.com` (e.g., `9.10.11.12`).
    - Managed by the domain registrar (e.g., Amazon Registrar, Inc.).
6. **Final Response**:
    - The Local DNS Server caches the result and returns the IP address to the browser.
    - The browser then contacts the **Web Server** directly using the resolved IP.

Notes

- Each DNS server in the hierarchy helps narrow down the search.
- The entire process is optimized by DNS caching to avoid redundant lookups.
- TTL determines how long each DNS record is stored in cache.

Amazon Route 53

- A highly available, scalable, fully managed and **authoritative** DNS service.
    - **Authoritative** means the customer (you) can update the DNS records.
- Route 53 is also a **Domain Registrar**.
- Provides the **ability to check the health** of your resources (e.g., via health checks).
- The only AWS service that offers a **100% availability SLA**.
- The name **Route 53** refers to **port 53**, the standard port for DNS.

Example

- A client sends a DNS query to Route 53 for `example.com`.
- Route 53 responds with the public IP address, e.g., `54.22.33.44`.
- The client then connects to the resource (e.g., an EC2 instance) in the **AWS Cloud** using that IP.

Route 53 – Records

- Used to define how you want to **route traffic** for a domain.

Each DNS record contains:

- **Domain/subdomain Name** – e.g., `example.com`
- **Record Type** – e.g., `A`, `AAAA`
- **Value** – e.g., `12.34.56.78`
- **Routing Policy** – defines how Route 53 responds to DNS queries
- **TTL** (Time To Live) – how long the record is cached by DNS resolvers

Route 53 supports the following DNS record types:

- **Must know**: `A`, `AAAA`, `CNAME`, `NS`
- **Advanced**: `CAA`, `DS`, `MX`, `NAPTR`, `PTR`, `SOA`, `TXT`, `SPF`, `SRV`

Route 53 – Record Types

- **A** – maps a hostname to an IPv4 address.
- **AAAA** – maps a hostname to an IPv6 address.

- **CNAME** – maps a hostname to another hostname.
  - The target must be a domain name that resolves via an `A` or `AAAA` record.
  - You **cannot create** a `CNAME` record for the top node of a DNS namespace (called **Zone Apex**).
  - Example: you **can't create** a CNAME for `example.com`, but you **can** for `www.example.com`.
- **NS (Name Server)** – defines the name servers for the hosted zone.
  - Controls how DNS traffic is routed for a domain.

Route 53 – Hosted Zones

- A **hosted zone** is a container for records that define how to route traffic to a domain and its subdomains.

Types of Hosted Zones

- **Public Hosted Zones**:
  - Used to route traffic on the Internet (for public domain names).
  - Example: `application1.mypublicdomain.com`
- **Private Hosted Zones**:
  - Used to route traffic within one or more VPCs (for private domain names).
  - Example: `application1.company.internal`

Pricing

- You pay **$0.50 per month** per hosted zone.

Route 53 – Public vs. Private Hosted Zones

Public Hosted Zone

- Used to route DNS traffic **on the public Internet**.
- Example: A client queries `example.com` and receives a public IP like `54.22.33.44`.
- Resources behind a public hosted zone include:
  - **S3 Bucket**
  - **Amazon CloudFront**
  - **EC2 Instance** (with **Public IP**)
  - **Application Load Balancer**

Private Hosted Zone

- Used to route DNS traffic **within a VPC** (private domain resolution).
- Resources use **Private IP addresses** and are only accessible inside the VPC.
- Example queries:
  - `webapp.example.internal` → resolves to EC2 instance `10.0.0.10`
  - `api.example.internal` → resolves to EC2 instance `10.0.0.35`
  - `db.example.internal` → resolves to Amazon RDS DB instance with private IP
- These names are resolved **only from inside the VPC**.

Key Difference

- **Public Hosted Zone**: for Internet-facing DNS names.
- **Private Hosted Zone**: for internal VPC-level DNS resolution.

Route 53 – Records TTL (Time To Live)

High TTL (e.g., 24 hours)

- Reduces traffic on Route 53
- May serve outdated records if IP changes
- Better for stable resources

Low TTL (e.g., 60 seconds)

- Increases traffic on Route 53 (may increase cost)
- Records stay up-to-date more frequently
- Easier to apply changes to records quickly

Additional Notes

- **TTL is mandatory** for every DNS record, **except for Alias records**
- The client caches the DNS result for the **duration of the TTL**
- Short TTL = more flexibility
- Long TTL = more efficiency

CNAME vs Alias

Context

AWS Resources (e.g., Load Balancer, CloudFront) expose an AWS hostname like:

- `lb1-1234.us-east-2.elb.amazonaws.com`

You want to access it using your custom domain, e.g.:

- `myapp.mydomain.com`

CNAME

- Points a hostname to **any other hostname**
  Example: `app.mydomain.com` → `blabla.anything.com`
- Valid **only for NON-ROOT DOMAINS** (e.g., `something.mydomain.com`)
- Cannot be used at the **zone apex** (i.e., `mydomain.com`)

Alias

- Points a hostname to an **AWS Resource**
  Example: `app.mydomain.com` → `blabla.amazonaws.com`
- Works for **ROOT and NON-ROOT DOMAINS** (e.g., `mydomain.com` and `something.mydomain.com`)
- Free of charge
- Supports native health checks

Route 53 – Alias Records

- Maps a hostname to an **AWS resource** (e.g., Load Balancer, CloudFront).
- It's an **extension to DNS functionality** provided by AWS.
- **Automatically recognizes** changes in the resource's IP addresses.
- Unlike `CNAME`, it **can be used at the Zone Apex** (e.g., `example.com`).
- Alias Records are always of type **A (IPv4)** or **AAAA (IPv6)**.

- **You can't set the TTL** manually — it's managed by AWS.

Example

Record Name	Type	Value
example.com	A	MyALB-123456789.us-east-1.elb.amazonaws.com

- Points to an AWS-managed resource like an **Application Load Balancer**.
- IP address may change, but the Alias record stays valid.

Route 53 – Alias Records Targets

Alias records in Route 53 can be used to point to the following AWS resources:

- **Elastic Load Balancers**
- **CloudFront Distributions**
- **API Gateway**
- **Elastic Beanstalk environments**
- **S3 Websites**
- **VPC Interface Endpoints**
- **Global Accelerator accelerators**
- **Route 53 record in the same hosted zone**

Important Notes

- You **cannot** set an **Alias record for an EC2 DNS name**

Route 53 – Routing Policies

- Define how **Route 53 responds to DNS queries**.
- Don't confuse "routing" with load balancer routing:
  - Route 53 does **not route traffic** — it only resolves DNS queries.

Supported Routing Policies in Route 53

- **Simple** – returns a single resource.
- **Weighted** – split traffic based on weights.
- **Failover** – primary/secondary DNS setup for high availability.
- **Latency based** – return the lowest-latency endpoint for the user.
- **Geolocation** – respond based on the user's geographic location.
- **Multi-Value Answer** – return multiple values to allow client-side load balancing.
- **Geoproximity** – respond based on user and resource proximity (requires **Traffic Flow** feature).

Routing Policies – Simple

- Typically used to route traffic to a **single resource**.
- You can specify **multiple values** in the same record.
- If multiple values are returned, the **client randomly selects one**.
- When using **Alias**, you can specify **only one AWS resource**.
- **Health Checks are not supported** with this policy.

Behavior

- **Single Value** example:

    - `foo.example.com` → `A 11.22.33.44`
- **Multiple Values** example:
    - `foo.example.com` →
        - `A 11.22.33.44`
        - `A 55.66.77.88`
        - `A 99.11.22.33`
    - The client picks one **randomly**.

Use Case

- Best for **basic DNS resolution** without failover or traffic control logic.

Routing Policies – Weighted

- Used to **control the percentage of requests** that go to each specific resource.
- Each DNS record is assigned a **relative weight**.

Traffic Calculation Formula

traffic (%) = (Weight for a specific record) / (Sum of all weights)

- The weights do **not** need to sum up to 100.
- Example: if weights are 70, 20, 10 → traffic is split 70%, 20%, 10%.

Requirements

- All DNS records must have the **same name and type**.
- Can be associated with **Health Checks**.

Use Cases

- Load balancing between multiple regions.
- Gradual rollout or A/B testing of new application versions.
- Staging vs production environments.

Special Cases

- Setting a **weight of 0** excludes a record from traffic routing (can be used to disable a record).
- If **all records have weight 0**, then **all records are returned equally**.

Routing Policies – Latency-based

- Redirects users to the **resource with the lowest latency**.
- Ideal when **latency is a priority** for your application.

How It Works

- Latency is measured **between the user and AWS Regions**.
- A user in Germany may be routed to a US-based resource if that region provides **lower latency**.
- The goal is to improve performance by minimizing network delay.

Additional Info

- Can be combined with **Health Checks** to support failover.
- Works well for **globally distributed applications**.

Route 53 – Health Checks

- **HTTP Health Checks** are available **only for public resources**.
- Health checks enable **automated DNS failover**.

Types of Health Checks

1. **Endpoint Monitoring**
   - Monitor an application, server, or other AWS resource via HTTP/HTTPS/TCP.
2. **Calculated Health Checks**
   - Combine the status of multiple health checks (logical AND/OR).
3. **CloudWatch Alarm Monitoring**
   - Trigger health status based on custom CloudWatch alarms.
   - Supports checks on DynamoDB throttles, RDS alarms, custom metrics.
   - Useful for **private resources** where direct health check is not possible.

Additional Notes

- Health checks are integrated with **CloudWatch metrics**.
- Route 53 uses health check status to decide how to answer DNS queries for failover-capable routing policies (e.g., latency, failover, geoproximity).

Health Checks – Monitor an Endpoint

- About **15 global health checkers** evaluate the health of your endpoint.

Key Settings

- **Healthy/Unhealthy Threshold**: default is 3 consecutive checks
- **Interval**: 30 seconds (can be set to 10 sec – higher cost)
- **Supported Protocols**: HTTP, HTTPS, TCP

Health Evaluation Logic

- If **more than 18%** of health checkers report the endpoint as healthy, Route 53 considers it **Healthy**.
- Otherwise, the endpoint is considered **Unhealthy**.

Response Conditions

- Health checks pass only if the endpoint returns **2xx or 3xx** status codes.
- Checks can also inspect the **first 5120 bytes** of the response body to validate custom logic.

Other Considerations

- You can choose **specific locations** for Route 53 to run health checks from.
- Ensure your firewall/router allows **incoming requests from Route 53 health check IPs**:
   - IP ranges available at: https://ip-ranges.amazonaws.com/ip-ranges.json

Route 53 – Calculated Health Checks

- Combine the results of **multiple health checks** into a **single aggregated health check**.
- Use logical operators: **OR**, **AND**, **NOT**.
- You can monitor up to **256 child health checks**.

Customization Options

- Define how many child checks must pass for the **parent check** to be considered healthy.

- Flexible threshold: useful when only a subset of services must be up.

Use Case

- Allows you to perform **maintenance on one part of your system** without marking the entire service as unhealthy.
- Helps avoid full failover during **partial outages** or **maintenance windows**.

Health Checks – Private Hosted Zones

- Route 53 health checkers operate **outside of the VPC**.
- Therefore, they **cannot access private endpoints** (e.g., resources in a private VPC or on-premises).

Workaround with CloudWatch

- You can create a **CloudWatch Metric** based on custom monitoring logic.
- Then associate it with a **CloudWatch Alarm**.
- Finally, create a **Route 53 Health Check** that monitors the status of the **CloudWatch Alarm**.

Use Case

- This approach allows DNS failover for **private resources** that Route 53 cannot reach directly.

Routing Policies – Failover (Active-Passive)

- Used to implement **active-passive failover** between two resources (e.g., EC2 instances).
- Route 53 serves the **primary** resource as long as it is healthy.
- If the **primary** resource fails the **health check**, Route 53 automatically routes traffic to the **secondary** resource.
- **Health Check is mandatory** for failover to function.
- Typical use case: **Disaster Recovery** setup.

Flow Summary

1. Client sends a DNS request.
2. Route 53 evaluates the health check.
3. If **healthy**, it returns the IP of the **primary** instance.
4. If **unhealthy**, it returns the IP of the **secondary** instance.

Routing Policies – Geolocation

- **Different from Latency-based!**
- This routing is **based on user location**.
- You can specify location by:
  - Continent
  - Country
  - US State
- If there's overlapping, the **most precise** location is selected.
- It's recommended to create a **"Default" record** (used when no match on location is found).
- Use cases:
  - Website localization
  - Restrict content distribution
  - Load balancing
- Can be associated with **Health Checks**

Routing Policies – Geoproximity

- Route traffic to your resources based on the geographic location of users and resources.
- **Ability to shift more traffic to resources** based on the defined **bias**.
- To change the size of the geographic region, specify **bias** values:
  - To expand (1 to 99): more traffic to the resource.
  - To shrink (-1 to -99): less traffic to the resource.
- Resources can be:
  - AWS resources (specify AWS region)
  - Non-AWS resources (specify Latitude and Longitude)
- You must use **Route 53 Traffic Flow** to use this feature.

Routing Policies – IP-based Routing

- Routing is based on clients' IP addresses.
- You provide a list of CIDRs for your clients and the corresponding endpoints/locations (user-IP-to-endpoint mappings).
- Use cases:
  - Optimize performance
  - Reduce network costs
- Example: route end users from a particular ISP to a specific endpoint.

CIDR Collection Example:

Locations	CIDR blocks
location-1	203.0.113.0/24
location-2	200.5.4.0/24

Record Mapping Example:

Record Name	Value	IP-based
example.com	1.2.3.4	location-1
example.com	5.6.7.8	location-2

Routing Policies – Multi-Value

- Use when routing traffic to multiple resources.
- Route 53 returns multiple values/resources.
- Can be associated with Health Checks (returns only values for healthy resources).
- Up to 8 healthy records are returned for each Multi-Value query.
- **Multi-Value is not a substitute for having an ELB.**

Example Records Table

Name	Type	Value	TTL	Set ID	Health Check
www.example.com	A Record	192.0.2.2	60	Web1	A

www.example.com	A Record	198.51.100.2	60	Web2	B
www.example.com	A Record	203.0.113.2	60	Web3	C

Domain Registrar vs. DNS Service

- You buy or register your domain name with a Domain Registrar, typically by paying annual charges (e.g., GoDaddy, Amazon Registrar Inc., etc.)
- The Domain Registrar usually provides you with a DNS service to manage your DNS records
- However, you can use another DNS service to manage your DNS records
- **Example**: purchase the domain from GoDaddy and use Amazon Route 53 to manage your DNS records

Diagram explanation

The user purchases the domain `example.com` from GoDaddy (acting as the domain registrar).
Then, instead of using GoDaddy's DNS service, the user configures Amazon Route 53 to manage the DNS records associated with the domain.

GoDaddy as Registrar & Route 53 as DNS Service

Scenario

- The domain is registered with **GoDaddy** (acts as the Domain Registrar).
- The DNS service is managed by **Amazon Route 53** (not by GoDaddy).

Key Points

- In GoDaddy, the DNS records section displays a message indicating that DNS information cannot be shown because custom nameservers are used.
- The user has configured **custom nameservers** in GoDaddy to point to Route 53 name servers.

Example

- **Domain**: stephanetheteacher.com
- **GoDaddy** is used to register the domain.
- **Route 53** hosts the public DNS zone for the domain.
- The user updates the nameservers in GoDaddy to the ones provided by Route 53, such as:
  - `ns-252.awsdns-31.com`
  - `ns-1468.awsdns-55.org`
  - `ns-633.awsdns-15.net`
  - `ns-1800.awsdns-33.co.uk`

How it works

1. Go to Route 53 and create a **Public Hosted Zone** for your domain.
2. Route 53 provides four **nameservers**.
3. Copy these nameservers and **update them** in the GoDaddy dashboard under the domain settings.
4. Once the nameservers are updated and propagated, Route 53 takes over DNS management for the domain.

3rd Party Registrar with Amazon Route 53

If you buy your domain from a **third-party registrar**, you can still use **Amazon Route 53** as your DNS service provider.

Steps to Configure

1. Create a **Hosted Zone** in Route 53
2. Update **NS Records** on the third-party registrar's website to use the **Route 53 Name Servers**

Key Points

- **Domain Registrar ≠ DNS Service**
- However, most domain registrars come with basic DNS features by default