Contents

[1 Global Infrastructure 2](#_Toc69569131)

[2 Identity Access Management 2](#_Toc69569132)

[3 EC2 2](#_Toc69569133)

[3.1 E2 Amazon Machine Image (AMI) (Important from exam perspective) 2](#_Toc69569134)

[3.2 Key Value Pair 3](#_Toc69569135)

[3.3 EC2 Instance Connect 3](#_Toc69569136)

[3.4 Security Group 3](#_Toc69569137)

[3.5 Install Apache on AWS instance 4](#_Toc69569138)

[3.6 EC2 User Data 4](#_Toc69569139)

[3.7 EC2 Launch Types 5](#_Toc69569140)

[3.7.1 On Demand 5](#_Toc69569141)

[3.7.2 Reserved Instances 5](#_Toc69569142)

[3.7.3 Spot Instances 5](#_Toc69569143)

[3.7.4 Dedicated Instances 5](#_Toc69569144)

[3.7.5 Dedicated Hosts 6](#_Toc69569145)

[3.8 EC2 Instance Types 6](#_Toc69569146)

[3.9 Placement Groups 7](#_Toc69569147)

[3.9.1 Cluster 7](#_Toc69569148)

[3.9.2 Spread 7](#_Toc69569149)

[3.9.3 Partition 7](#_Toc69569150)

[4 Scalability and High Availability 7](#_Toc69569151)

[4.1 Scalability 7](#_Toc69569152)

[4.1.1 Vertical Scalability 7](#_Toc69569153)

[4.1.2 Horizontal Scalability 7](#_Toc69569154)

[4.2 High Availability 8](#_Toc69569155)

[5 Elastic Load Balancing (ELB) 8](#_Toc69569156)

[5.1 Target Groups 8](#_Toc69569157)

[5.2 Classic Load Balancer (CLB) 9](#_Toc69569158)

[5.3 Application Load Balancer (ALB) 9](#_Toc69569159)

[5.4 Network Load Balancer (NLB) 10](#_Toc69569160)

[5.5 Auto Scaling Group 10](#_Toc69569161)

[5.5.1 Auto Scaling Alarms 11](#_Toc69569162)

[5.6 Health Check 12](#_Toc69569163)

[6 EBS 12](#_Toc69569164)

[7 AWS Lambda 12](#_Toc69569165)

# Global Infrastructure

Region (Physical Location contains one or more availability centers) 🡪Availability Center (One or more data centers)

-🡪 Data Center (computers, switches etc.)

Edge Location (Smaller than availability center) (cache center to cache content. Cloud Front is used (CDN) provided by AWS).

# Identity Access Management

1. Users
2. Groups
3. Policies
4. Roles

Secure the root account with multi factor authentication

# EC2

* Renting virtual machines (EC2)
* Storing Data (EBS)
* Load Balancing (ELB)
* Scaling services using an auto scaling Group (ASG)
* Payment mode is per second

## E2 Amazon Machine Image (AMI) (Important from exam perspective)

(AMI stands for, an operating system used to launch the VM)

1. **Amazon Linux**
2. **Amazon Linux 2**
3. **Windows**
4. **Ubuntu**

**Instance Type** refers to the capability of machine (RAM, Processor etc.) **t2.micro is free**

You can develop custom AMI for your needs that will allow you to install pre-configured software etc. **AMI are locked to a specific region.** However you can copy them to different regions. And they are stored in **S3.**

## Key Value Pair

A key pair consists of a public key that AWS stores, and a private key file that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

## EC2 Instance Connect

An online browser interface from AWS to connect to EC2 instance.

**SSH**

**SSH Utility works on (Linux, Mac, Windows 10)**

**Puty works on (Linux and Window <=10)**

To enter a machine through SSH use command

ssh -i ec2tutorial.pem ec2-user@ec2-54-221-25-226.compute-1.amazonaws.com

**Exam Question**

Permission 0644 for “sshfile.pem” are too open because private keys are accessible

To Fix this problem run the **chmod 0400** “sshfile.pem”

**In order for EC2 instance to be accessible from the E2 Connect, SSH port should be enable (for example 22)**

## Security Group

A security group acts as a virtual firewall for your EC2 instances to control **incoming** and **outgoing** traffic. Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance. ... If you don't specify a security group, Amazon EC2 uses the default security group.

* Single group can be attached to multiple EC2 instances
* An instance can have multiple security groups
* Security groups are tied to region
* Connection **timeout error** is related to security group while **connection refused** is related to application/instance.
* Inbound traffic is **blocked** by default
* Outbound traffic is **opened** by default.
* Security Groups are region locked

**Referencing security Groups from other security groups**

A security group can reference other security groups; this will allow other instances to connect to first instance by just attaching the referenced security group to the instance.

**For example, Instance E1 has referenced SG1 and SG2 for inbound, so we attach SG1 to E2, then E2 can connect to E1 directly, without any further configuration.**

**Elastic IP**

An elastic IP is attached to your account and it will allow public access, need to buy it I think. However use of elastic IP is not recommended and reflects poor architecture. One should use load balancer or Route53 for DNS for better management and scalable.

## Install Apache on AWS instance

* Yum update –y
* Yum install –y httpd.x86\_64
* systemctl start httpd.service
* systemctl enable httpd.service
* curl localhost:80

Enable the port 80 from the security group.

echo “Hello World “ > /var/www/html/index.html

echo “Hello World from $(hostname -f)“ > /var/www/html/index.html

## EC2 User Data

**EC2 user data will be run with the root user.**

#!/bin/bash

# install httpd(Linux version 2)

yum update -y

yum install -y httpd.x86\_64

systemctl start httpd.service

systemctl enable http.service

echo "Hello World from $(hostname -f)"" > /var/www/html/index.html

## EC2 Launch Types

### On Demand

Predictable price, short workload. Costly, and billing start from the first minute after instance is up. Billing is calculated on per second basis

### Reserved Instances

1. **Reserve instance** (long workloads)

* Up to 75% discount compared to on demand
* Payment is upfront
* Long term commitment
* Recommended for steady state usage (like database)

1. **Convertible reserved instance** (long workloads with option to switch machine/hardware)

* Flexible
* Up to 54 % discount

1. **Scheduled Reserved Instance** ( get an instance on Sunday from 6 to 9 )

* Launch within time window you reserve
* when you require a fraction day/week/month

### Spot Instances

For short workloads, cheap, but can be lost (less reliable). Mostly used for batch jobs, data analysis or workload that are resilient to failures. Not recommended for critical jobs or databases

* Up to 90% discount as compared to on demand
* Define a **max spot price**, then you will get an instance once the **current spot price** < max price set by you.
* The hourly spot price varies based on offer and capacity
* If the current spot price gets larger than max price set by you, AWS will notify you and you have 2 minutes to handle, after 2 minutes AWS will own the instance.
* **Spot Block Strategy,** this will allow you to block an instance for a specific time frame (1 to 6 hours), so that no interruptions will be done, but you have to pay for instance upfront. (In rare cases instance may be reclaimed)

### Dedicated Instances

Hardware will not be shared, and will be dedicated to single customer.

* Instances will be running on hardware dedicated to your account
* Instances may share hardware but only for the instances under your account
* No control over instance placement (only automatic placement)
* No visibility of sockets, cores and Host Id

### Dedicated Hosts

Book an entire physical server and control the instance placement, Visibility to underlying sockets /physical cores of the hardware. Allocated to your account for a period of 3 years. **More expensive**.

* Useful when you have complicated licensing model
* Also applicable where strong compliance rules exist for dedicated hardware

**Price Comparison for m4.large Instance**

|  |  |
| --- | --- |
| **Launch Type** | **Price (per hour)** |
| On demand | $0.10 |
| Spot Instance (Spot Price) | $0.03 to 0.045 up to **70% off compared to on demand)** |
| Spot Block (1 to 6 hours) | ~ Spot Price |
| Reserved Instance (12 months) no upfront | $0.062 |
| Reserved Instance (12 months) all upfront | $0.058 (**around 50% off compared to on demand**) |
| Reserved Instance (36 months) no upfront | $0.043 |
| Reserved **Convertible** Instance (12 months) no upfront | $0.071 |
| Reserved **Dedicated** Instance (12 months) all upfront | $0.064 |
| Reserved **Scheduled** Instance (recurring schedule on 12 months term) | $0.9 -$0.95 (5%- 10% discount) |
| Dedicated Host | On demand price |
| Dedicated Host Reservation | Up to 70% off |

## EC2 Instance Types

1. R: applications that need lot of RAM – in **memory cache**
2. C: applications that need good CPU compute/databases
3. M: applications that are balanced (think medium) **general web apps**
4. G: applications that require GPU power, **machine learning**
5. I: applications that require lot of I/O operations **databases**
6. T2/T3 burstable instances (up to a capacity)
7. T2/T3 unlimited: unlimited burst

[**https://ec2instances.info**](https://ec2instances.info)(comprehensive guide in choosing instance type)

Burstable instances, will allow to handle unexpected peaks in load for a shorter period of time. Every burst will be compensated through **burst credits**. If the load is continuous and not for shorter period of time, instance will run out of credits, and performance will be **degraded**.

T2/T3 unlimited will allow to have unlimited burst, however you have to pay for it.

## Placement Groups

How actual instances will be place on aws infrastructure.

### Cluster

Instances will be placed together in same availability zone and will have low latency) high performance and high risk

* Same rack, same availability zone

### Spread

instances are spread across underlying hardware (limit to 7 instances per group per availability zone) for critical applications

### Partition

# Scalability and High Availability

## Scalability

Ability of an application to handle greater loads by adopting.

### Vertical Scalability

This kind of scalability will adopt by increasing the size of instance. Example Replace the junior operator with senior operator. This kind of scalability works for non-distributed applications like database.

* From t2.nano
* To u-l2tb l.metal

### Horizontal Scalability

This kind of scalability will adopt by increasing the number of instances of the application. Example hire more junior operators to handle the load. This kind of scalability works for distributed applications like web apps.

* Auto Scaling Group
* Load Balancer

## High Availability

High availability means you have multiple instances (at least 2) of same application in different availability zone.

* Auto Scaling Group multi Availability zone
* Load balancer multi Availability zone

# Elastic Load Balancing (ELB)

* Managed load balancer by aws
* High availability
* Integration with large number of aws services
* AWS will take care of upgrade and maintenance
* Provide some configurations/customizations to configure it according to your needs
* Expose a single end point for client to connect through DNS.
* Spread load across multiple downstream instances
* Seamlessly handle instances failure
* Perform regular health checks of instances
* Provide SSL termination on the balancer, and pass the request in non-encrypted form to the instances.
* Enforce stickiness with cookies
* High availability
* Separate public traffic from private traffic
* Load balancer can scale but not instantaneously, contact AWS for a **warm-up**
* 4xx errors refer to client generated errors, 5xx refers to sever generated and 503 error means load balancer is at capacity/overloading or no registered target

## Target Groups

Target group tells a load balancer where to direct traffic to: EC2 instances, fixed IP addresses; or AWS Lambda functions, amongst others.

* Each target group is used to route requests to one or more registered targets (instances).
* Can route to EC2 instances
* Can route to ECS tasks
* Can route to Lambda functions (balancer in front of serverless)
* Can route to private IP addresses

**Types of Load Balancers in AWS**

1. Classic Load Balancer (v1 2009 old generation) (Http, Https, TCP)
2. Application Load Balancer (v2 2016 new generation) (Http, Https, WebSocket)
3. Network Load Balancer (v2 2017 new generation) (TCP, TLS (secure TCP), UDP)

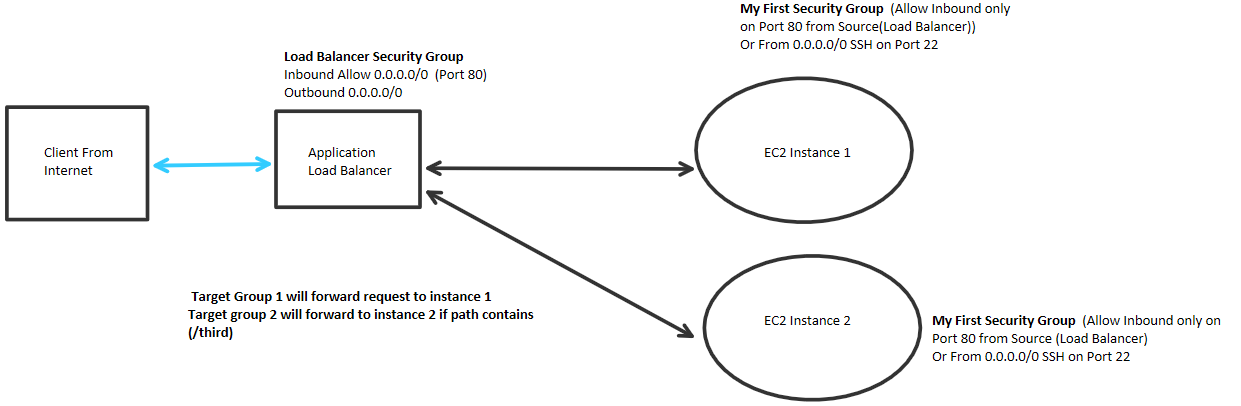
**Load balancer can be private or public**

## Classic Load Balancer (CLB)

* Http, Https are layer 7
* TCP is at layer 4
* Health checks are TCP or http(s) based
* **One load balancer for each application**

## Application Load Balancer (ALB)

* **Layer 7 only** (HTTP)
* **Fixed Hostname similar to Classic (not static/fixed IP)**
* Load balance to multiple HTTP applications across machines **(target groups)**
* Load balance to multiple applications on the same machine (ex. Containers)
* Support for **Http2** and **Web socket**
* Supports traffic redirect from Http to Https
* Support Route Routing
* Routing based on path in URL (example.com/users & example.com/posts)
* Routing based on hostname in URL (one.example.com & other.example.com)
* Routing based on query string, headers (example.com/users? id=123&orders=false)
* Best fit for **container** or **micro services** based applications. (Docker and Amazon ECS)
* **Has a port mapping feature that allow to route to dynamic port in ECS**
* **One load balancer can handle multiple applications**
* **X-Forwarded-For** (Client IP)
* **X-Forwarded-Port** (destination port client used to connect to balancer)
* **X-Forwarded-Proto** (Protocol used by client to connect to balancer)
* X-Forwarded headers are set by load balancer and will be sent in request to the instance/application.



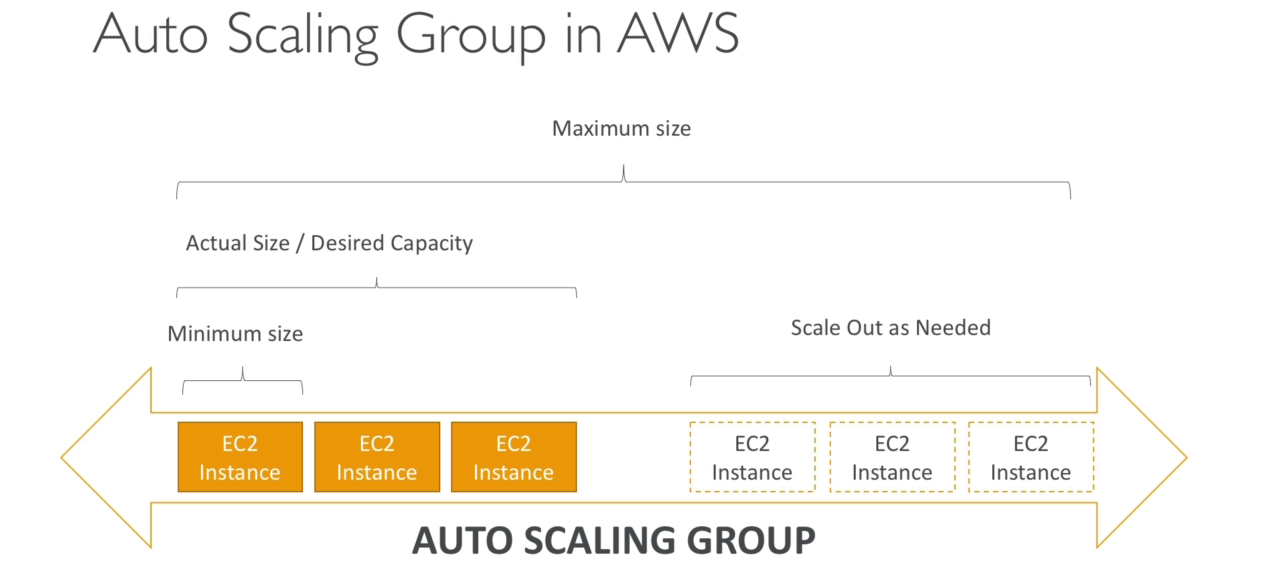
***Figure 1: Hands on Lab***

## Network Load Balancer (NLB)

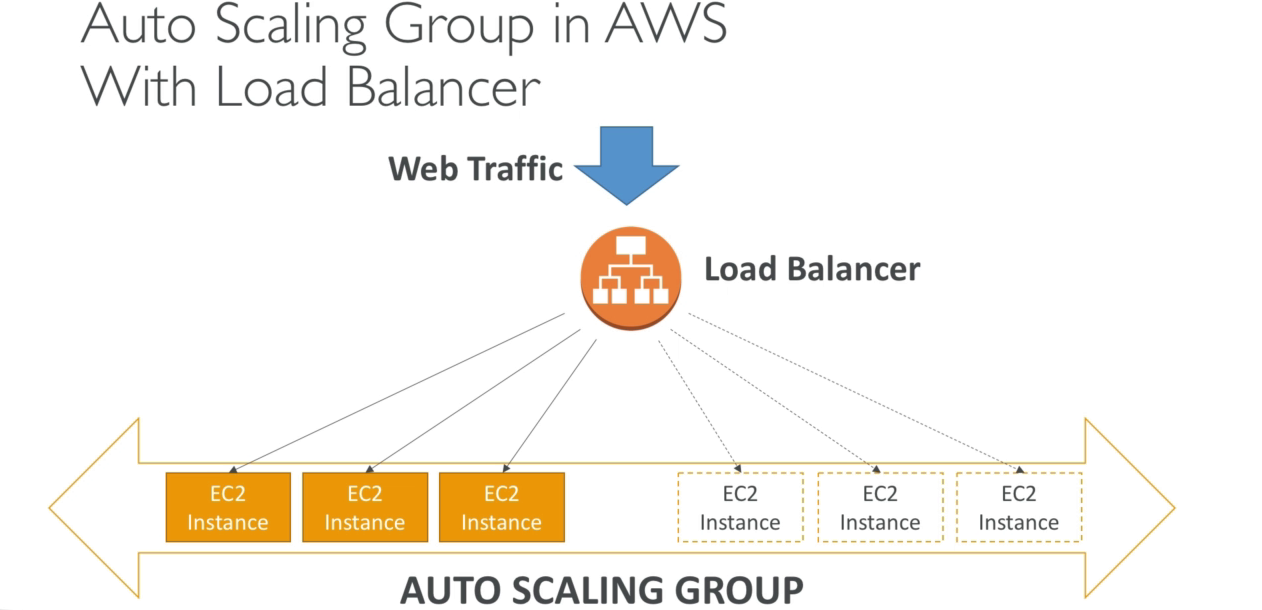
* Level 4
* Extreme performance is required
* For **TCP, TLS and UDP** communication
* Can handle millions of requests per second
* Latency ~ 100ms (400ms for ALB)
* One static IP per Availability Zone
* **Elastic IP can also be assigned (ALB can have static hostname instead of static IP)**

## Auto Scaling Group

* Scale out (Add EC2 instances) to match the increased load
* Scale in (Remove EC2 instances) to match the decreased load
* Set min and max instances
* Automatically **register** new instances to load balancer if needed (automation)



***Figure 2: Auto Scaling Group***



***Figure 3: Auto Scaling Group***

**Things Require for ASGs**

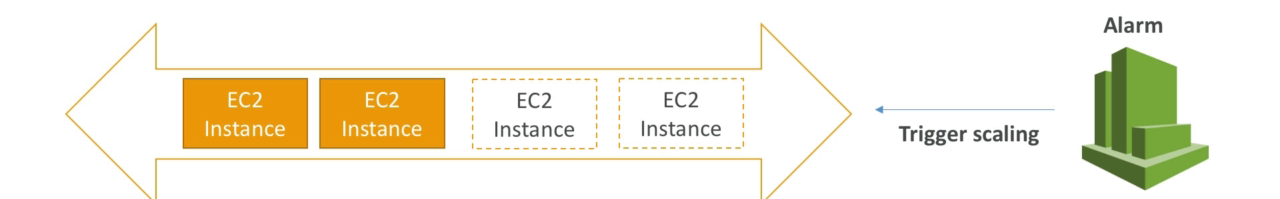
1. A launch configuration
   1. AMI + Instance Type
   2. EC2 User Data
   3. EBS Volume
   4. Security Groups
   5. SSH Key Pair
2. Min Size, Max Size and Initial Capacity
3. Network and Subnet Information
4. Load Balancer/Target Group Information
5. Scaling Policies

**Q: How does auto scaling work?**

**Answer: Auto scaling can be done based on alarms from cloud watch**

### Auto Scaling Alarms

* An alarm monitors some metrics of Ec2 instances, when a metric of machine reach certain value, the alarm goes up and tells the ASG to **scale out**, when the metric goes down certain value the alarm tells ASG to **scale in**.
* Metric can be like Average CPU usage
* Metrics are computed for all the instances under ASG (Average Value)
* Scale out Policy can be defined based on the alarm
* Scale in Policy can be defined based on the alarm



***Figure 4: Auto Scaling with Cloud Watch (Alarms)***

## Health Check

* Health checks are performed to know whether instance is healthy and can handle request
* Usually health check is done by requesting on <url:port/health> , if response is 200 (OK) then instance is available and can handle request, otherwise not
* Health checks by default happen every 5 seconds, but it can be changed

# EBS

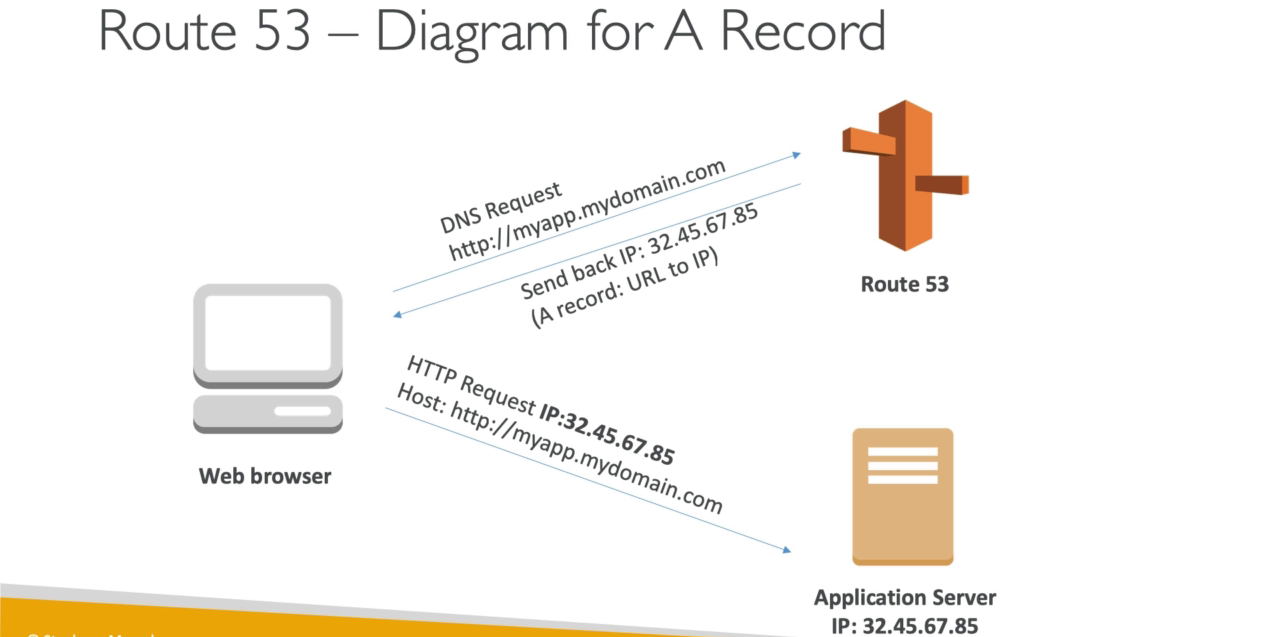
* EC2 instance loses root volume on termination of instance, so for a permanent storage EBS storage is used
* EBS storage is like a USB stick over a network.
* EBS are locked in AZ (availability zone), provisioned in one AZ, it cannot be accessed in another AZ

# Route 53

* A managed DNS (Domain name system)
* DNS is a collection of rules and records which help clients understand how to reach a server through URL’s
* Allow client requests to reach actual IP’s through URL
* Types of records in AWS
  + **A:** URL to IP4
  + **AAAA:** URL to IP6
  + **CNAME:** URL to URL (**only work with non-root domain and paid**). For example, route the domain name to the aws resource (load balancer, ec2 instance)
  + **Alias:** URL to AWS resource (**work with root and non-root domain, free, health checks also work**)

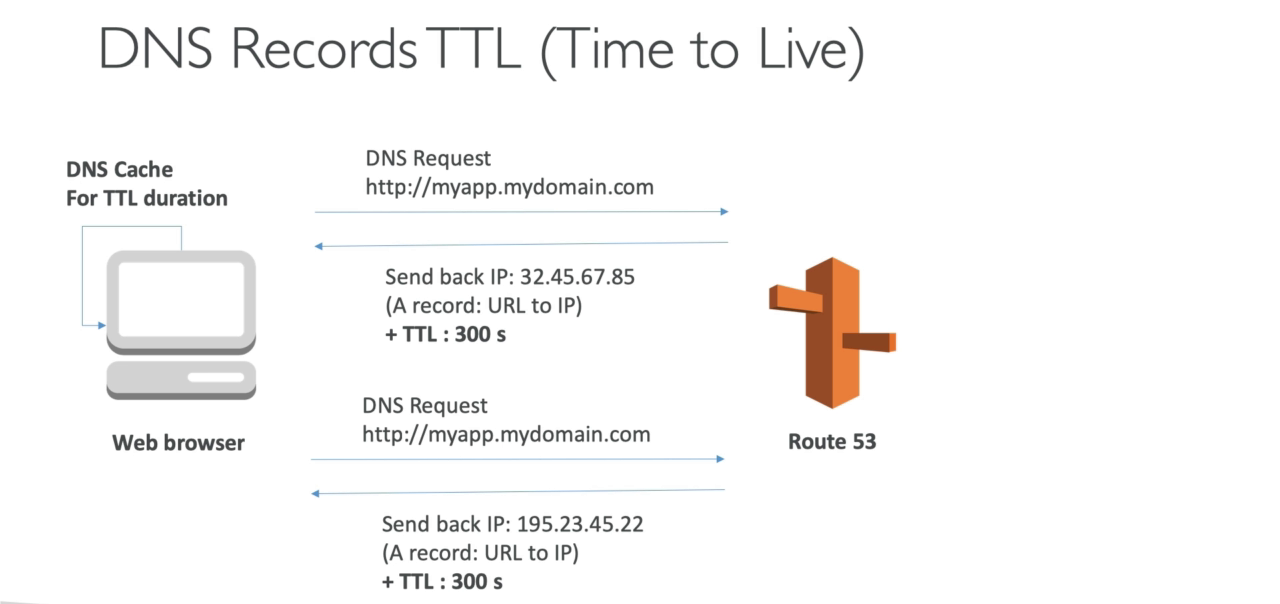
## A Record Example

* First send a DNS request to route 53 to resolve URL to actual IP
* Then HTTP second request will be sent by browser to the actual IP



## TTL

* TTL value will allow to lower the requests on DNS
* For 300s web browser will use cache IP address, instead of requesting IP from DNS again
* If some changes are done to the record, for cache time we will have an outdated record



# AWS Lambda

* Server less
* Pay only for the compute time you used for example if only a single request came for whole month then you only pay for that 2 to 4 seconds, not like EC2 where you have to pay for complete month.
* Infrastructure is provisioned and scaled automatically by AWS
* Integrated with AWS stack
* API Gateway, Kinesis, Dynamo DB, S3, Aws IoT, CloudWatch Events, CloudWatch, SNS, Cognito, SQS

**Example**

Image🡪 S3🡪AWS Lambda 🡪 New Image in S3 (Compressed)

Store metadata in Dynamo DB