Session 8

⇒ IAM Users (Some like user identity).

⇒ More way to triggered lambda

↙ Imp

API Gateway (API end Point).

[Can give this to consumer].

Don't Need Have access to S3 bucket.

⇒ First Need to Create a resource.

Deploy the API

. Use case to upload a pic and then Do the Some Application.

Amazon Polly Service (Practical).

⇒ Maintain entire versions of code

Aliases

⇒ . Rollback imp can be done using versions.

. Alias ⇒ Defined in API Gateway.

⇒ Split some traffic.

Weighted Alias

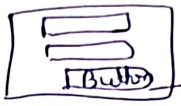
→ Cloudwatch works on the Push Methodology:

Prometheus & Grafana

(PUSH).

Practical Implementation.

1.> Feedback form → front end



→ API Gateway

Feedback form → Lambda

↓

entire processing

↓

200 else error

→ email (SNS).

CORS Error (Cross origin Resource Sharing).

⇒ When server to server communication (from Browser).

Basically users Don't Have Access to our End Users.

(API Gateway)

API Gateway → Lambda → Job.

/start

Create API end-point and that will trigger Lambda function.

API Gateway ⇒ Build API ⇒ Create Resource ⇒ Define Method.

⇒ Deploy API (To use it) ⇒ Will get Invoke url.

⇒ Wibaff — Ask user to select a photo. Upload the photo to (S3) and then automatically (API Gateway) rest of the things will work accordingly.

Lambda Versions Management — Versions Tab inside lambda.

Versions (V1 & V2)

In future say client does not like (V2) then we can switch back to (V1).

Aliases — To switch back to say older/previous version.

Weighted Alias — But still API Gateway is not sure where to route the traffic.

So after changing to previous version

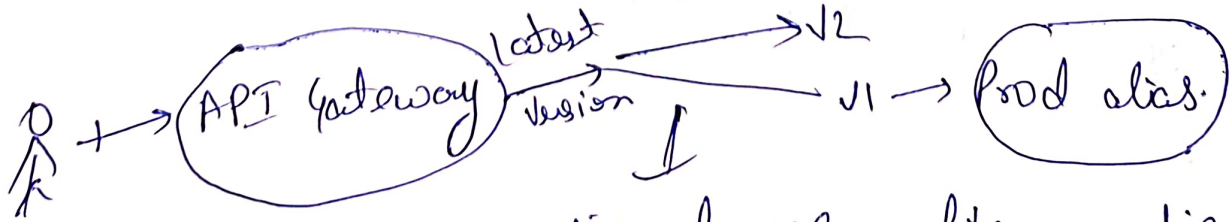
⇓

Creat Alias

⇓

Go to API Gateway

⇓



Do this change while creation

Method

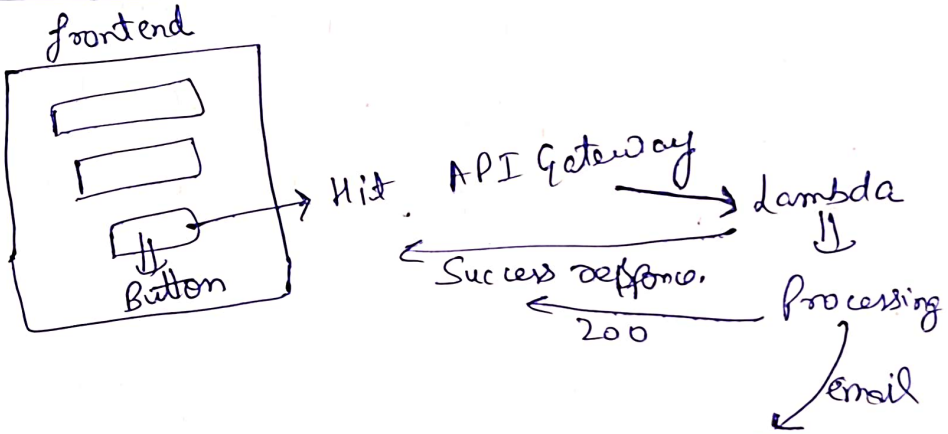
How to Split traffic.

Created a new version of code & want to split the traffic. 80% ⇒ V1 & 20% → V2 } Set it in Alias.

Canary Deployment.

Canary Deployment is a software release strategy where a new version of an application is gradually rolled out to a small subset of users before rolling it out to everyone.

Load Balancer **Feedback form** **Application**



- S3 can also work as hosting static websites. (Static Website Hosting).

CORS error . Cross origin Resource Sharing.

#

Frontend → Hostname https://gfg32 ——— .com.

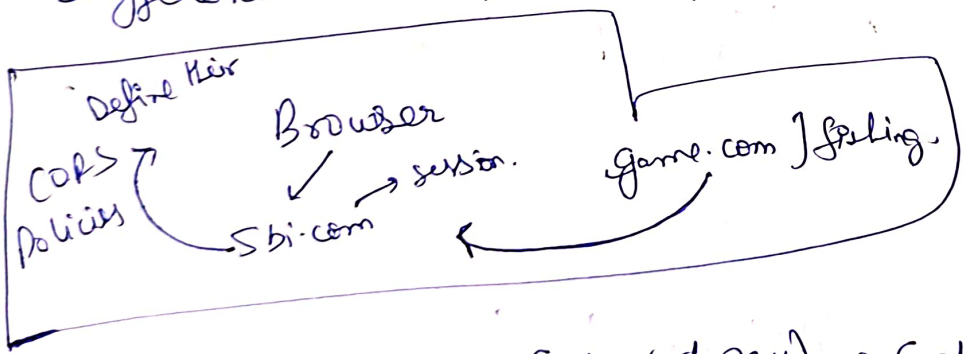
Backend → API Gateway

Lambda ↗

] 2 different Hosts.

⇒ frontend is running at different url/Host other then Backend and trying to hit from frontend to Backend. (Data sending).

⇒ A (CORS) → Browser error (Host should be same). error happen when your web app (frontend) tries to make a request to server (Backend) with a different domain, and the server doesn't allow it.



⇒ Inside resource (API Gateway) ⇒ Enable (CORS)

Static Website Hosting (Using S3, Lambda & SNS (API Gateway))

Steps

Flow Diagram

(1) S3 bucket create Object (feedback.html) upload

⇓

(2) Create lambda function (In this SNS connection, Publish all is Here).
(code).

⇓

(3) API Gateway (Invoke url added in html file).

a) Build API (Post)

b) Create Resource.

c) Create Method (PUT, POST, OPTIONS, DELETE etc)

d) Deploy API

CORS error can be removed from API Gateway Part

(4) SNS Service

a) Create Topic → This topic ARN will be in lambda function part.

Need to create a TOPIC, then Subscribe it.

b) Create Subscription

(5) Look to CloudWatch (if face any errors resolve accordingly).

Lambda — Versions ⇒ we can also add Weighted.
Aliases ⇒ Weighted Alias.

1) Route 53 ⇒ Map Custom Domains.
   - ↳ Scalable DNS.
   - ↳ A reliable way to route users to internet Applications.
   - ↳ Is a Highly available & Scalable cloud DNS web service.

Another Practical with AWS

⇒ Lambda is very light and used/charged only when used, still people preferred to use EC2 instances.

Reason — Lambda is Good only for Normal Applications where we have a very small running Application.
(Max Timout limit of lambda is 15 min)
Light weight & Not Complex.

Max Memory 10GB
Not Good for Complex Applications.

Lambda Use case

Website → Request OTP → Get OTP.

Program which will generate Random 4 Digit No ⇒ OTP ⇒ SNS

⇒ Don't Have any Kernel level access changes.

⇒ Lambda is event-Driven. (sending one event to API Gateway)

⇒ for all event-driven part Lambda | Limited invocations then only Lambda is Good.

Now How Virtual Machines ~~Deploy~~ used In Real World to Deploy the servers.

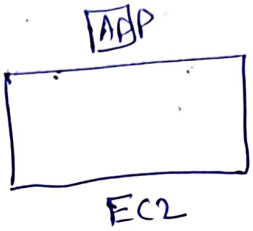
Viral (More & More People).

Then ~~Banks~~ Oracles.

// Website

To resolve this we Have ASG

Oracles Have Limited RAM/CPU.



⇒ ASG ⇒ Takes Metrics from CloudWatch

⇒ Also called as Horizontal Scale

⇒ Till now, Have Manually Created EC2 instances.

Auto Scaling Group

// Manage load on Servers

(eg) Black Friday Sale.

Steps Create a EC2 instance template

// ASG can use this template to launch Replica.

rule { Check my CPU usage & Memory

CPU > 80%

Then launch another EC2 instance which is just Replica

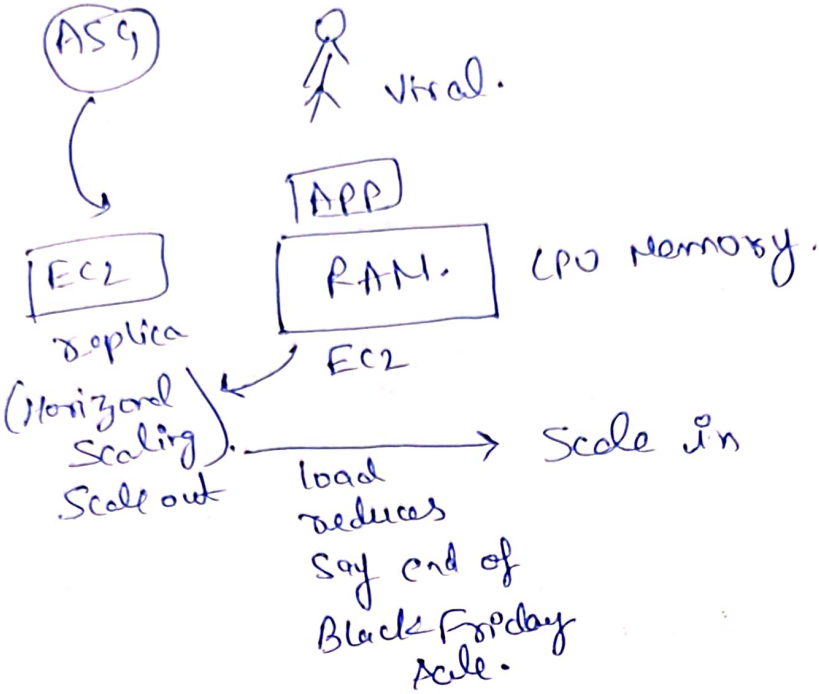
Auto Scale }

Suppose after end of Black Friday Sale. load is less, then will automatically

Scale in from Scale out

// Terminate extra services to reduce the cost.

Diagram



AMI ⇒ Amazon Machine Image

It contains everything needed to launch an EC2 Instance, Including

a.) OS (Ubuntu, Amazon Linux).

b.) Application Server (Apache, Linux).

c.) Runtime environment (eg. NodeJs, Python, Java).

d.) Any S/w or configuration we have set up.

Steps

Create Custom AMI

Instances. ⇒ Launch Instance
(Running Manually).
⇓
Connect
⇓
Root User.
⇓
Apache Install.
⇓
Systemctl. start httpd
⇓
/var/www/html ⇒ index.html

⇓
Content in file.

Now, when have to do **Scale out** event then don't need to Manually do this.

Approach Take a copy of EC2 instance
⇓
ASG
⇓
Then can create a same copy of existing/ Copied EC2.
⇓
Go to Instance from which need to Creat Image
⇓
Actions
⇓
Image & Temple ⇒ Create Image.
⇓
whatever EBS volume you have will take a copy/ of that stored volume.
Snapshot
⇓ EBS volume Have call the Ryo.
⇓
Creat Image (Private Image).
⇓
AMI Section (Custom AMI will be sent to ASG Now).
⇓
ASG (Inside EC2).

Systemctl enable httpd.
⇓
This command will automatically Enable Apach Server at the Startup (Boot).

Pre install apache web server.
↑
Image.
↙ ↘
Apache Server will be Started on Boot.
html file

⇓

Create Auto Scaling Group.

⇓

Choose Launch Template (ASG Group Name).

⇓

Click on Create a launch template.

⇓ (which AMI) (which Security Group etc

Create launch Template. (Template is Ready).

⇓ Again go to

ASG Next.

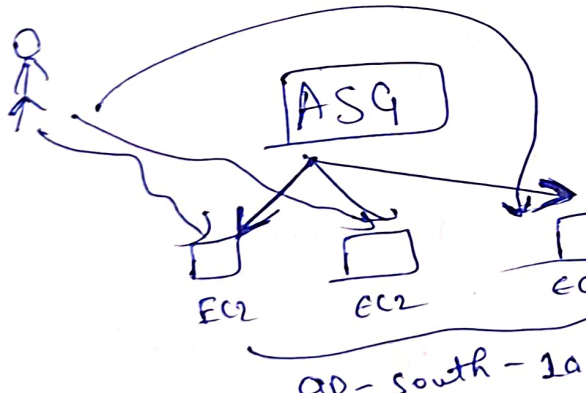
⇓

Select Multiple Availability Zone. (In which VPC wanted to launch the instance).

⇓

Next. (Load Balancing).

Example

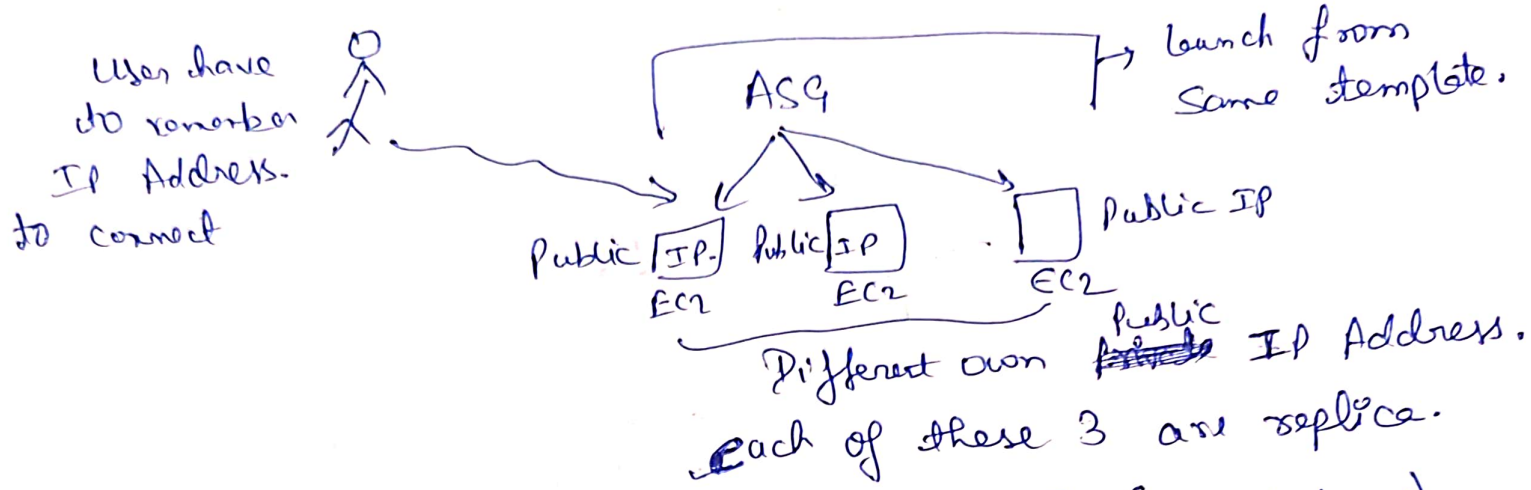


(All 3 are replica only).

Suppose here if Some disaster Happen then. Entire region is down.

Note ⇒ ASG to Distribute EC2 instances in Different Availability Zones (for High Availability).

Load Balancing — Why it is required Here.

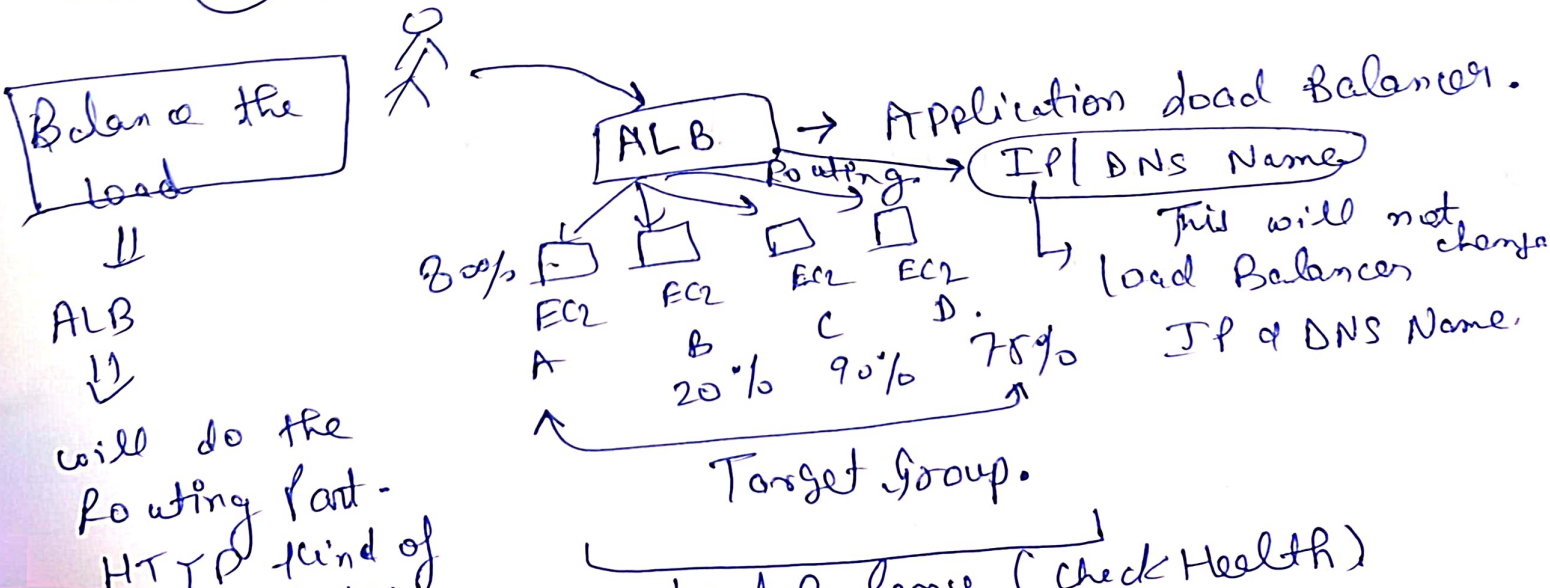


Each of these 3 are replica.

→ If all users are hitting the same IP (EC2 instance) so there is no Benefit of launching 2 instances.

⇒ If I gave IP address to clients of first 2 EC2 instances. and after some time due to Disaster / issue my these 2 EC2 instances are not in service, then ASG will launch another replica but client does not know IP address of this new Replica.

Approach User don't Have to Make request Direct to EC2 instance, instead can Make request to (ALB) Under (ALB) all EC2 instances will be present.



⇒

⇓

Application load Balancer.
(HTTP, HTTPS).

(2) Network load Balancer. (TCP, UDP, TLS)

⇓

Group Size (ASG group related configurations).
(Scaling Parameters).

⇓

Notifications ⇒ SNS Topics
Email sending.

⇓

Add Tags (optional).

⇓

Auto Scaling Group will be created.

---

* Do this whole process Itself as well.

Homework

Create a VI which will request OTP → validate OTP → send OTP on email, using SNS

ASG & Load Balancer

1.) Launch an Instance (Do some Configurations There).

2.) Go to the Launched Instance
⇓
Actions ⇒ Create Image.

3.) Go to ASG ⇒ Choose Launch Template.
⇓
Creat a Launch Temple.

4.) Back to Auto Scaling Group (Refresh)
get the Launch Template Name
and then Create a ASG (Group)