ec2 instance (ubuntu)
connect - putty
commands - update, apache2, service
index --> deploy

⇒ EC2 instance

Ubuntu

- Allow HTTP traffic
- Advanced details

User data

```
#!/bin/bash
```

shebang

```
sudo apt-get update -y
sudo apt-get install apache2 -y
sudo systemctl start apache2
```

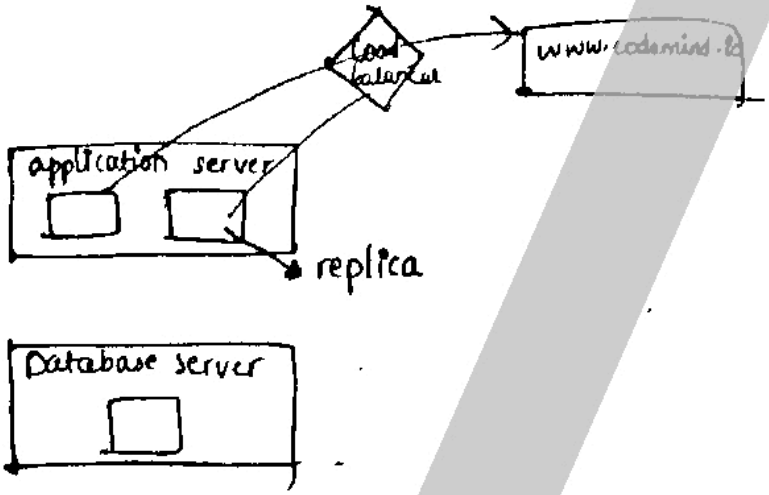
- Launch Instance

⇒ Load balancing is invidual service.
Without load balancing, auto scaling will not work.

Load balancing

i) redundancy
ii) high availability
- having ≥ 2 instances. (If A is down, other works)
- If more traffic is targetting, server goes down but due to load balancer we can access multiple servers.
- At a time trying to access server → concurrent

- 500 connections at a time to server.
  Eg: let capacity of server be 500 connections
- If other 500 connections are targetting,
  for flexible solution we use Load balancers
- It follows concept of 'round routing'



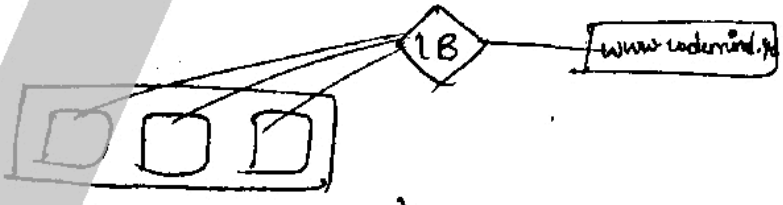

- load balancer → vpc & Internet

⇒ It distributes the traffic by using round


- Eg: capacity crossed / unhealthy servers.
- Load balancer pings to instance

reply ✓ healthy  reply ✗ unhealthy.

- Designed by AWS (Software as a Service)
  → maintained & managed by AWS.
- All servers are of same application.




(All 3 are same app)

, We need to take diff/multiple AZ's.

→ Because if one is down, everything is down.

→ No need to take care of public ip, when there is load balancer.

* Change something in app to check whether load balancer is working or not.

sudo su
apt-get update

⇒ apt-get install apache2 -y

~~apt-get~~ systemctl restart apache2

- sudo ~~vim~~ vi /var/www/html

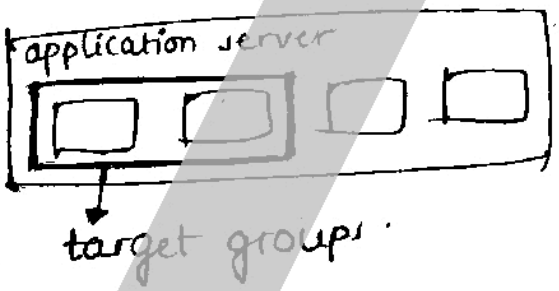
  sudo vim index.html

  esc :wq  i ⇒ insert

## Load Balancer

1) Target groups
2) Load Balancer

- LB routes the traffic to only target group. (cluster of servers).




target groups

↓

create target groups

↓

Instances // Routing targets to

↓

target group name. // aec-tg

↓

Protocol (HTTP) : 80

↓
VPC
↓
protocol version
• HTTP1

---

Health checks
↓
HTTP (or) Pf not web (ICMP)
↓
health check path (/)
↓
register targets (how many we want to attach to target grp?)
↓
select instances
↓
include as pending below.
↓
create target grp

HTTP → better performance
HTTPS → slow

---

~~Associate~~ • to check health status, click on it and come down.

## Load Balancer

create load balancer

- App LB (Application)
- Network LB (Routers, switches, TCP, UDP)
- Gateway LB (Routers)

• we can select this & then HTTP

aec - LB (name)

↓

scheme

Internet-facing

Everyone sending to same database ⇒ port-link (Internal LB)

↓

IpV4 (Dual stack ⇒ ipv4 + ipv6)

↓

vpc

↓

AZ's — us-east-1a, us-east-1b — Instances in ⇒ (Which are in target grp)

↓

Security grp's // check in Instances.

select

↓

Listener

HTTP , select a target group

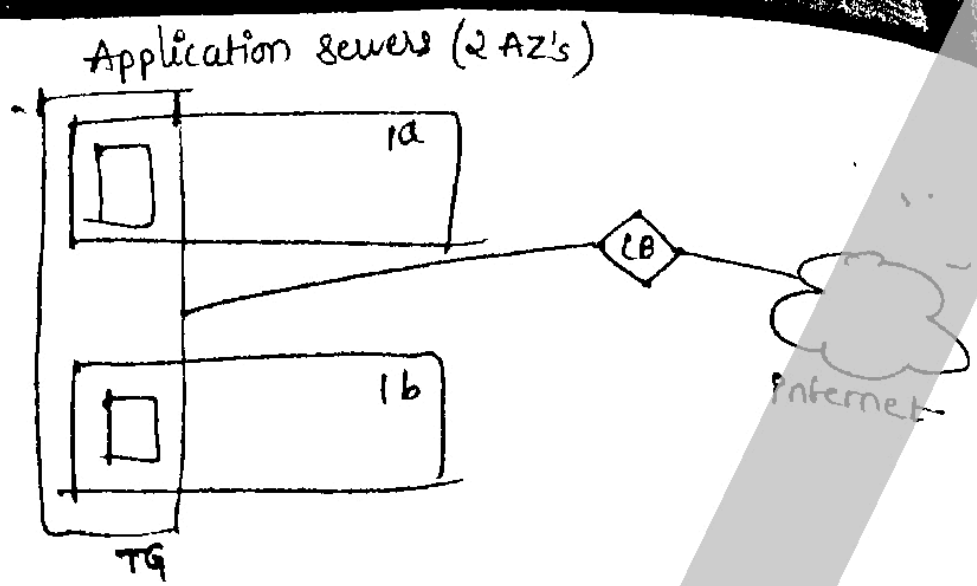
↓

create load balancer

- provisioning — unused → used (target grp)

⇒ Load balancer

- DNS name copy → mapped to application dns name (codemind.io)
- public ip 1 } servers.
- public ip 2 }




traffic will be redirected by security groups.

- If more traffic is coming, then we can create ec2-instances and attach to ~~load balancer~~ target group (But new AZ can't be created, at the time of attaching LB we need to attach).
- We are doing manually ..... we go for auto scaling

---

1) Vertical (scaling can be done)
2) Horizontal (In AWS, auto scaling can be done)

⇒ Vertical scaling

Scale up  Scale down

- Gaming app → ec2 with high config but it needs still more high config (scale up)
  → If we want ~~to scale~~ less config (scale down)

→ Auto scaling is not possible in AWS.

* Vertical (only one ec2 instance)
→ shut down (ec2 instances) → stop

Actions
↓
instance setting
↓
change instance type
↓

[Manual process]
(turnoff & turnon, needs boot)

Gaming app" scaling
vertical instance
+horizontal scaling

* horizontal (one ec2 instance is copied into other ec2 instance & launced)

scale-in → deleting 1 by 1
scale-out → increasing 1 by 1

• New one is created & attached to target grp
[Automate]

⇒ Autoscaling needs load balancing (dependent)
• If copy will be taken from image
(Image can be replicated)
• If we delete image, autoscaling will not work.

→ To take image
↓
stop instance (optional but recommanded)
↓
Actions
↓
image & templates
↓
create image

↓

Image name

↓

create image

AMI → Amazon machine Image

- AMI Images

## Autoscaling

Launch template

↓

Create launch template

↓

- name

↓

Template version description

(production)

↓

Go down

- My AMI

↓

⊙ owned by me

↓

Security groups

(launch-wizard-9)

(launch-wizard-10) → create launch template.

Launch config (old)

Launch templat (new)

Auto scaling grp
↓
name
↓
launch template
↓
~~create~~ next
↓
VPC, AZ (1a, 1b)
↓
requirements
↓
next
↓
Attach to an existing LB
↓
choose from your existing TG
(Jaya TG)
↓
No VPC lattice service
↓
Health check
(100sec)
↓
next
↓
Desired capacity
(2)
↓
Minimum capacity (Same)
(2)
↓
Maximum capacity
(5)
↓
Next → Next → ~~Next~~ Create AS Group.

If we select 1c, AS grp launches in 1c also, but load balancer doesn't work

(should not delete ≤ 2)

CPU
2 3
memory
8 16

2-3
3-5

delete

1) load balancing
2) target groups
3) autoscaling groups
4) Instances
5) ami-image

## Databases

Collection of data in a systematic format.

- Trillions of data in multiple servers and accessed easily.

③ → flat file database (Excel, csv) ⇒ AIML appns.
→ Relational database (TABLES) / SQL
→ Non relational database. (KEY VALUE)
JSON - Javascript object notation.

- MySQL
- Amazon Aurora
- Microsoft SQL server
- MariaDB
- Oracle
- Dynamo DB

## Amazon Relational Database Service (RDS):

Managed service.

- Server maintainance & energy foot print
- Software installation and patches.
- Database backups and high availability
- OS installation & patches
- scalability
- Data security

on-premisis  EC2  DB
Non-managed  Unmanaged (less part)  Managed

- Don't use Amazon RDS if 1,50,000 reads / writes per second
- High durability (like S3)
- completely safe.

Amazon aurora → DB

Dynamo DB service

1) Non-relational DB

→ AWS RDS
- MYSQL
- MIC SQL
- AMA
- MARADB
- POSTGRE
- ORACLE → POSTJRESQL

→ NO SQL/NON RELATIONAL
- AWS dynamodb
- cassandra (Apache service)
- Mangodb.

RDS

Create database

↓

Standard create

↓

MySQL

↓

- production
- dev / test
- free tier

↓

- Multi AZ deployment is not possible in free tier

Setting

- DB Instance Identifier name
  dr-2025 (DB engine)

↓

Credintials

// default admin.

↓

Master password (@ not accepted)

(Confirm)

↓

Instance configuration.

- Burstable classes
  (reserve 20% extra) 40GB
  provisioning 20% extra

↓

Allocated storage

(20GB)

↓

Autoscaling // need for burstable classes.

☑ Enable.

↓

1000GB (1TB)

↓

// Already have ec2 Instance

Connect to an EC2 Instance

// no

Don't connect to an EC2 Instance

↓

IPV4

↓

Vpc // vpc where application is present

↓

AZs

public access

⊙ Yes

. No

↓

VPC security group

. Create new

↓

New security grp name

dr-2025-rds-allow

↓

Additional config

↓

. Database port

3306

↓

password authentication

↓

Go down

. initial database name

//technicalhub

↓

Backup

☐ Enable

↓

Encryption

☐ Enable

↓

Maintainance

☑ Enable

↓

Create database

Monitoring

☐ Enable

Jaya #123

security group

↓

select

↓

edit inbound rules

↓

Anywhere.

↓

save rules

## work bench

MySql Connection ⊕

setup New Connection.

Connect Name: drive ready

Host name: ctrl+v ← // rds → click on database. → connectivity & security → end point.

User name: admin

password: store in vault → route

• ok

drive ready

public - sql work bench

private - ec2 instance to rds database. - private db

(ec2 sg & rds sg should communicate with each other

// only this ec2 instance can communicate with each other.

In EC2 create sg

↓

sg name //dr-fn

↓

description //allow rds connection.

↓

Inbound rule

MySQL / Aurora

.sg search

→launch wizard (2

select

//launch wizard (2)

rds

with private access.

public access ⊙No

Security grp provide

~~Ubuntu~~

~~sudo install~~

sudo apt-get update

sudo apt-get install mysql-client

~~sudo~~ mysql -u <username> -p -h <endpoint>

admin

(rds instance at the time of creation)

Enter password:

mysql> show databases;

# Without EC2

1) Standard create
2) Engine => MySQL
   Engine version
3) Templates => free tier
4) DB identifier name => //name
5) Credintials
   - Master user name //admin
   - password
6) Instances
   - Burstable
7) Storage
   - 20 GB
   - autoscaling → 1000 GB
8) Connectivity
   - don't connect to EC2 instance
   - VPC, subnet
   - Public => Yes
   - Security group //name
     //A-Z
   - port => 3306
9) Database authencation
   - password
10) Monitoring ☐
11) Additional config
    Backup ☐
    Encryption ☐
    Maintainance ☑

# S3 bucket hosting

① create bucket with unique name
② Add files/upload file (drag & drop & select)
   • upload
③ 1) static website hosting (enable)
   • give index.html

object URL
↓
google

2) block public access (disable)
3) Edit object ownership (ACLs enabled)