If the usage is increased in the db so we can do like buy a bigger machine and then use that but this does not matter in js because its a single threaded language

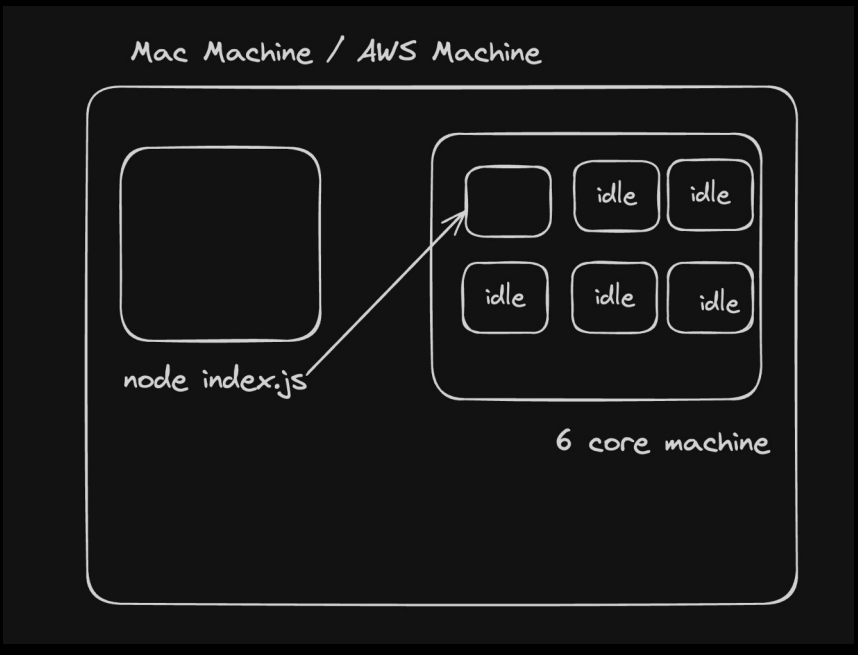
something off topic like
imagine you have 16 cores and 16 application running it does not mean that one application is running on a single core only if I run lets say zoom it can run on the core on which the chrome was running which is done using CONTEXT SWITCHING its switches based on the usage

the js and the node js cannot run on multiple cpu threads at the same time thats the issue
everything else such as golang, python,rust etc can run

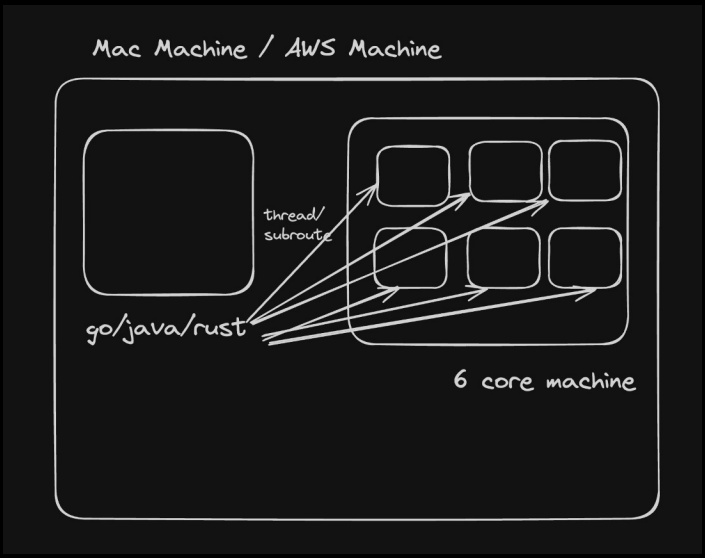
Vertical scaling

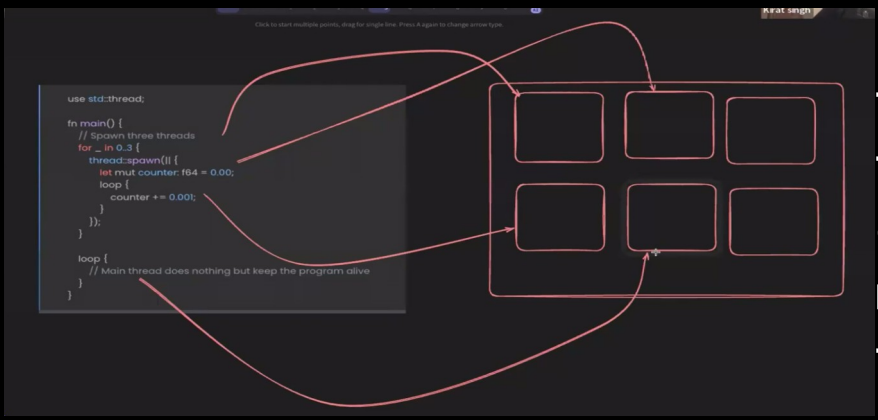
Vertical scaling means increasing the size of your machine to support more load

Single threaded languages



Multi threaded languages

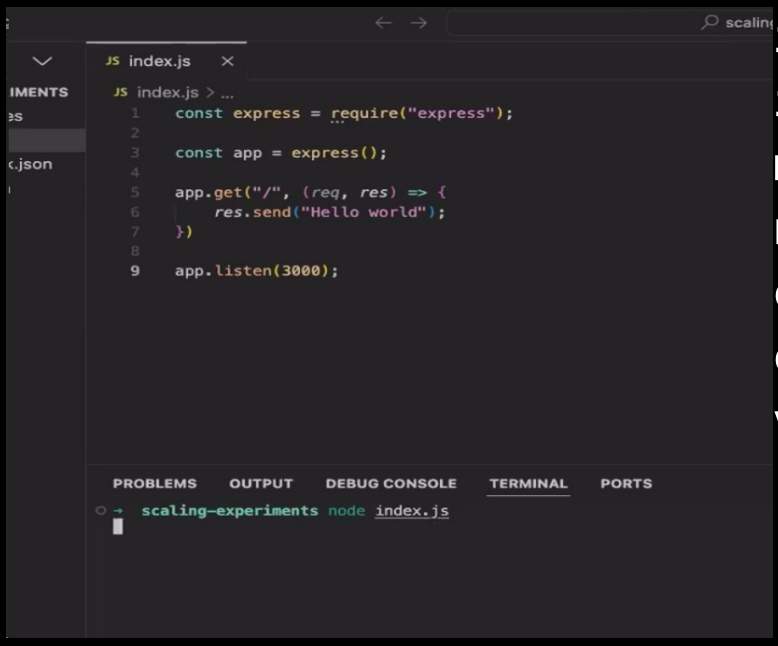




in rust you can create thread and then spawn them
context switching is mainly done to assign the process to the cores

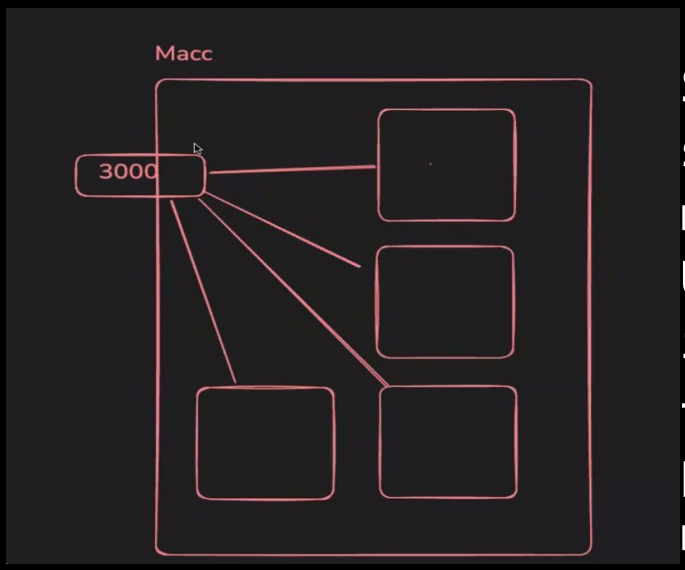
IF YOU ARE FORCED LIKE YOU HAVE TO USE ALL THE CORES OF THE MACHINE AND YOU HAVE TO USE THE NODE JS PROCESS THEN HOW YOU WILL DO THAT?
First principal approach open the node js process like is 10 cores than 10 times, you are like starting 10 different process which is pretty bad

all this is ok will it work for the express server??



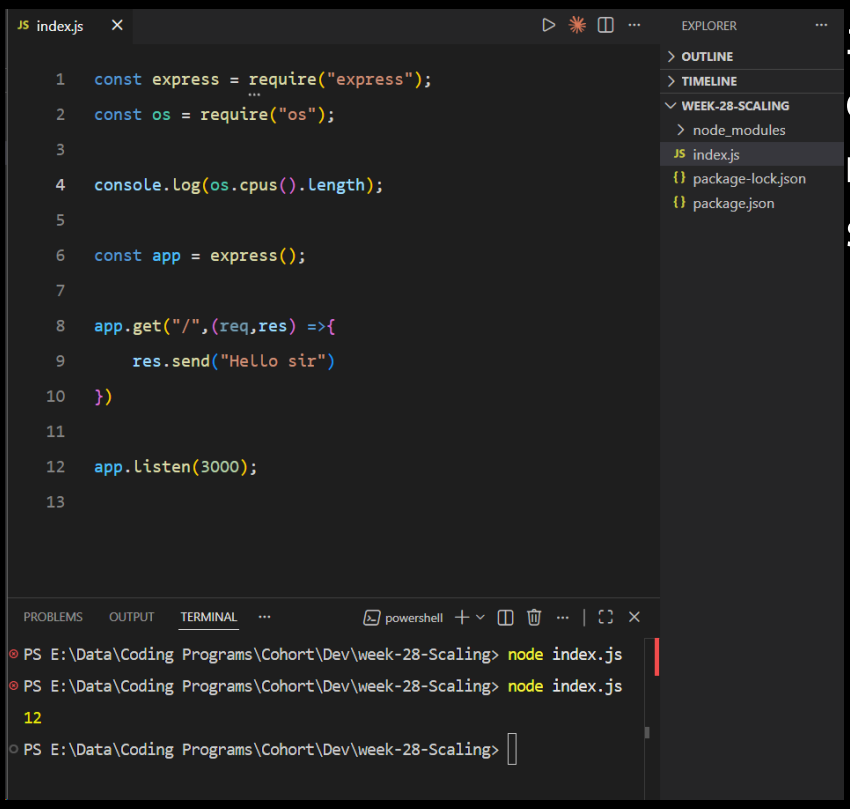
in this the issue is like if you will try to run multiple process it will not run due to port clashing for express so you cannot do that to scale vertically

HOW THIS WAS POSSIBLE IN RUST THEN?
In rust its like on one process it runs multiple threads like multiple process because its multi threaded and can use multiple cores…



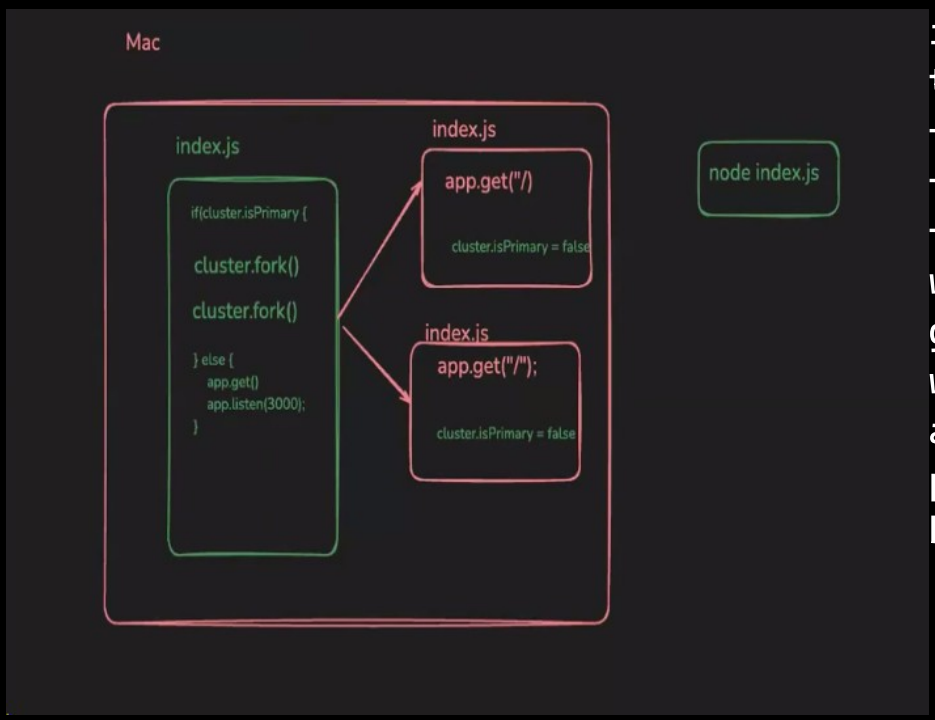
So how can you like use a single port and use that to run multiple node js process??
USING CLUSTER MODULE IN JS
its like similar to forks in c++ you can create a parent process which can create multiple child process

EXPLORING CLUSTERS



in this the cpus().length tells the number of cores in the system

SO what is happening now is we create a cluster which has the code for everything which like create a fork which contain the same code which can create another fork this becomes a loop so to fix that here comes a cluster.isPrimary this helps to assign like which is primary and which is child
so how we run is



in this see for the parent the isPrimary is true and for the other clusters its false that means for the false one the express code will work which is pretty good and that is what we want
abh jabh be req aayegi 3000 par the req will be load balanced among the systems

```
index.js
index.js > ...
1  import express from "express";
2  import cluster from "cluster";
3  import os from "os";
4
5  const totalCPUs = os.cpus().length;
6
7  const port = 3000;
8
9  if (cluster.isPrimary) {
10   console.log(`Number of CPUs is ${totalCPUs}`);
11   console.log(`Primary ${process.pid} is running`);
12
13   // Fork workers.
14   for (let i = 0; i < totalCPUs; i++) {
15     cluster.fork();
16   }
17 } else {
18   const app = express();
19   console.log(`Worker ${process.pid} started`);
20
21   app.get("/", (req, res) => {
22     res.send("Hello World!");
23   });
24
25   app.get("/api/:n", function (req, res) {
26     let n = parseInt(req.params.n);
27     let count = 0;
28
29     if (n > 5000000000) n = 5000000000;
30
31     for (let i = 0; i <= n; i++) {
32       count += i;
33     }
```

Chat ⌘L Edit ⌘K

the parent process will run the if code and the else code will be done by the child process

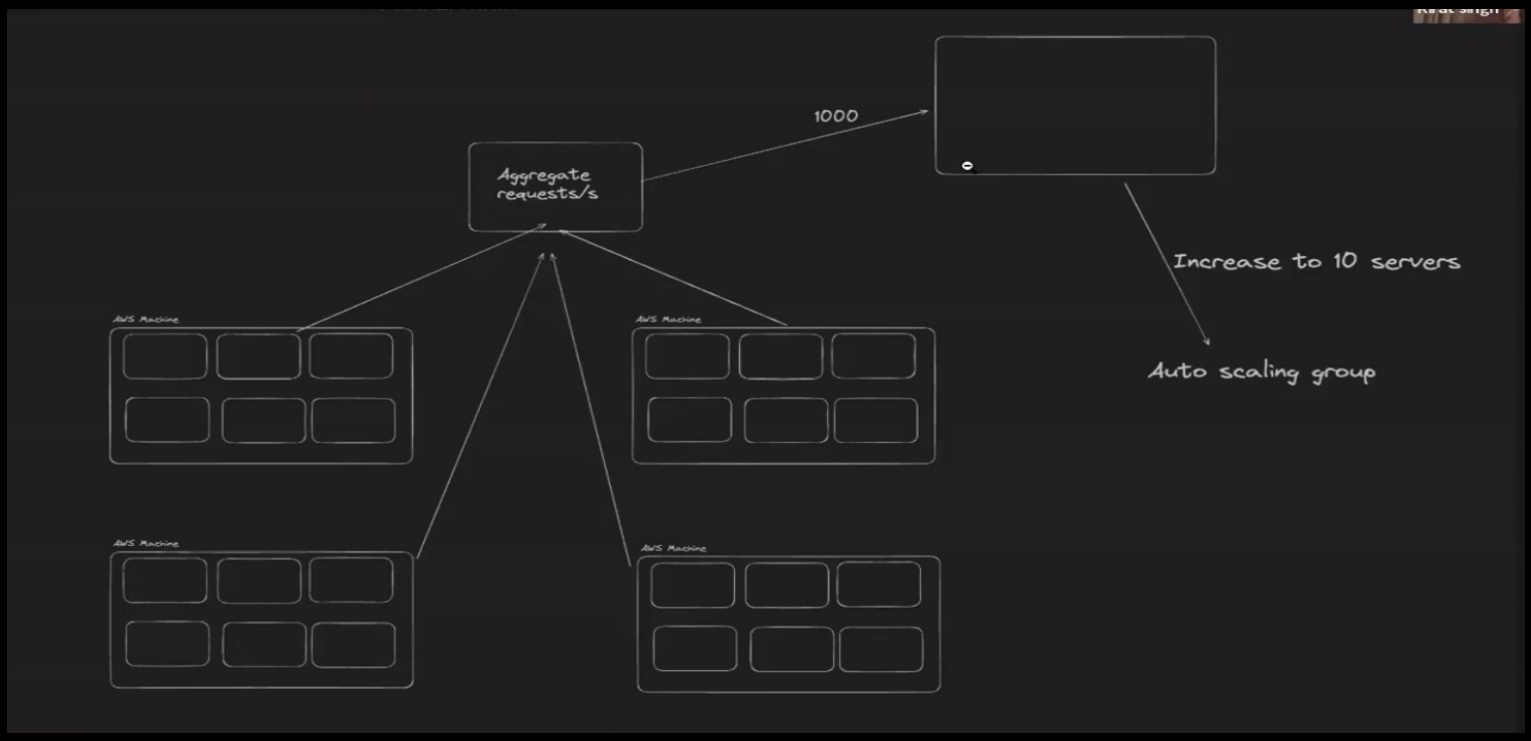
the cluster.fork() this runs process based on the number of cores in the system

but in this there is also a issue like whenever a browser is running its somehow gets pinned to a same process and not able to use multiple cpus even after doing the fork thing
but if you like run on different browser it works
the browser somehow caches and runs on a single thread only

```
22  app.get("/", (req, res) => {
23    res.send("Hello World!");
24  });
25
26  app.get("/pid", (req, res) => {
27    res.send("Hello World !" + process.pid);
28  });
29
```

in production how we do is like we create two different files bin.js , index.js
in the index.js just the whole logic for the express server and the bin.js file contains the if else part

SUMMARY WAS THAT THE MULTI THREADED PROCESS WAS 10 TIMES FASTER THAN THE SINGLE THREADED PROCESS



how this is working like whenever like think 250k request are hitting the backend one system cannot handle that so you increase the number of system but there is a issue if the user decreases you want the system to reduce also if the load is less this thing is handled by the **AUTO SCALLING GROUPS** which is like it tells the aws server the users are more increase the number of system it does that and think if at night the number of users decreases so reduce that based on the users it will decrease and it decreases slowly slowly not very fast

the request which are coming are called aggregate requests , it scale up or downs the statless http servers these are only for stateless http server only
fal.ai ⟸ lets you generate images

this is not for websockets

Capacity estimation

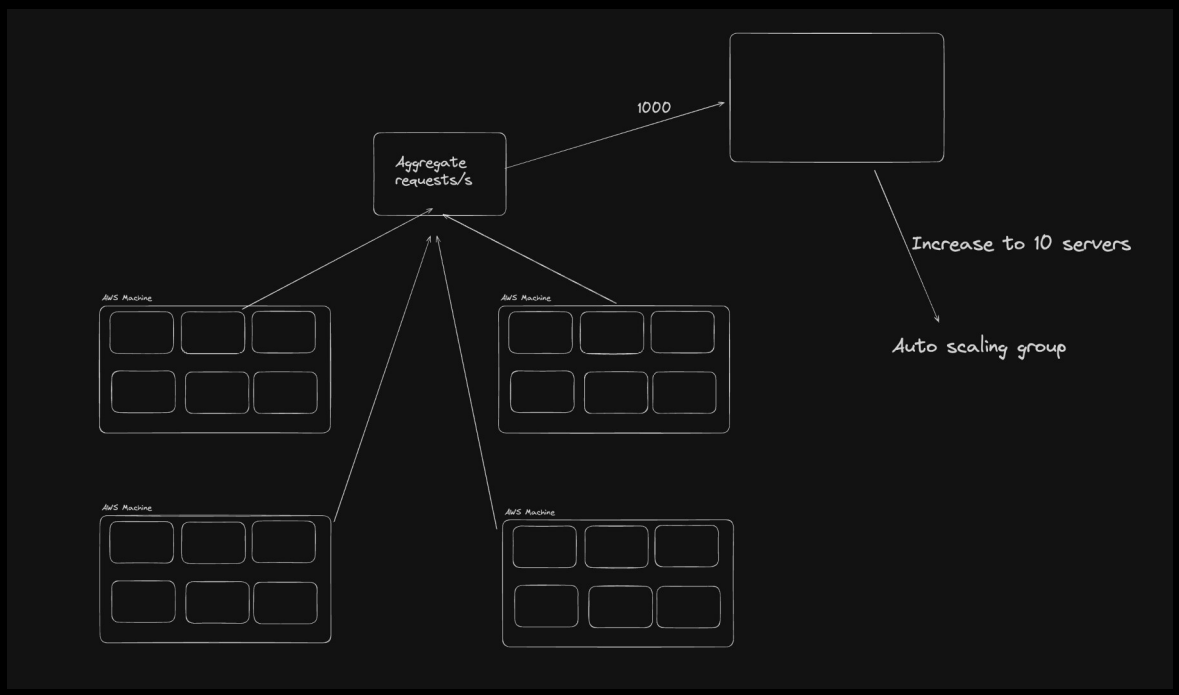
This is a common system design interview where they'll ask you

1. how would you scale your server
2. how do you handle spikes
3. How can you support a certain SLA given some traffic

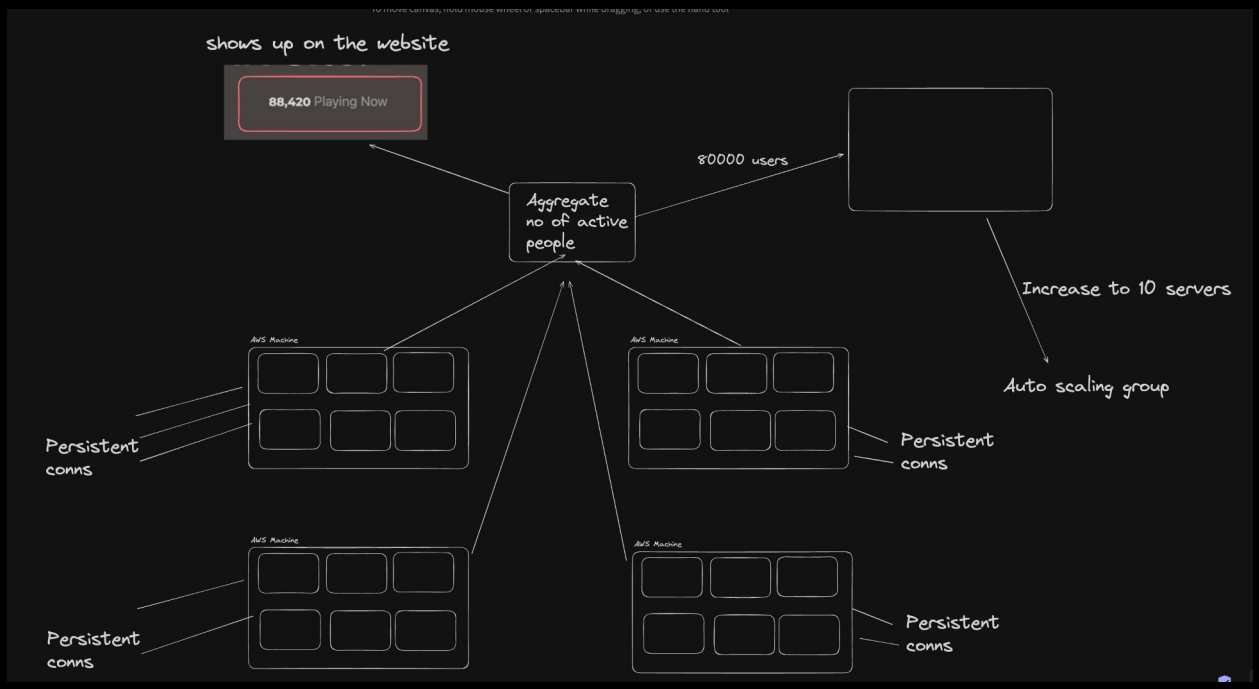
Answer usually requires a bunch of

1. paper math
2. Estimating requests/s
3. Assuming / monitoring how many requests a single machine can handle
4. Autoscaling machines based on the load that is estimated from time to time

Example #1 - PayTM app

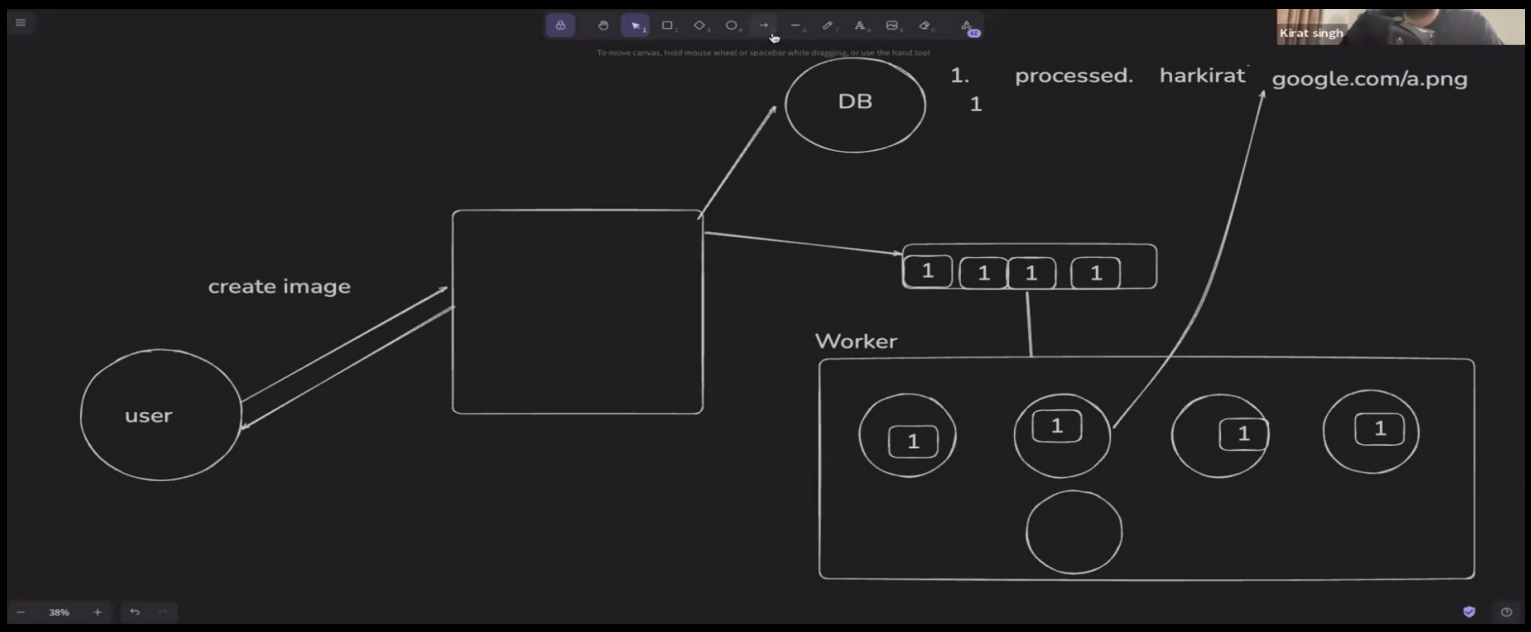


Example #2 - Chess app



THIS WAS FOR EASY TASKS NOT FOR HEAVY OPERATION

FOR GENAI APPLICATION AND VIDEO UPLOAD PIPELINE
you cannot scale up or scale down in this
how to scale which has long running backend because in this we the process in between to down scale the system as it take long time the process may break down in between

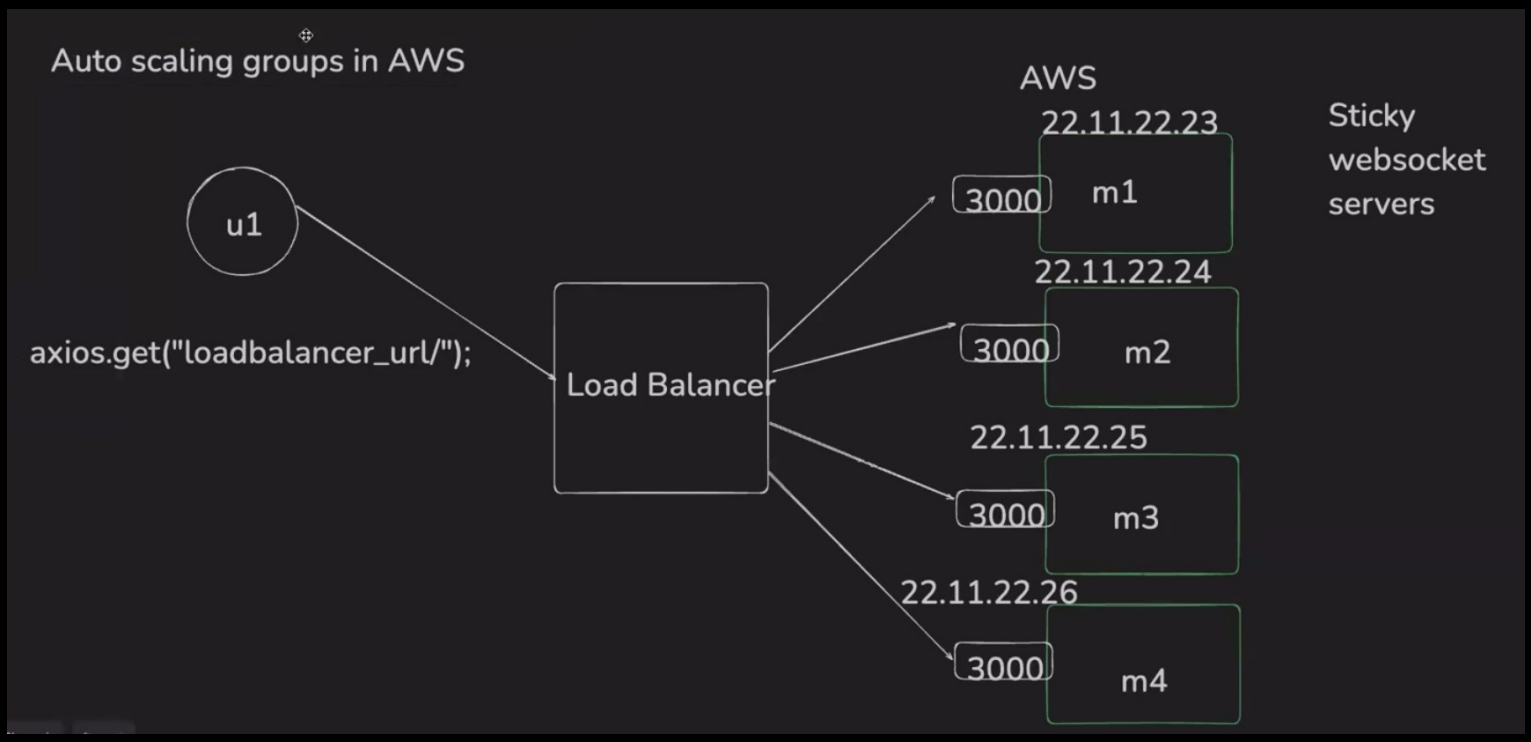


how this is done like take the input from the user stored in the db like in in_process something with name and id and then pushed to some queue, like kafka queue or redis queue and then is used by some worker one by one and as they complete the task the data is changed in the db that task completed and how the user gets to know by using polling every 5 min the system checks any changes, now any changes like that

the first box which is http layer is just used to handle easy task from multiple users and the worker does the task asynchronously picking things from the queue , pubsub etc
THIS IS CALLED AS ASYNCHRONOUS BACKEND COMMUNICATION USED BY YOUTUBE UPLOADING, LEETCODE, CODEFORCES , AI IMAGE GEN WEBSITE(IDEOGRAM) ETC

WS scaling is more difficult than this

IN COUNTER STRIKE we use STICKY WEBSOCKET SERVER



this was the older method in which we used to start multiple servers on multiple ips and then use the load balancer to balance the load based on the number of users and then send the users accordingly and its like increase and decrease the user based on the concurrent users it upscales and downscales

this stuff is provided by the AWS already which is called as auto scalling groups which does this by themselves managing the number of servers based on the usage

ASGs in AWS

An **Auto Scaling Group** (ASG) in AWS (Amazon Web Services) is a service that automatically adjusts the number of EC2 (Elastic Compute Cloud) instances in a specified group to meet the demand for your application. It helps ensure that you have the right amount of compute capacity at any given time by automatically scaling up (adding more instances) or scaling down (removing instances) based on demand. This helps optimize costs and ensures that your application is always running efficiently.

Features

1. **Automatic Scaling:**
    - Based on predefined metrics like CPU utilization, memory usage, or custom metrics, ASGs scale your EC2 instances up or down to maintain optimal performance and availability.
2. **Health Checks:**
    - ASGs perform regular health checks on the EC2 instances in the group. If an instance becomes unhealthy, it is automatically replaced with a new one.
3. **Scaling Policies:**
    - You can configure scaling policies to define how the scaling should happen. This could be based on a schedule, like increasing the number of instances during peak traffic hours, or dynamically adjusting based on real-time metrics.
4. **Launch Configurations/Launch Templates:**
    - These specify the configuration of instances launched within the group, such as the AMI (Amazon

Machine Image), instance type, security group, key pair, and other parameters.
5. **Desired, Minimum, and Maximum Instance Counts:**
    - You define the minimum and maximum number of instances the group can scale between. The **desired capacity** is the target number of instances you want the ASG to maintain at any time.
6. **Elastic Load Balancer Integration:**
    - ASGs can be integrated with Elastic Load Balancing (ELB), so new instances launched by the Auto Scaling Group are automatically registered with the load balancer, ensuring traffic is distributed across healthy instances.

Why use Auto Scaling Groups?

- **Cost Efficiency:** ASGs help reduce costs by ensuring you're only running as many instances as needed to handle the load.
- **High Availability:** Automatically replacing unhealthy instances keeps your application running smoothly.
- **Flexibility:** ASGs support both manual and dynamic scaling policies to cater to a wide range of use cases.

AWS - auto scaling group
GCP - MIDs (Manage instance group)
we are like creating a ec2 machine with our code and then we will create a image of that so that we can replicate the same machine for different systems
keypair in the ec2 is like assigning people who all can access the machine and the data inside that

chmod 400 pdkey.pem (DO IN GITBASH)
ssh -i pdkey.pem ubuntu@16.170.98.95

using option 3 in the node to install in ec2

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
nvm install --lts
node -v
npm install -g bun
git clone https://github.com/100xdevs-cohort-3/ASG
go to folder /ASG
bun install

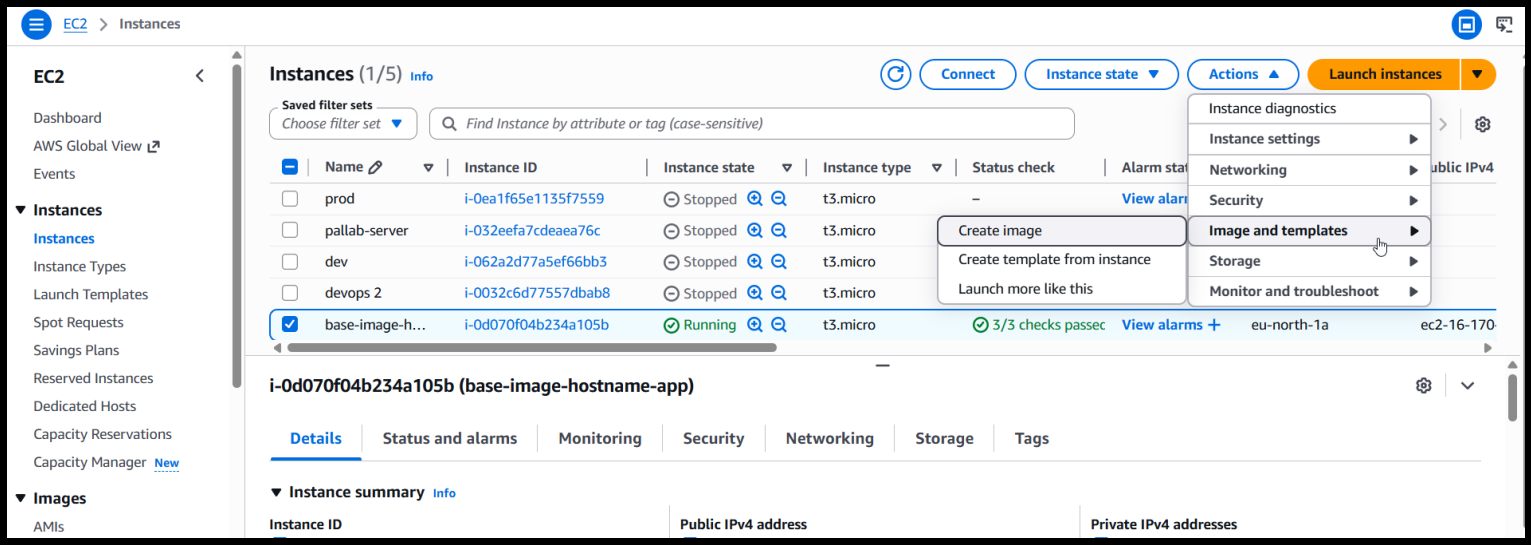
- bun bin.ts
- Install pm2

then go to publicip:3000
but we have to change the security group
but that alright because we don't have to use this we just have to clone this
npm install -g pm2
now to start bun file using pm2 use this
which node(get the location of the node)
which bun
pm2 start --interpreter /home/ubuntu/.nvm/versions/node/v22.14.0/bin/bun bin.ts

```
pm2 start --interpreter
/home/ubuntu/.nvm/versions/node/v24.14.1/bin/bun
bin.ts

pm2 logs 0  to check the process logs
pm2,forever will help if the server crashes it will
restart and also helps in running in background
```

Creating an ami/image

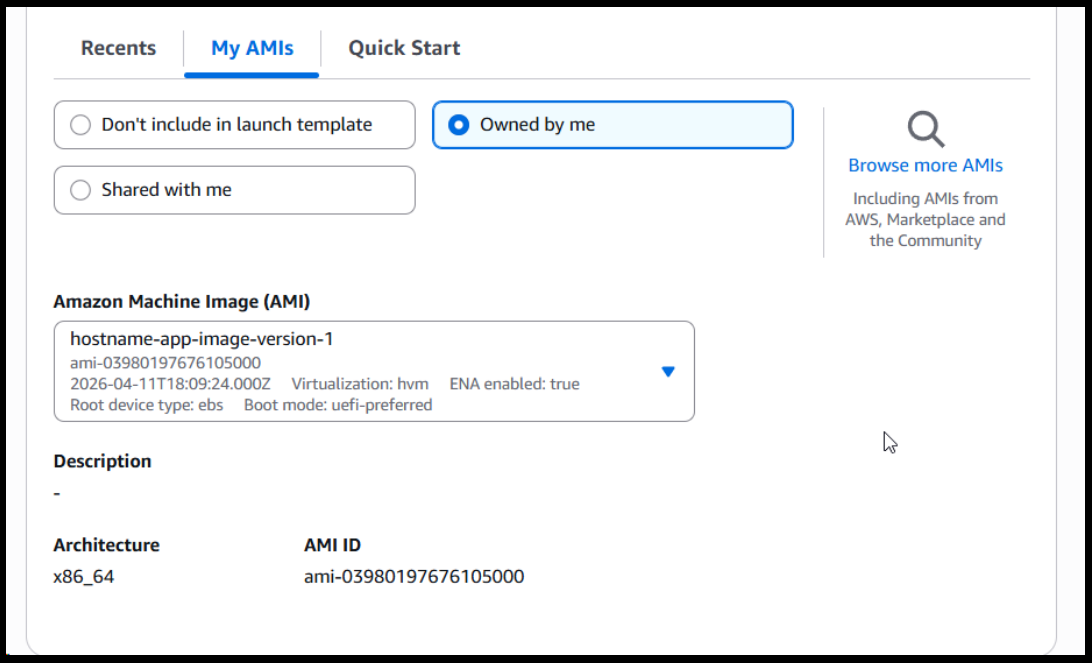
An **AMI (Amazon Machine Image)** in AWS (Amazon Web Services) is a pre-configured template used to create a virtual server, known as an EC2 (Elastic Compute Cloud) instance. It contains the operating system, application software, and settings required to launch and run the server.



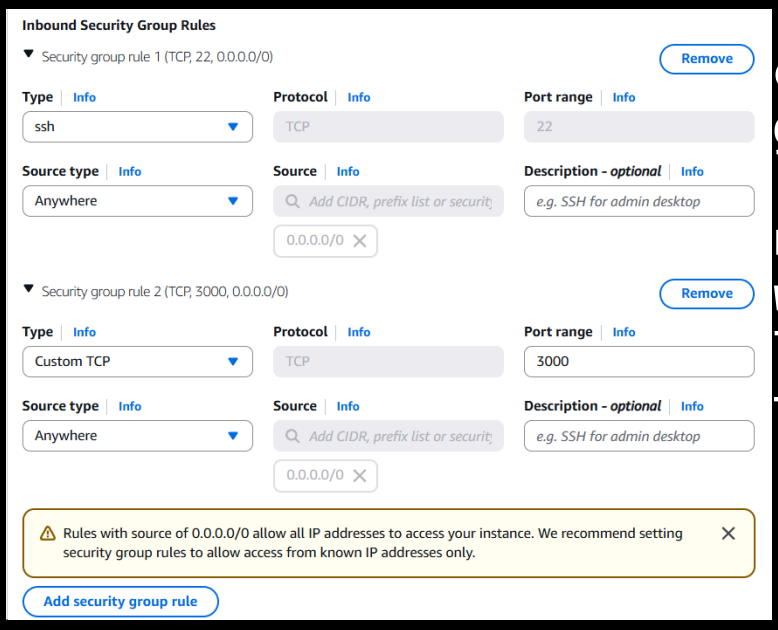

```
Give a name and then create
hostname-app-image-version-1 then create image

it has ubuntu, npm, node , bun, pm2 , application
code
```

now while creating a instance you can select my amis and then use that
but we don't what to do it manually we will use auto scaling groups in aws
ELASTIC BLOCK STORE == chota bada ho jata hai on usage
now we have to create a launch templeate
from instance there is a option launch templeate

hostname-app-launch-template



when we are creating the system we want our port to be exposed on 3000



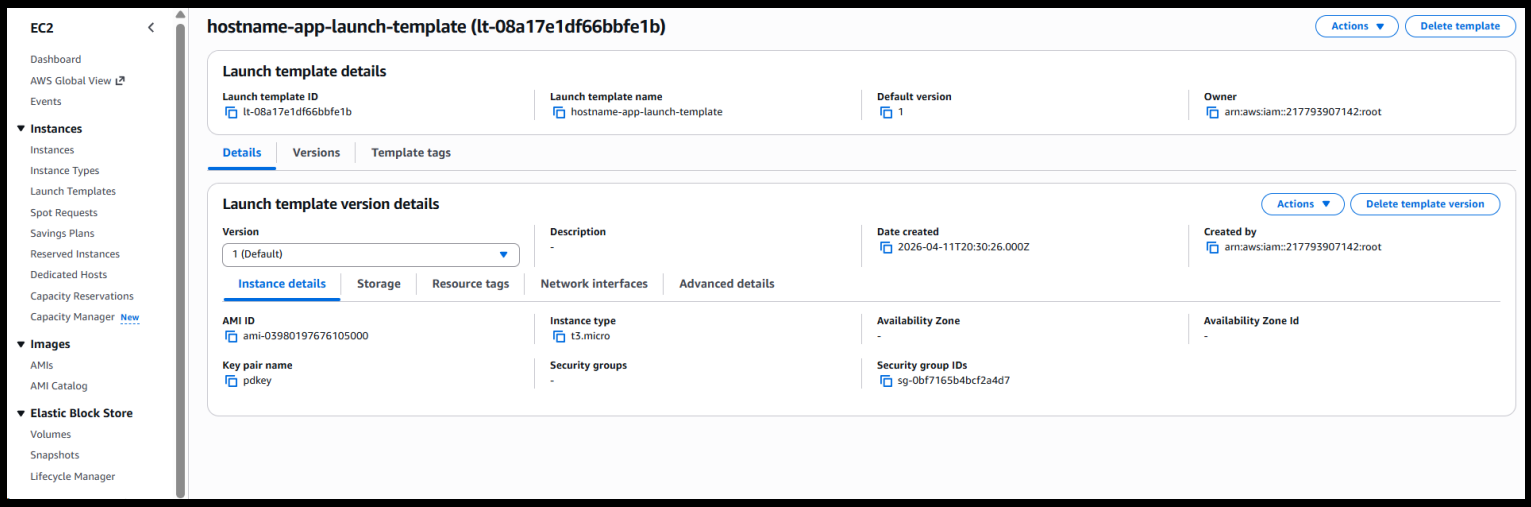
created a new security group

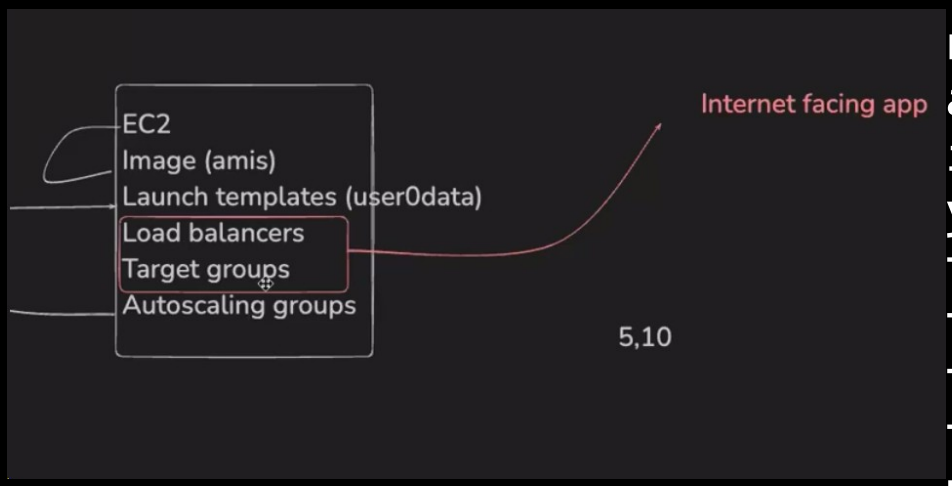
now this is exposed to whole world for now and later we will expose to the loadbalancer only

THESE ARE JUST BASICS BEFORE LEARNING THE KUBERNETES SO WE ARE JUST LEARNING THIS
now in the advanced section add this thing in the user data as the server will start but will not start the code on its own so we have to add the commands in that
cd ~/ASG
pm2 start --interpreter /home/ubuntu/.nvm/versions/node/v24.14.1/bin/bun bin.ts
KIND OF SCRIPT WHICH RUNS

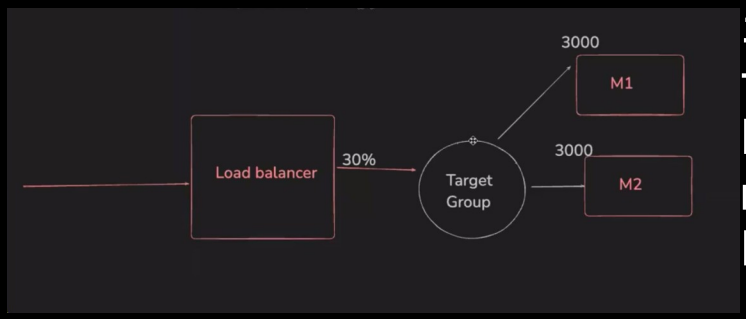
BASHRC FILE IMPORTANCE
THIS FILES CONTAIN LIKE THE NODE PATH FOR EG WHEN WE RUN NODE --VERSION HOW IT BRINGS THE VERSION??
IT HAS THE PATH STORED WHICH IT ACCESS AND GIVES US THE ANSWER IN THIS FOR MANY VALUES THE PATHS ARE STORED WHICH HELPS IN ACCESS THAT
EG: FOR NODE, NPM, BUN, NPX ,PM2 ETC…
IT CONTAINS THE PATH
Now the point is the bashrc file is loaded whenever the terminal is loaded but in this the template its not started so to fix that we have to add that file also
#!/bin/bash
cd ~/ASG
export
npm install -g pm2
PATH=$PATH:/home/ubuntu/.nvm/versions/node/v24.14.1/bin/node
pm2 start --interpreter /home/ubuntu/.nvm/versions/node/v24.14.1/bin/bun bin.ts
in this the last second command is for adding the npm path to the path variable which already exist

Launch Template is created now creating the Target Groups



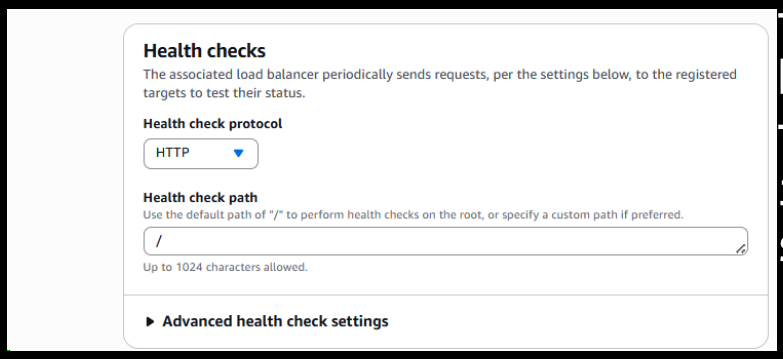


now see if your application is not internet facing so you need to just say to autoscaling groups that use this template and create the machines this is mainly can be used for traning ml algos etc
for the internet facing application we have to use the load balancers and target groups which we will create now so before creating autoscaling group we have to create the above two things
Abh the point is load balancer ko kaise pata ki konse machine ko load bhejna hai??
this is done using target groups this thing help to send the request to proper system



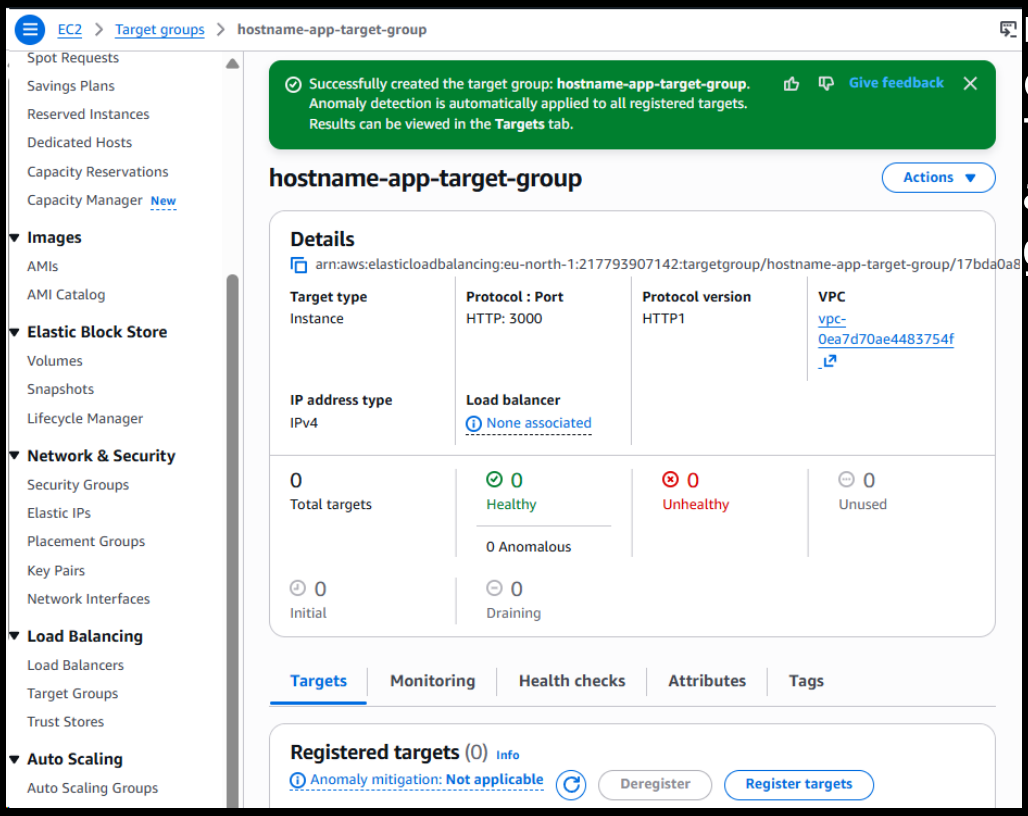
its just like ki joh load target group ko pata hota hai haar system kae barae mai fhir joh load balancer hai woh batata hai group ko

NOW FIRST CREATING A TARGET GROUP
select instance as target type as we are sending to that only
than protocol http and port 3000



this is imp as this url is hit by the load balancer through this only it identifies whether the system is running or not

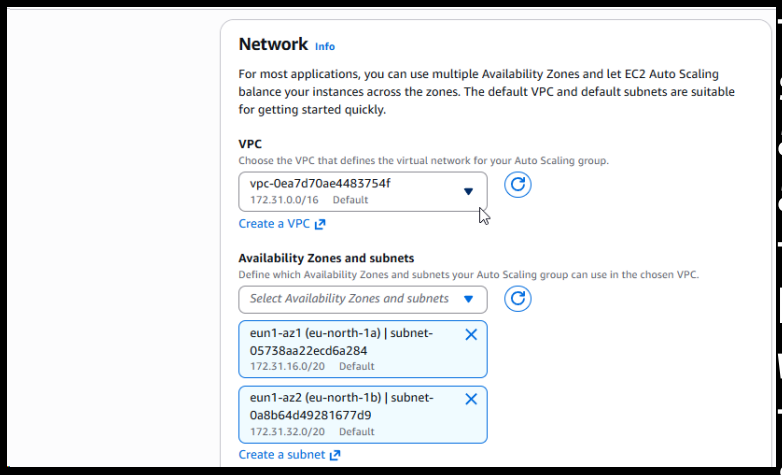
after creation of this we can set the target groups but we will not do that because if me mention that then we have to manually assign every time that it should use this system but for us this work will be done by the load balancer



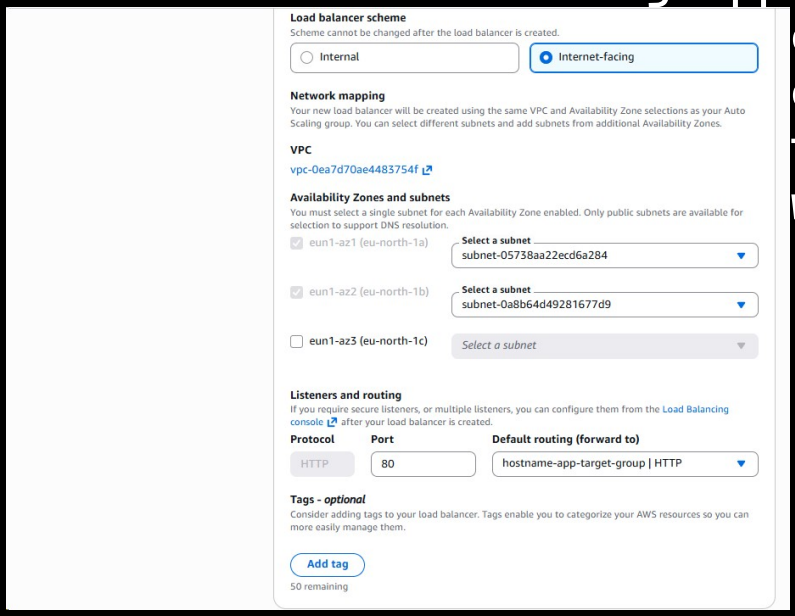
now we have to create loadbalancer and auto scaling group

Thankfully while creating the auto scaling group we can also create the load balancer in between the process only

now in the auto scalling group select the template
now in the next step we can also override the template but not required and we can also create
VPS = virtual private cloud

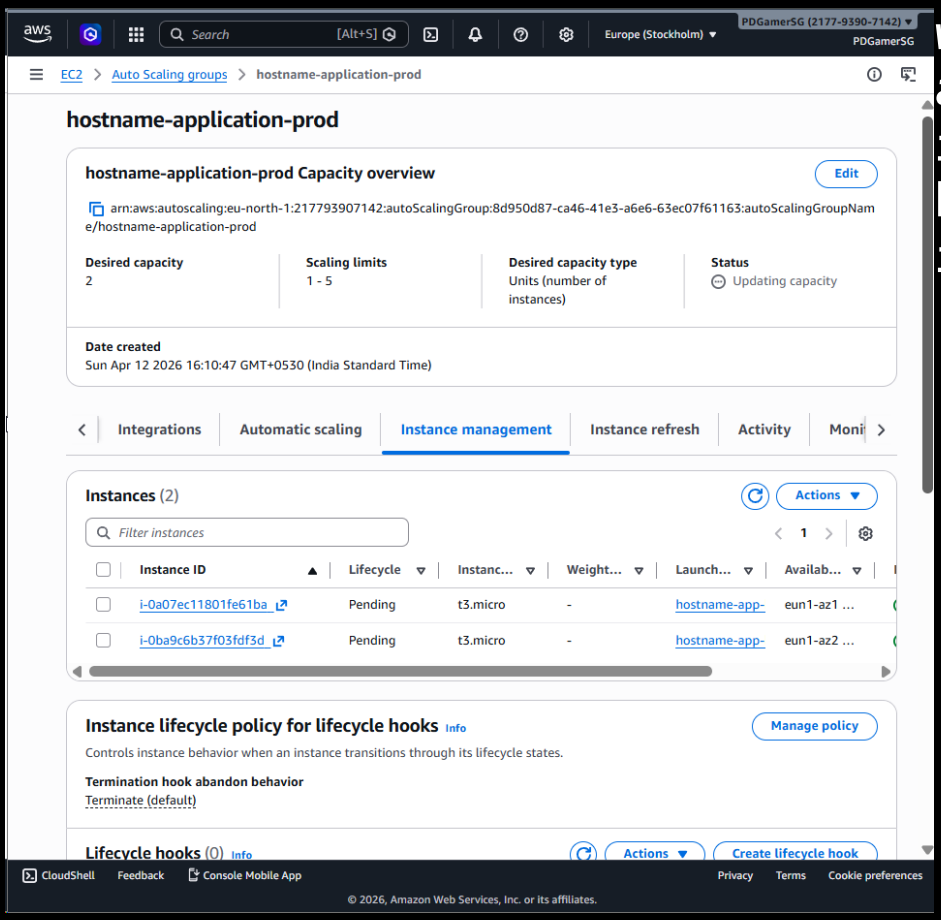


this availability zones we shoulde choose as many as availabile for http application but not for the encoding application because in that the data working can take long time to process and if the system is constantly switch between the users then its a problem so for transcoding or youtube you should not do that
this available zones is like where all your code will be running
third step create a new load balancer
create a internet facing application



choose the target group currently listening on the port 80 and later we will add 443 also

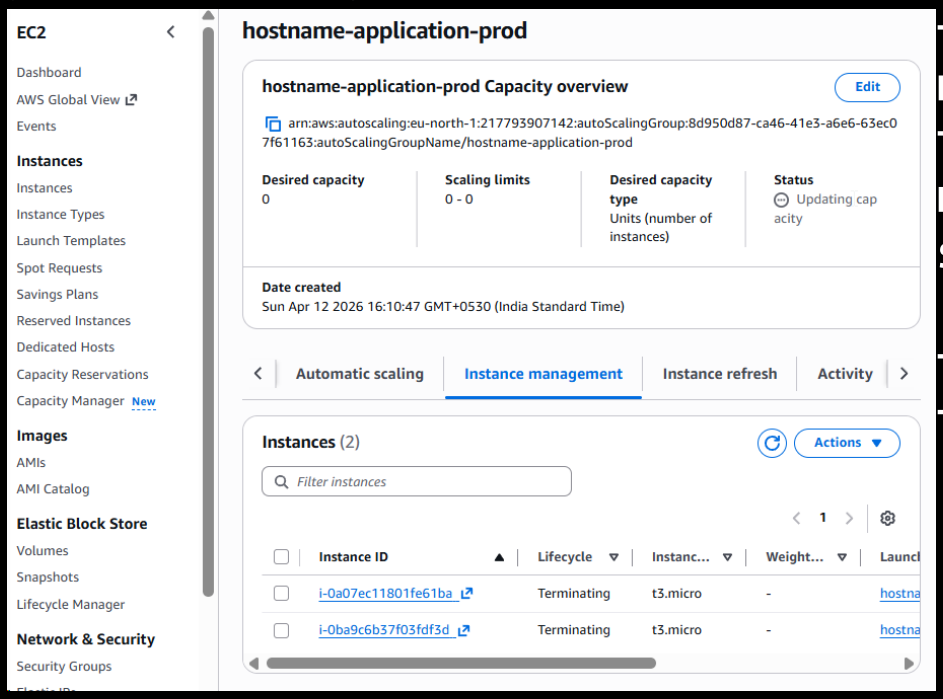
now mention the desired properties for system
Desired capacity type 2
Min desired capacity 1
Max desired capacity 5
NOW JUST CREATE LETS GOOOOO



we can see in the auto scaling group instance management how many instance it has started

as my desired capacity is 2 if I crash one of them it will automatically start the new server damn
we can also ssh into the system
ssh -i pdkey.pem ubuntu@16.171.54.217
and already in that the ASG folder is present see that using ls
the issue is now we created the machine but the code is not running on its own we can check that using logs
do in the machine
vi /var/log/cloud-init-output.log
now to make changes you can do is click on launch template click on action click on modify template then change the code in the advanced section and then create the template
now go in the ASG ,click on the asg name and then click on the edit of the launch template and change the version of the template from 1 → 2

now to change the code what we do it either kill it yourself or just



from this edit option make it 0 0 and in the instance management it will show terminating

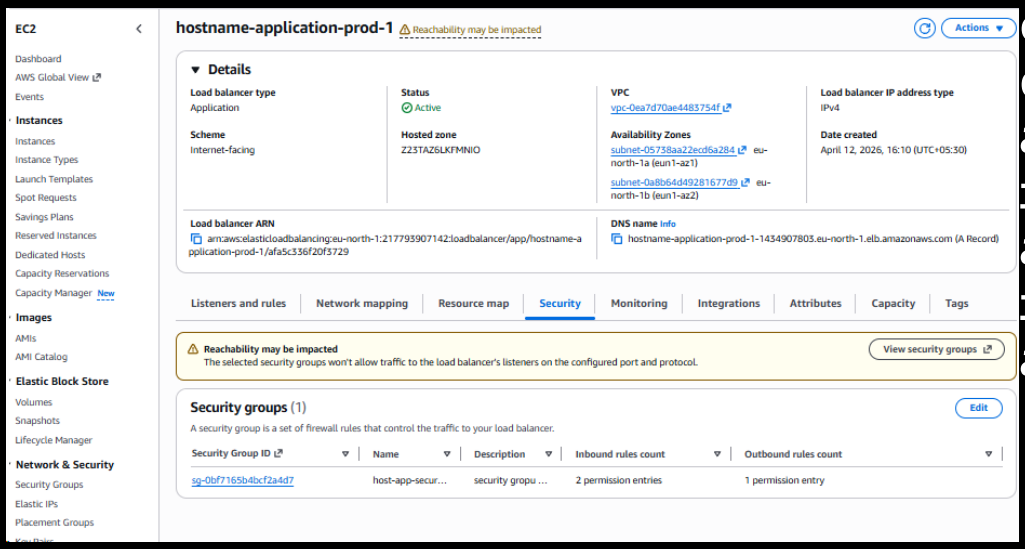
then again update it to 2 2 5

FLOW FOR UPDATING THE TEMPLATE
create the template then go to ASG update the template to latest change the number to 0 0 0 and then wait for instance to terminate and then change them again to 2 2 5

```
#!/bin/bash
cd /home/ubuntu/ASG

export
PATH=$PATH:/home/ubuntu/.nvm/versions/node/v24.14.1/bin/

export NVM_DIR="/home/ubuntu/.nvm"
source "$NVM_DIR/nvm.sh"

npm install -g pm2
pm2 start bin.ts --interpreter bun
```

this script is running lets gooooo

now after this if you try to access the page using the load balancer it will not work it also has the security groups change that tooo



click on the name of security group add the inbound rule
allow http request from anywhere

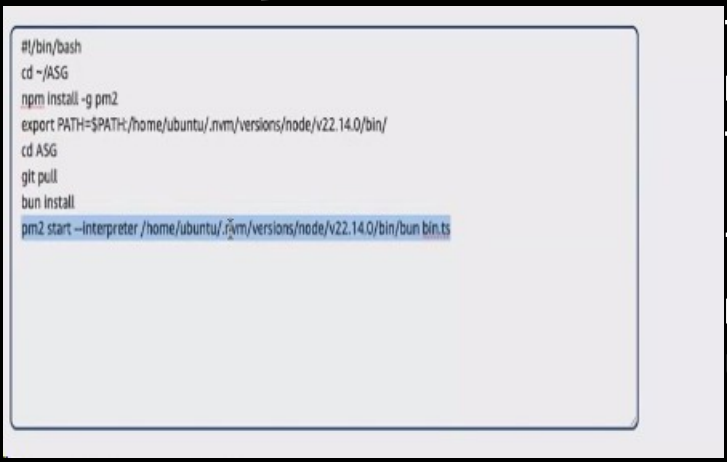
loadbalancer url/host
shows which machine is used like it changes everytime

first create template
then create target group
then create ASG
in that create load balancer
in the ASG integration it should show the target group
and then using the load balancer ip you can access or it may give security port issue fix that and enjoy

NOW IMAGINE IF LOAD IS INCREASED JUST INCREASE THE NUMBER OF SYSTEM IN THE ASG AND ENJOY BUT THIS IS A MANUAL WAY OF DOING

SO HOW TO DO THE AUTOMATIC SCALING BASED ON THE LOAD? YOU CAN DO IT USING **SCHEDULED ACTIONS** LIKE BASED ON THE TIME PERIOD IN THE MORNING REDUCE THE MACHINES IN THE EVENING INCREASE THE SYSTEMS

OR USE **DYNAMIC SCALING**
based on the cpu usage it will downscale or upscale or inside this you can also use simple scaling in which based on the usage increase it will set a alarm that assign 20 machine like that



to auto deploy code to the machine ugly way of doing this
based on CICD Pipeline like raat ko drain the machines and make it back to up and all the servers will have the latest code

delete everything if not required AMI, loadbalacer, target groups, instances etc