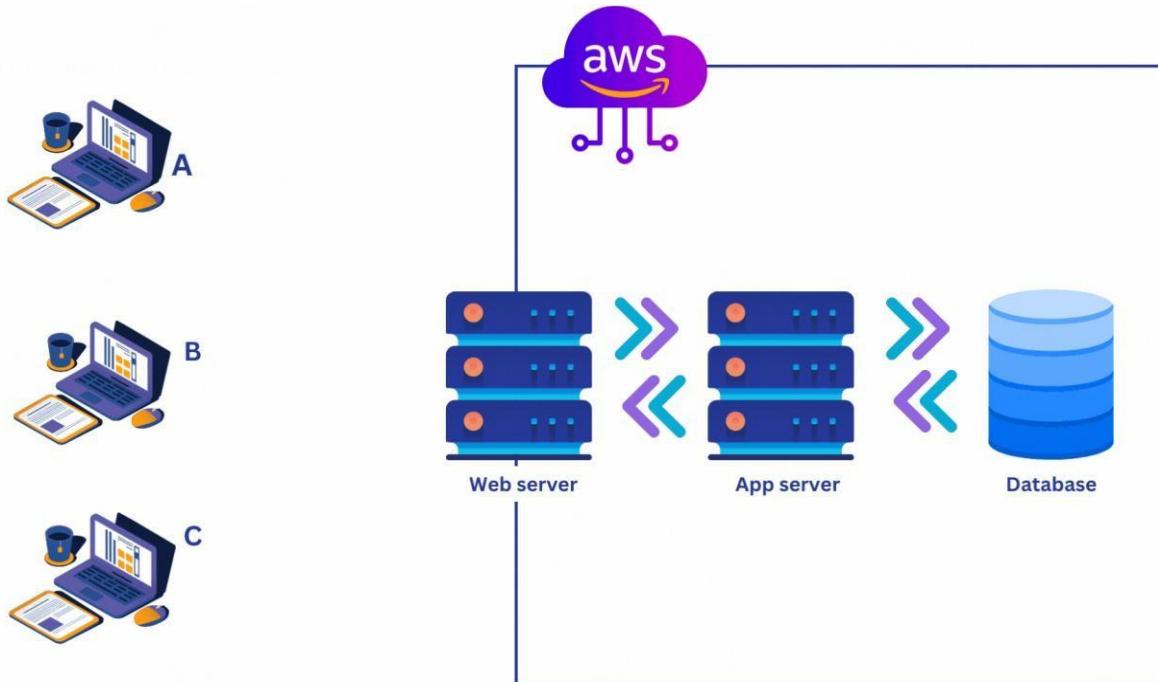


Highly Available AWS Multi-Region 3-Tier Architecture



II Prerequisites

- AWS Account
- Basic knowledge of Linux

III List of AWS services

- Amazon CloudFront
- Amazon Route 53
- Amazon EC2
- Amazon Autoscaling
- Amazon Certificate Manager
- Amazon Backup service
- Amazon RDS
- Amazon VPC
- Amazon WAF
- Amazon CloudWatch

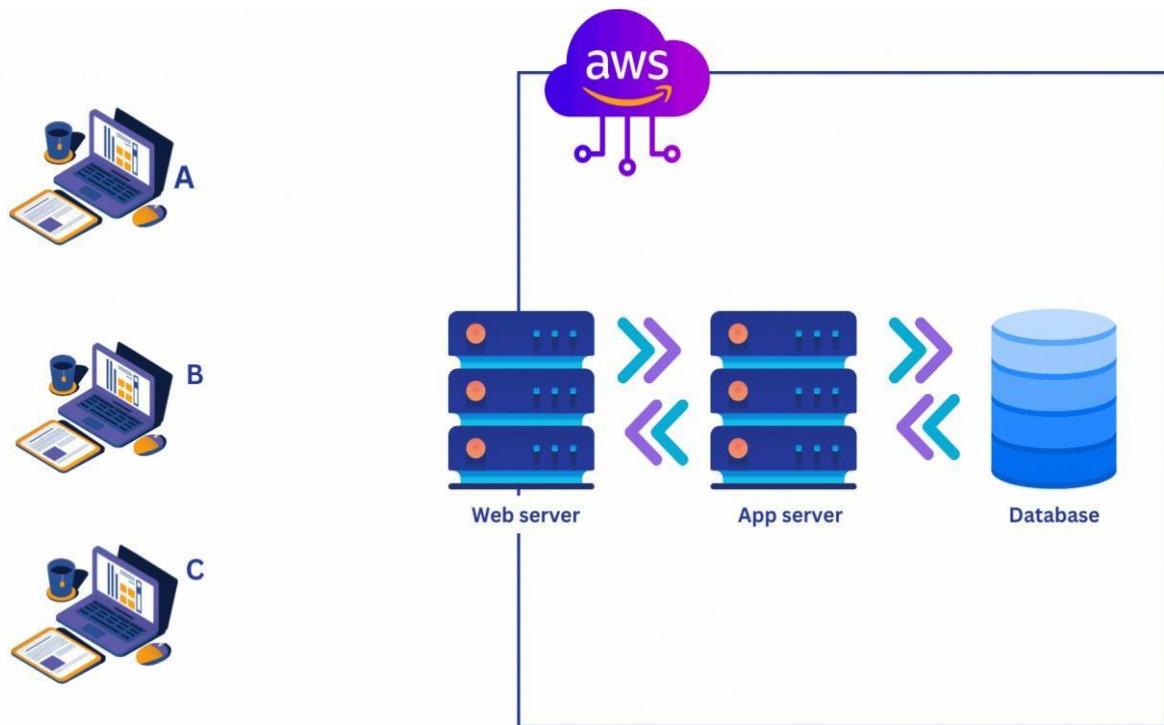
IV Plan of Execution

- What is three-tire architecture
- The architecture of the project
- A step-by-step guide with screenshots
- Testing
- Resource cleanup
- conclusion

What is Three-tier architecture

Three-tier architecture is a software architecture pattern that separates an application into three layers.

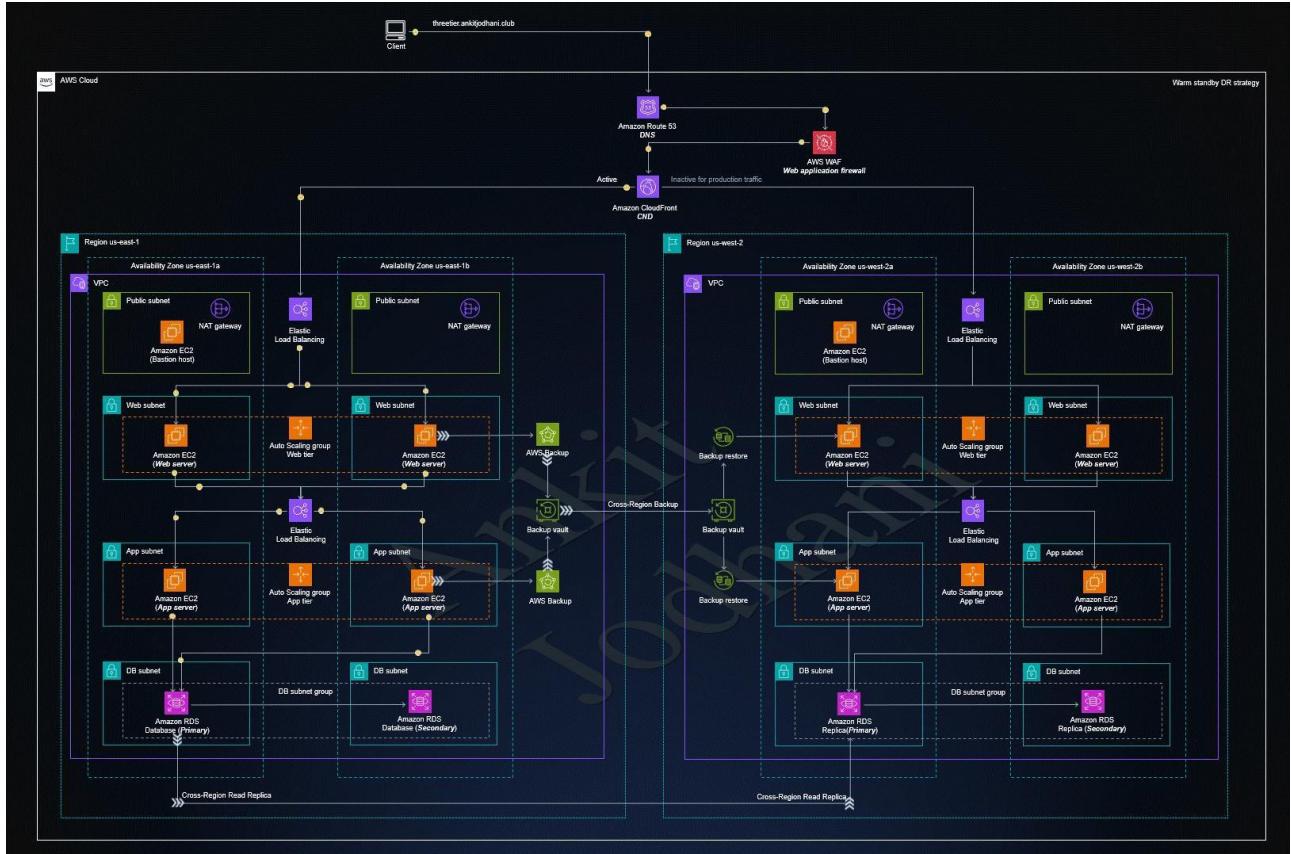
- ◆ Presentation layer handles user interaction
- ◆ Application layer(backend logic) processes business logic and data processing
- ◆ Data layer (database) manages data storage and retrieval



Each layer has distinct responsibilities, allowing for modularity, scalability, and maintainability. This architecture promotes the separation of concerns and facilitates easy updates or modifications to specific layers without impacting others.

Architecture of the Project

Let's see the architecture of today's project. we are going to follow a goal-driven approach that helps us to put in minimum effort and gain lots of results. it's very important to understand what we are going to build and to understand you can follow the below architecture. I request you please go through it once. it helps you a lot while building this project.



7 A Step-by-step guide

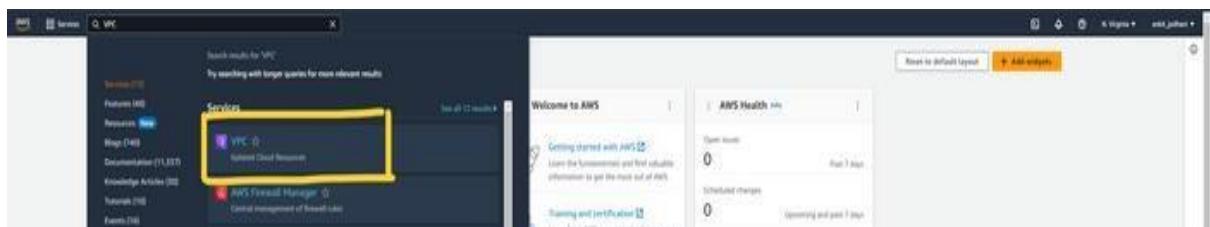
we are following a **Warm standby Disaster recovery strategy** so we are going to utilize two regions during our deployment. **us-east-1** AKA North Virginia as **primary** and **us-west-2** AKA Oregon as **secondary** or DR.

◆ VPC

firstly we are going to set up VPC in both regions to isolate our resources from the internet. The below image contained all the subnets, their IP range, and their uses. you can use your own VPC setup if you have a better idea. and if you are a beginner, please create VPC as I have shown below.

VPC	172.20.0.0/16			
Availability zone	us-east-1a / us-west-2a		us-east-1b / us-west-2b	
uses	name of the sbunet	subnet ip range	name of the subnet	subnet ip range
ALB frontend	pub-sub-1a	172.20.1.0/24	pub-sub-2b	172.20.2.0/24
ALB backend				
Web servers	pri-sub-3a	172.20.3.0/24	pri-sub-4b	172.20.4.0/24
App servers	pri-sub-5a	172.20.5.0/24	pri-sub-6b	172.20.6.0/24
Databases	pri-sub-7a	172.20.7.0/24	pri-sub-8b	172.20.8.0/24

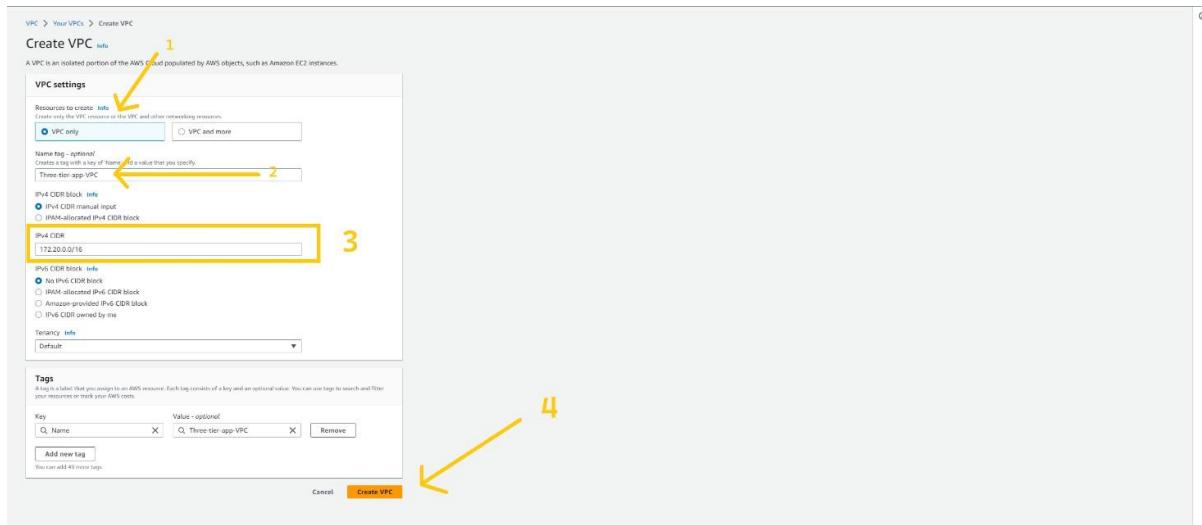
please log in to your AWS Account and type VPC in the AWS console. and click on VPC service.



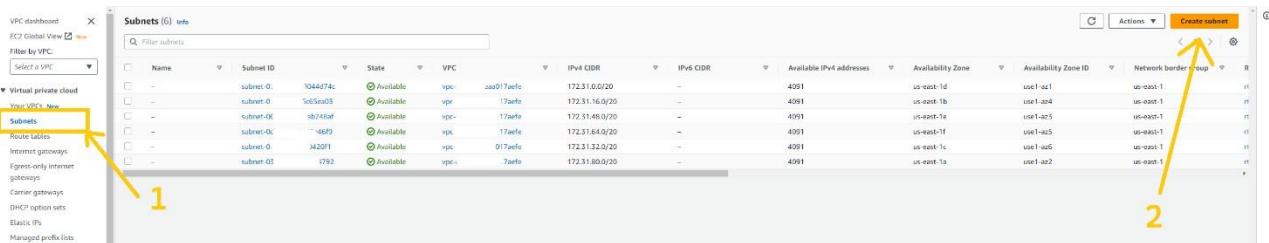
click on Your VPC's button on the left and then click on Create VPC the button on the top right corner of the page



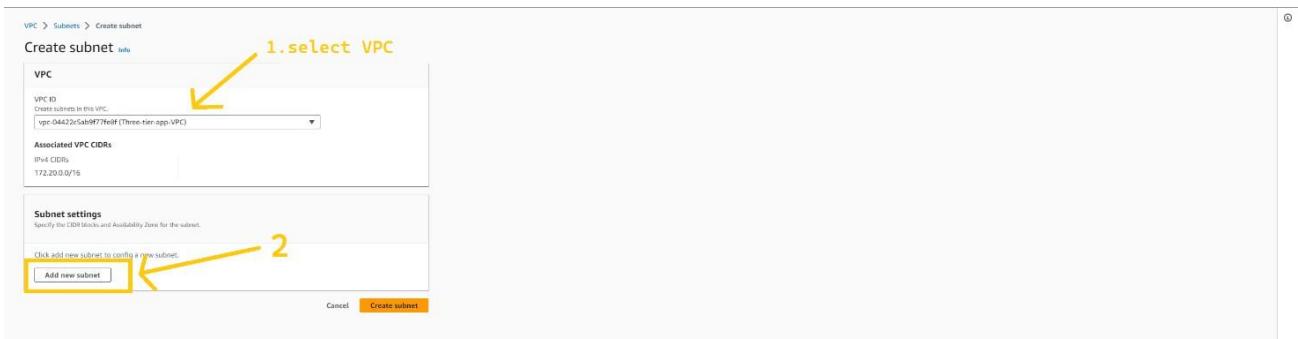
here we can see the form where we can fill the configuration of VPC. please enter the name that you want to keep and the IPV4 CIDR block. in my case CIDE block is 172.20.0.0/16.



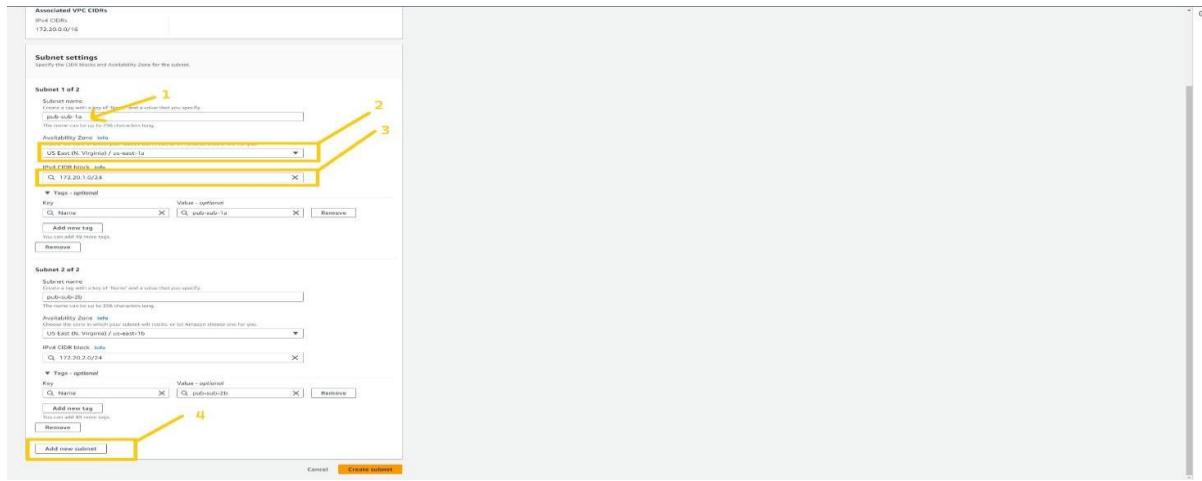
now click on the subnet button which is located on the left side and then click on the Create subnet button on the top right corner of the page.



please remove the default VPC ID and choose the VPC ID that we have just created in the VPC ID field. and click on the Add Subnet button at the bottom.



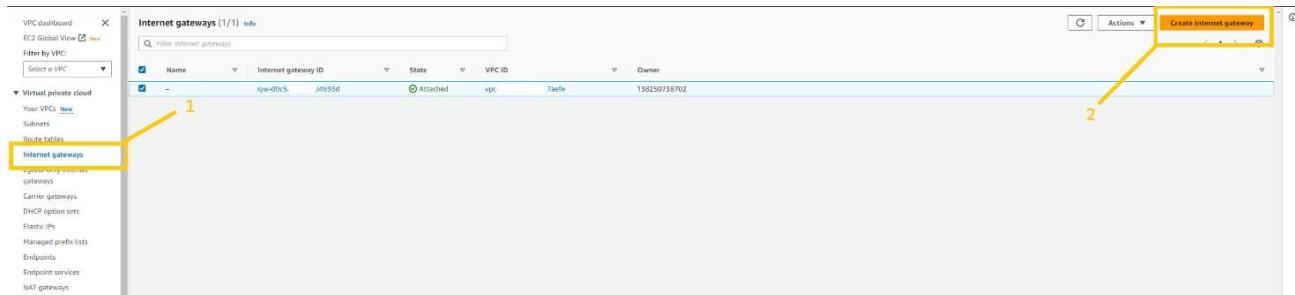
now we need to configure our subnets. Again, you can use the VPC configuration image that I shared earlier on the blog to get the IP range and to know which subnet will be used for what purpose. we are going to create a total of 8 subnets of which 2 of them are public and the rest of 6 subnets are private. you can create a subnet as I have shown in the below image. after adding all the subnets click on Create subnet button.



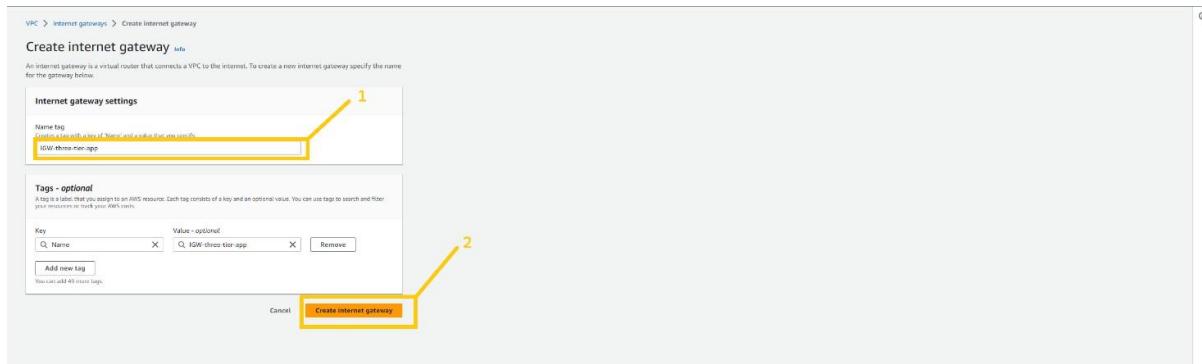
after the successful creation of all 8 subnets, they look like this. you can verify with my subnets.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4 addresses	Availability Zone	Availability Zone ID	Network back
pub-sub-1a	subnet-0f8474f27932bcf63	Available	vpc-04422c5db9ff77ef	172.20.0.0/16	-	251	us-east-1a	use1-az2	us-east-1
pub-sub-1b	subnet-0aa0510956d6f02	Available	vpc-04422c5db9ff77ef	172.20.0.128/16	-	251	us-east-1b	use1-az1	us-east-1
pri-sub-1a	subnet-02fdec1f60203d611	Available	vpc-04422c5db9ff77ef	172.20.0.7/24	-	251	us-east-1a	use1-az2	us-east-1
pri-sub-1b	subnet-0423624d9e14b6f6	Available	vpc-04422c5db9ff77ef	172.20.0.8/24	-	251	us-east-1a	use1-az2	us-east-1
pri-sub-2a	subnet-023e2415d2ed0f599	Available	vpc-04422c5db9ff77ef	172.20.0.1/24	-	251	us-east-1b	use1-az1	us-east-1
pri-sub-2b	subnet-0c190056fe5213a	Available	vpc-04422c5db9ff77ef	172.20.0.2/24	-	251	us-east-1b	use1-az2	us-east-1
pri-sub-3a	subnet-02fdec1f60203d611	Available	vpc-04422c5db9ff77ef	172.20.0.7/24	-	251	us-east-1a	use1-az2	us-east-1
pri-sub-3b	subnet-0423624d9e14b6f6	Available	vpc-04422c5db9ff77ef	172.20.0.8/24	-	251	us-east-1a	use1-az2	us-east-1

now we are going to create Internet Gateway also known as **IGW**. it is responsible for communication between VPC, VPC's public subnet with the Internet. without IGW we won't be able to communicate with the Internet. so let's create that. click on the internet gateways button at the left panel. and then click on the Create Internet gateways button on the top right corner of the page.



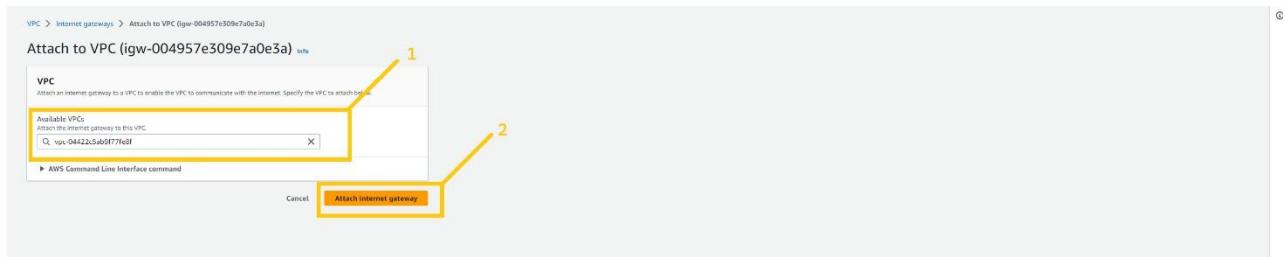
give any name you want to give to IGW. and click on Create Internet gateway button.



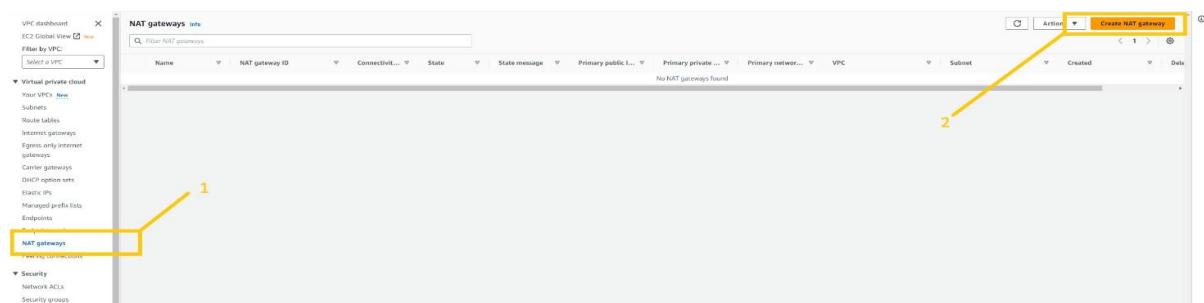
after creating an internet gateway, we need to attach it with VPC to use it. for that click on the Action button. here you can see the drop-down list. please select the option Attach to VPC.



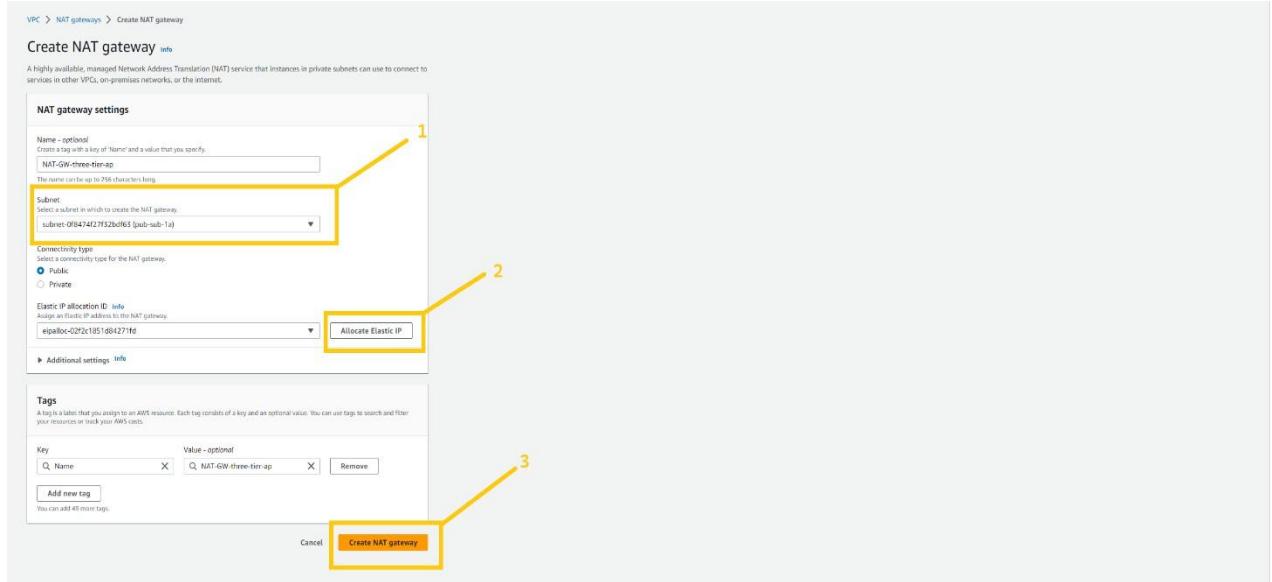
please select VPC that we have created just now from the Available VPC list. and then click on the Attach Internet gateway button.



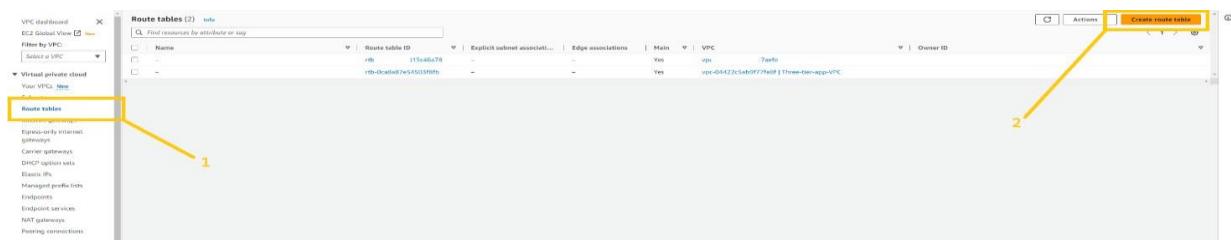
Now we need to create a NAT gateway. NAT gateway is responsible to connect resources that are in the private subnet to communicate with the internet. all the resources which will be there in a private subnet will communicate to the internet through the NAT gateway. we will keep the NAT gateway in the public subnet so that it can access the internet. NAT gateway is a chargeable resource. so, you will be charged by AWS as long as you keep it up. Now to create a NAT gateway click on the NAT gateways button on the left panel of the web page. and then click on the Create NAT gateways button in the top right corner of the page.



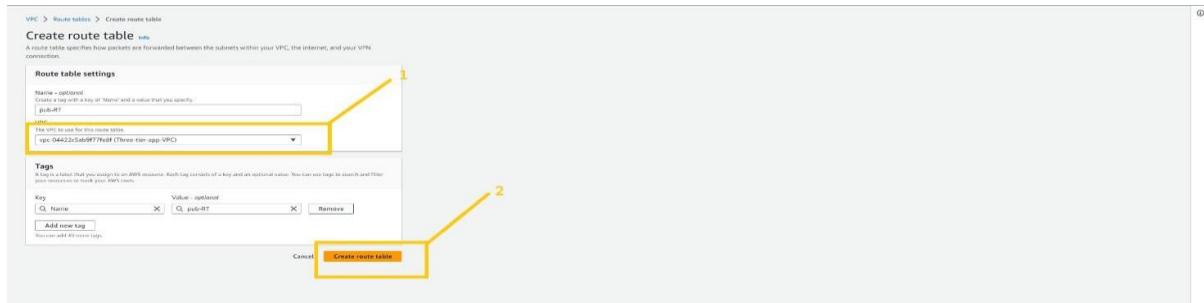
give any name you want to give to the NAT gateway, but be cautious with selecting a subnet. **You have to select one of the public subnets among the two, either pub-sub-1a or pub-sub-2b.** then click on the Allocate Elastic IP button to allocate Elastic IP. and then click on the Create NAT gateways button. NAT gateways creation takes 2-4 minutes.



Now we need to have a route table to handle traffic for public subnet and private subnet and for that, we need to create a Route table. we are going to create two route tables one for the public subnet and another one for the private subnet. first, we are going to create RT for the public subnet. so, click on the Route table button which you can see on the left panel. and click on the Create Route table button on the top corner of the page.



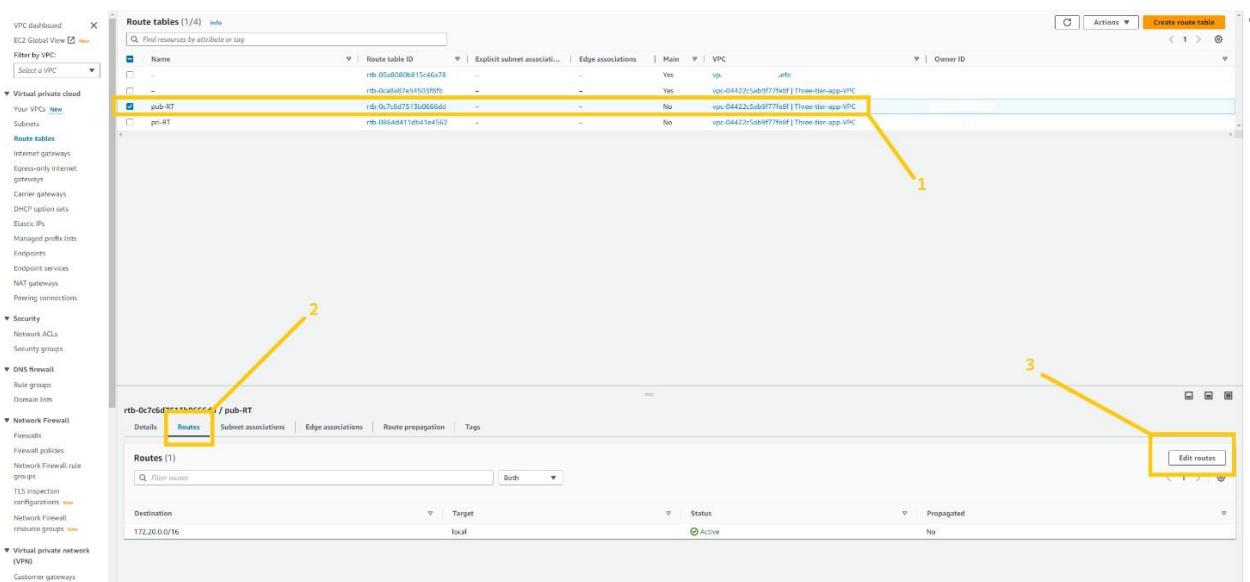
give a name to your RT such as Pub-RT. please give a name that is appropriate for resources then it will be easy to organize the things. make sure you select the correct VPC. and then click on the create route table.



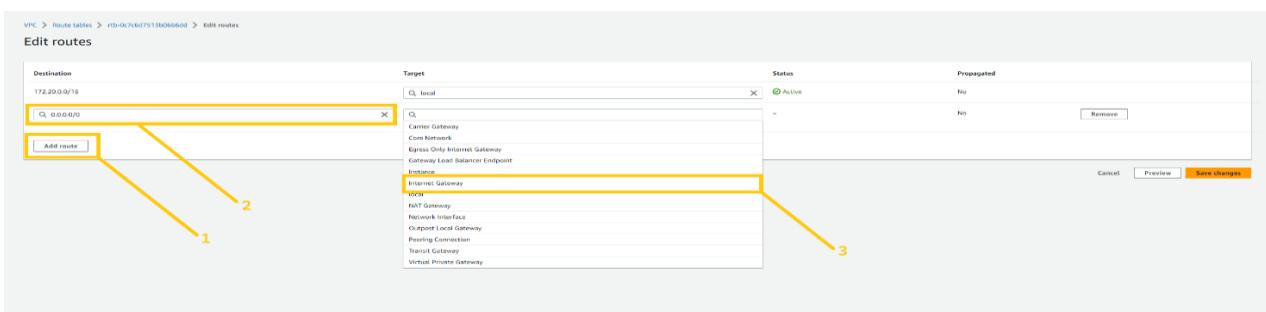
let's create RT for the private subnet.



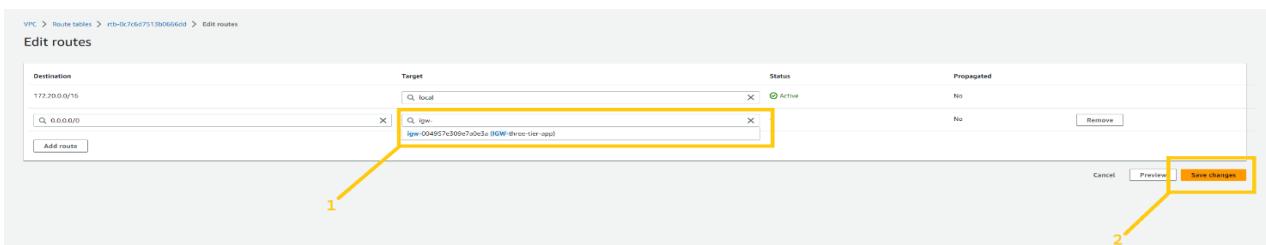
Now, we need to do some association with both RTs so select **Pub-RT** and click on the Routes tab at the bottom and then click on the edit route button.



click on the Add Route button. and select 0.0.0.0/0 in the destination field. and then click on the Target field. As soon as you click on the Target field one drop-down will open and here you have to select Internet gateway, shown in the below image.



here you can see the IGW that we created earlier. select that IGW and click the save changes button.



keep Pub-RT selected and click on the Subnet associations tab next to the Routes tab. and then click on the Edit subnet associations. as shown in the below image.

The screenshot shows the AWS VPC dashboard with the 'Route tables' section selected. A specific route table, 'pub-RT', is highlighted with a yellow box. Below it, the 'Subnet associations' tab is selected. At the top right of the table, there is a button labeled 'Edit subnet associations'. A yellow arrow points from the text 'click on the Edit subnet associations.' to this button.

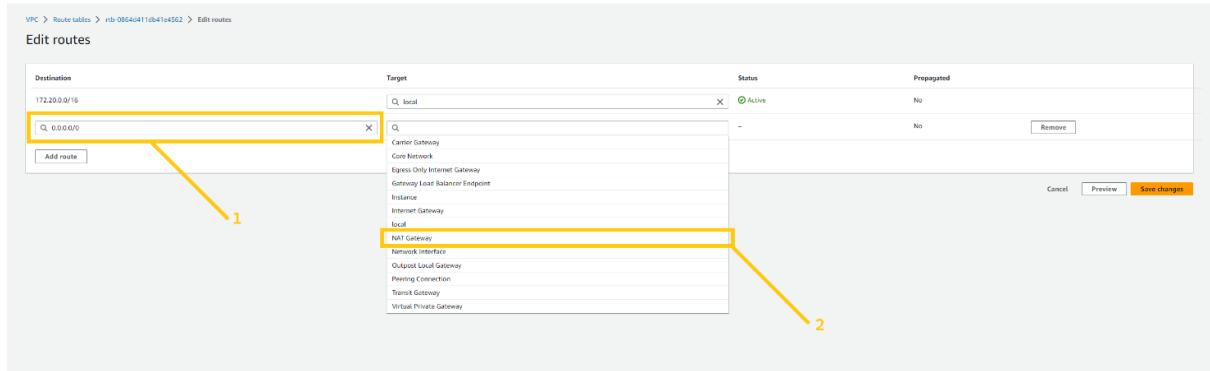
now select both public subnets. **pub-sub-1a** and **pub-sub-2b**. and click on the save associations button.

This screenshot shows the 'Edit subnet associations' dialog for the 'pub-RT' route table. In the 'Available subnets' list, two subnets are selected: 'pub-sub-1a' and 'pub-sub-2b'. A yellow arrow points from the text 'now select both public subnets.' to these subnets. At the bottom right of the dialog, there is a 'Save associations' button, which is also highlighted with a yellow box. A second yellow arrow points from the text 'click on the save associations button.' to this button.

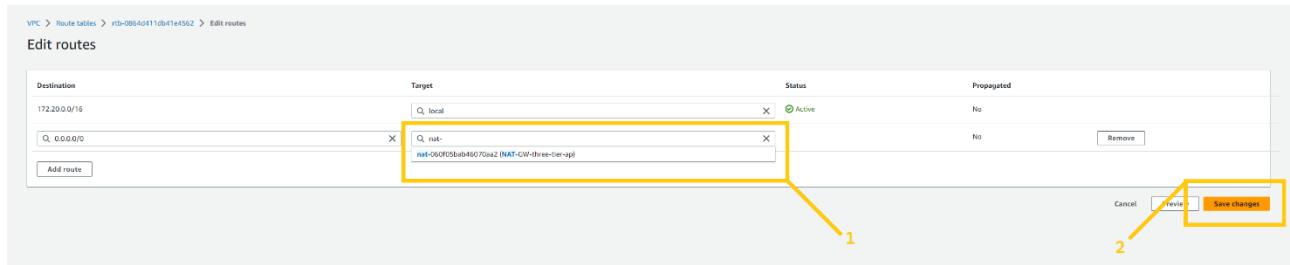
now we have to do the same thing for the Pri-RT as well. but there is one slight change. let me show you. Please select Pri-RT and click on the Routes tab at the bottom of the page.

The screenshot shows the AWS VPC dashboard with the 'Route tables' section selected. A specific route table, 'pri-RT', is highlighted with a yellow box. Below it, the 'Routes' tab is selected. At the top right of the table, there is a button labeled 'Edit routes'. A yellow arrow points from the text 'click on the Routes tab at the bottom of the page.' to this button.

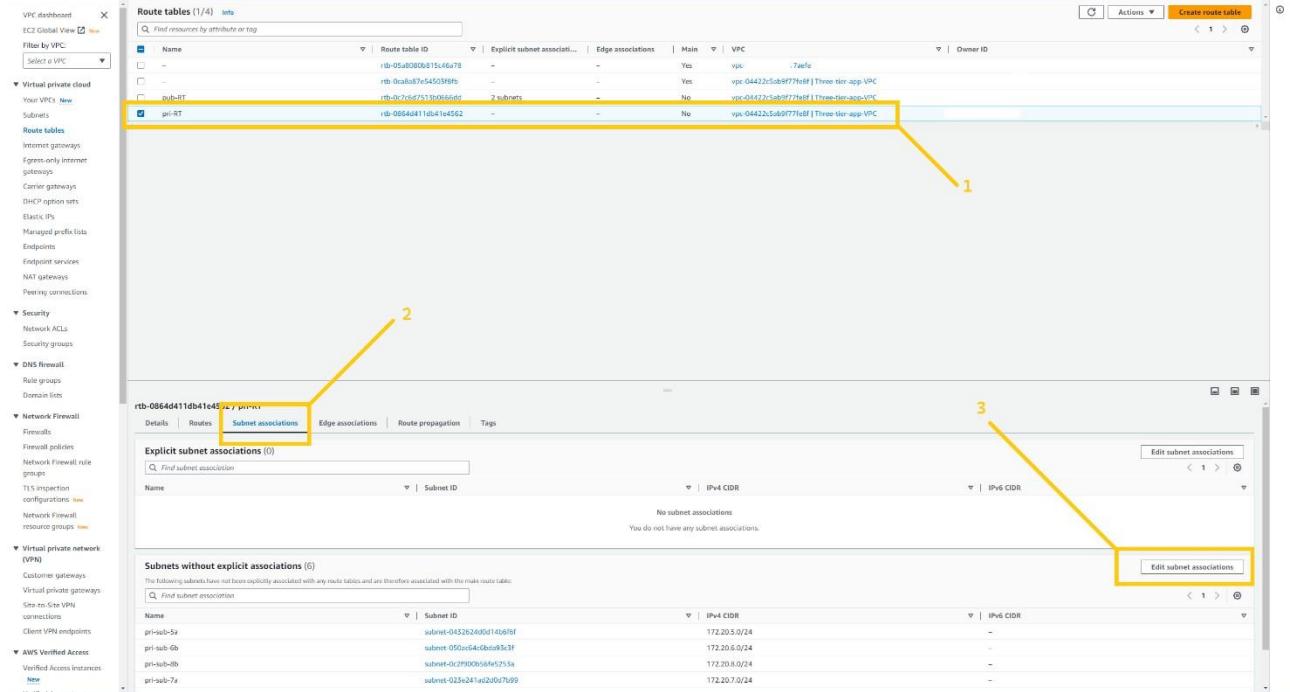
Here please select 0.0.0.0/0 in the destination field and click on the target. As soon as you click on the target you will see the drop-down list. Please select NAT gateway from the drop-down list. As shown in the below image.



select the NAT gateway that we have just created. and click on the save changes button.



keep Pri-RT selected and click on the subnet associations tab at the bottom next to the Routes tab. And then click on the Edit route associations button.



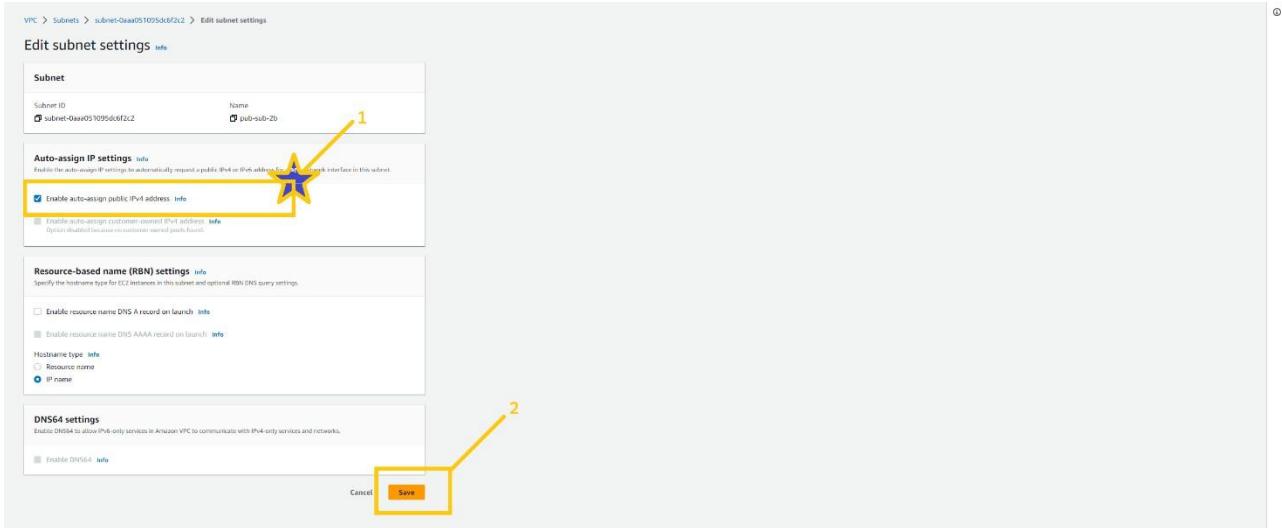
Here you can see the same situation as we saw before. But here we are going to select all the 6 private subnets. And then click on the save association button.

Before we move ahead, I want to change the settings of VPC and two public subnets. So just click on the Your VPC button on the left panel and select VPC that we have created and click on the action button and there you will see the drop-down menu. Select the Edit VPC setting button. As shown in the image.

And here please enable **Enable DNS hostname** checkbox by clicking on it. and then click on the Save button

Please go to the subnet page and select the public subnet and click on the action button and then choose the Edit subnet setting button from the drop-down list.

Here you have to mark right on **Enable public assign public IPV4 address**. And then click on the save button



And here we are done with VPC configuration in the primary region. In my case **us-east-1 (N.Virginia)**. But we have to do the same setup in the secondary region as well. As you know I am going to use the **us-west-2 (Oregon)** as my second region AKA Oregon.

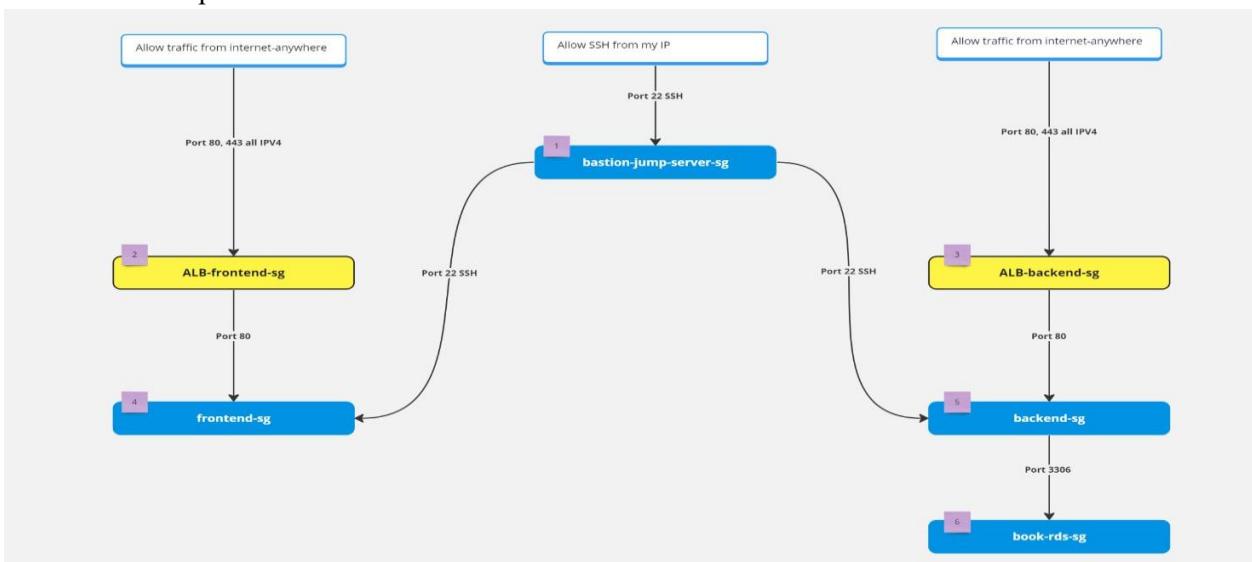
Your task is to set up VPC in the secondary region. All the setup is completely similar. You just have to change the region. And please do VPC set up in the secondary region.

I hope you did the setup. Now let's move ahead

◆ Security Groups (SG)

Security groups are very essential part of the infrastructure. Because it can secure the resources in the cloud. SGs are a kind of firewall that allow or block incoming and outgoing traffic. SGs are applied to the resources like ALB, ec2, rds, etc. One resource can have more than one SG.

So let's first understand. How SG will be used in our architecture and how we are going to apply that. Please see the below image you will get all the ideas. Which resource depends on what. And what are the port numbers we need to allow etc...



To create SG, click on the security groups tab on the left panel and here you will see the Security Groups button. Note that SGs are specific with VPC. So we can't use SG which is created in a different VPC. So when you create SG please make sure that you choose the right VPC. click on the crate

security button on the top right corner.

Name	Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules count	Outbound rules count
default	sg-0008749094f7ce8	default	vpc-aefc	default VPC security gr...		1 Permission entry	1 Permission entry
-	sg-034e5e622210fa12	default	vpc-044225a60f77fe8f	default VPC security gr...		1 Permission entry	1 Permission entry

We will create our first SG for **bastion-jump-server**. Give any name and description you want but please remove the default VPC and add VPC that we have just created. Then click on the Add rule button in inbound rules. And add SSH rule and add your IP in the destination. Please don't do anything with the outbound rule if you don't have a good understanding. And then click on the create security group button.

Now let's create SG for the **ALB-frontend**. Again steps are similar but add the rule HTTP AND HTTPS from anywhere on the internet because both ALB are internet facing. But please select the right VPC.

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: Name cannot be edited after creation.

Description:

VPC:

Inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Anywhere (IPv4)	0.0.0.0/0
HTTPS	TCP	443	Anywhere (IPv4)	0.0.0.0/0

Add rule

Outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	Custom	0.0.0.0/0

Add rule

Tags - optional

Tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key: Value - optional: Add new tag

Add new tag

You can add up to 10 more tags

Create security group

Create SG for **ALB-backend**. ALB-backend is also internet-facing. Again, allow HTTP and HTTPS from anywhere.

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: Name cannot be edited after creation.

Description:

VPC:

Inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Anywhere (IPv4)	0.0.0.0/0
HTTPS	TCP	443	Anywhere (IPv4)	0.0.0.0/0

Add rule

Outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	Custom	0.0.0.0/0

Add rule

Tags - optional

Tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key: Value - optional: Remove

Add new tag

You can add up to 10 more tags

Create security group

Create SG for frontend servers. Our frontend server will be in a private subnet so add the HTTP rule and select the source as **ALB-frontend-sg**. So only ALB-frontend can access the frontend server on port 80. You have to add one more rule SSH allows from **bastion-jump-server-sg**. So that the bastion host can log in to web servers.

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: Name cannot be edited after creation.

Description:

VPC:

Inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Custom	0.0.0.0/0
SSH	TCP	22	Custom	0.0.0.0/0

Add rule

Outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	Custom	0.0.0.0/0

Add rule

Tags - optional

Tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key: Value - optional: Remove

Add new tag

You can add up to 10 more tags

Create security group

Let's create SG for the backend server. Again, steps are completely similar to **frontend-sg**. You have to allow port 80 from **ALB-backend-sg** so that only **ALB-backend** can request to the backend server and add the rule SSH allows from **bastion-jump-server-sg**. So that the bastion host can log in to backend servers.

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: **backend-sg**
Description: **backend-sg**
VPC: **vpc-044225a0f77fe8ff**

Inbound rules

Type: **HTTP**, Protocol: **TCP**, Port range: **80**, Source: **Custom**, Destination: **sg-0efffbacfcf04**
Type: **SSH**, Protocol: **TCP**, Port range: **22**, Source: **Custom**, Destination: **sg-0efffbacfcf04**

Outbound rules

Type: **All traffic**, Protocol: **All**, Port range: **All**, Destination: **Custom**, Destination: **0.0.0.0/0**

Tags - optional

Key: **Q_Name**, Value: **Q_backend-sg**

Create security group

Lastly, we are going to create SG for RDS instance. Allow port 3306 MySQL/Aurora from **backend-sg** so that only the backend server will be able to access it. and no one else can access our database.

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: **book-rds-sg**
Description: **book-rds-sg**
VPC: **vpc-044225a0f77fe8ff**

Inbound rules

Type: **MySQL/Aurora**, Protocol: **TCP**, Port range: **3306**, Source: **Custom**, Destination: **sg-0a23c06e76ab642a**

Outbound rules

Type: **All traffic**, Protocol: **All**, Port range: **All**, Destination: **Custom**, Destination: **0.0.0.0/0**

Tags - optional

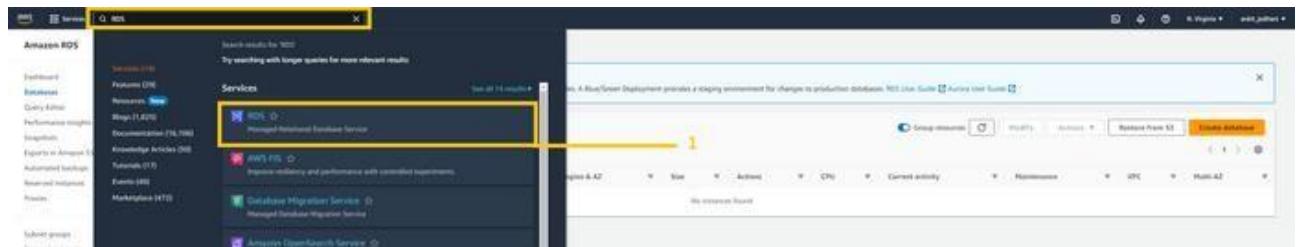
Key: **Q_Name**, Value: **Q_book-rds-sg**

Create security group

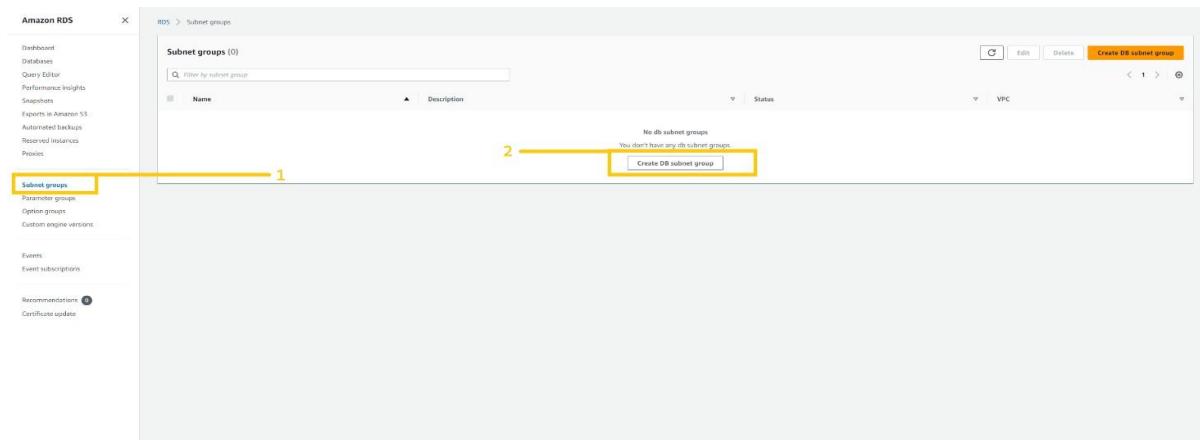
And here our SG setups are complete now. Your task is to do the complete same setup for the secondary region. In my case, it is **Oregon (us-west-2)**.

◆ RDS and Route 53

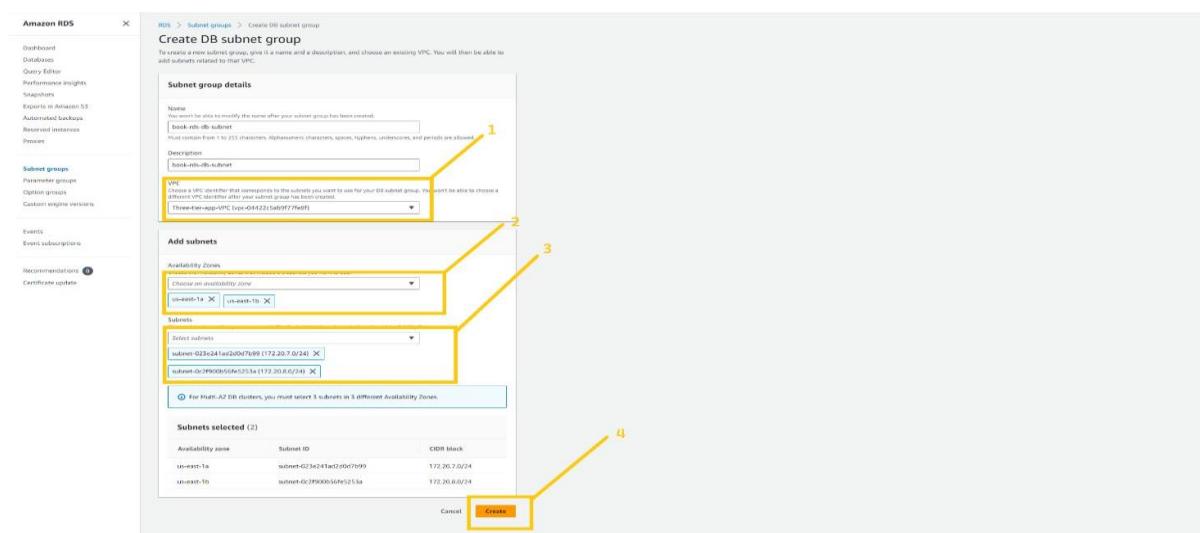
Now we are going to set up a database for our application. And for that, we are going to utilize the RDS service of AWS. So, let's head over to the RDS dashboard. Just search RDS in the AWS console. And click on the service.



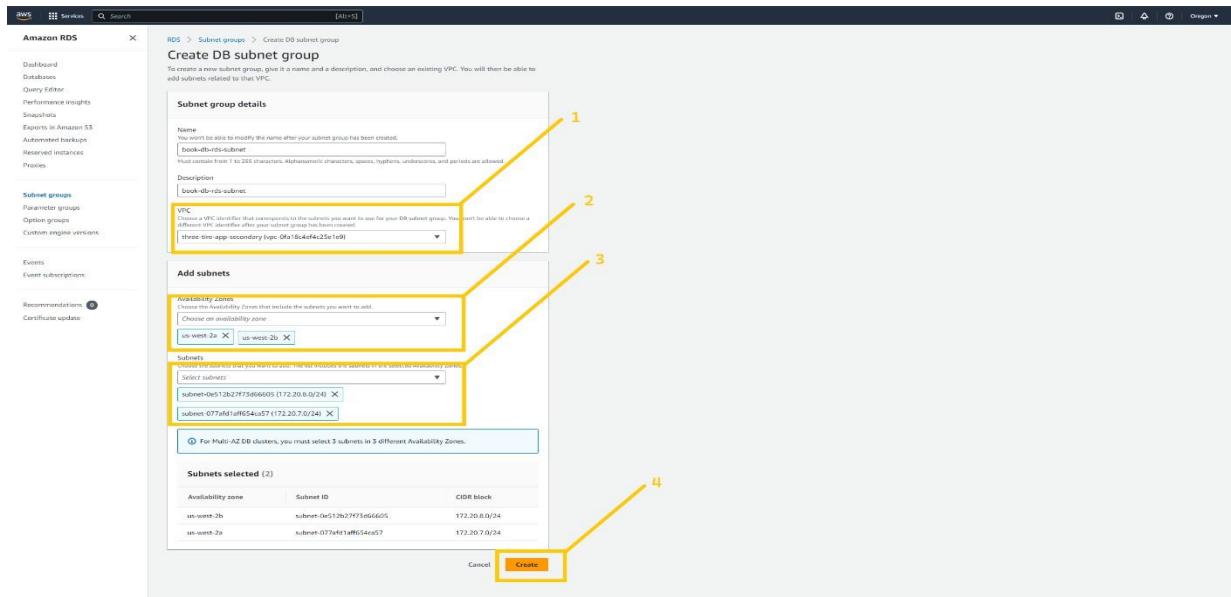
Now first we need to set up a subnet group. It specifies in which subnet and Availability zone our database instance will be created. So, click on the subnet group button on the left panel. And click on the button Create database subnet group which is in the middle of the web page.



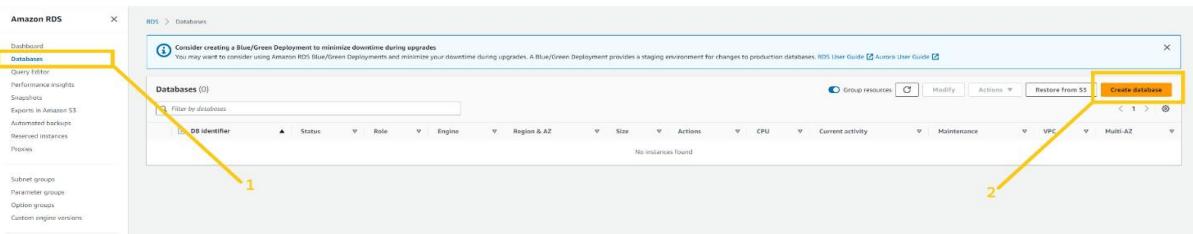
Here we can configure our VPC, subnet, and availability zone. Give any name to your subnet but make sure you select the correct VPC. and select AZs **us-east-1a** and **us-east-2b**. According to the architecture that I have shown you, our database will be in private subnet **pri-sub-7a** and **pri-sub-8b**. so please select as I have shown in the below figure. And then click on the create button.



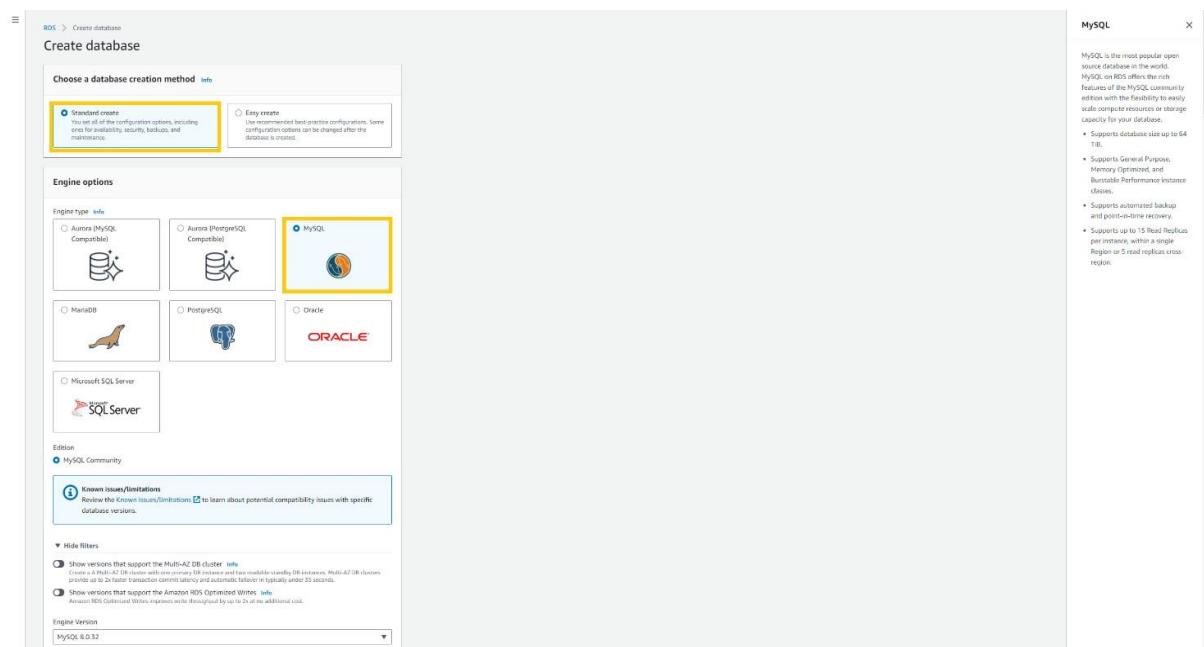
Note: we need to create a subnet group in the **Oregon** region as well. All the configuration is similar to the above, just need to change the Availability zone **us-west-2a** and **us-west-2b**



Now come to the **N. Virginia** region and here we are going to create a database. So, click on the database button on the left panel and then click on the created database button.



On this page, we can configure our database. Select stander create because I'm going to show you each and every step. select MySQL in the engine option because our application runs on MySQL database. If your app runs on other engines, you can select that one. Furthermore, you can select the engine version my application is compatible with MySQL version. But you can select according to the developer guild.



Scroll down, and select Dev/test as template. If you select the free tier then you won't be able to deploy RDS in a multi-availability zone. Select Multi-AZ DB instance from availability and durability option. In settings give any name to your database. In the credential setting give the username of the database in the Master username field and give the password in the Master password field. And then confirm the password below. Please do remember your username and password.

Again, scroll down, select Burstable class in the instance setting and select the instance type. Actually, it depends on your application uses. But for learning purposes, I am selecting t3.micro. now in storage type select General purpose (GP2) and allocate 22 GiB for database. Please uncheck the auto-scaling option to keep our costs low. And in the connectivity option please select the option according below screenshot.

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

Amazon RDS Optimized Writes - new [Info](#) [Edit](#)

Show instance classes that support Amazon RDS Optimized Writes

DB instance class: Info

Standard classes (includes m classes)

Memory optimized classes (includes r and x classes)

Burstable classes (includes t classes)

db.t3.micro
2 vCPUs | 1 GB RAM | Network: 2.085 Mbps

Include previous generation classes

Storage

Storage type: Info

General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage: Info
22 GB

(?) Provisioning less than 100 GB of General Purpose (SSD) storage for high-throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. Learn more [here](#).

Storage autoscaling: Info

Dynamic scaling support for your database's storage based on your application's needs.

Enable storage autoscaling
Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

Connectivity info

Compute resource:

Choose whether or not to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You will manually set up a connection to a compute resource later.

Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

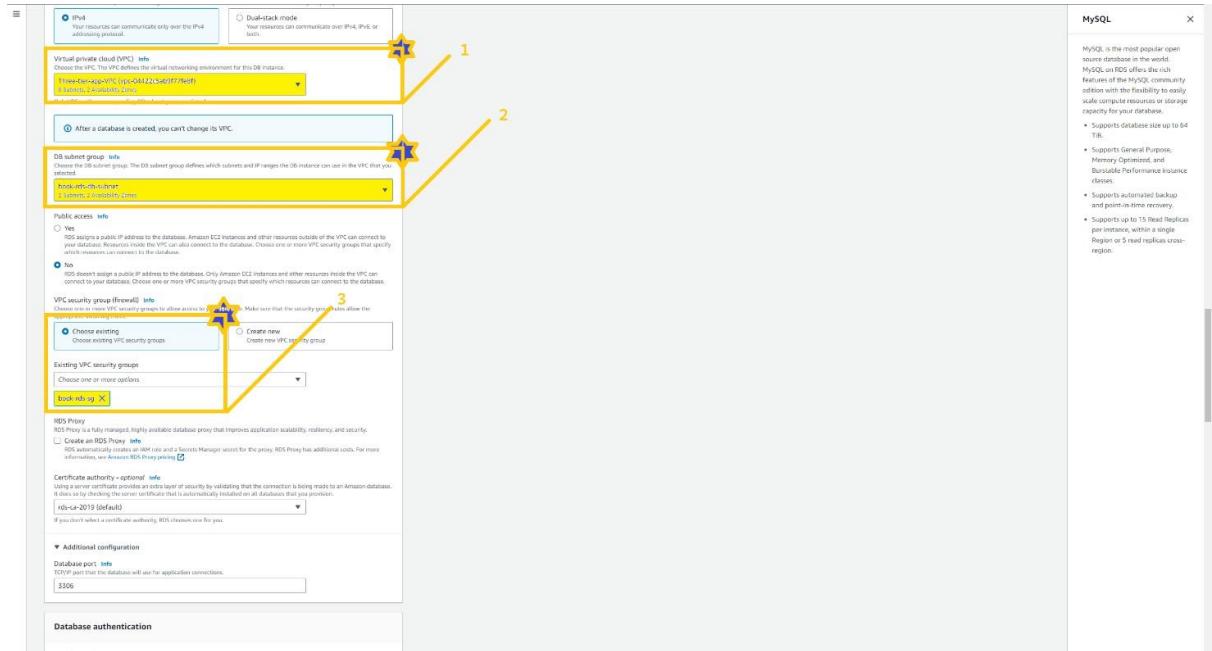
Network type: Info

In VPC Auto-Route mode, make sure that you associate an IPv4/IPv6 block with a subnet in the VPC you specify.

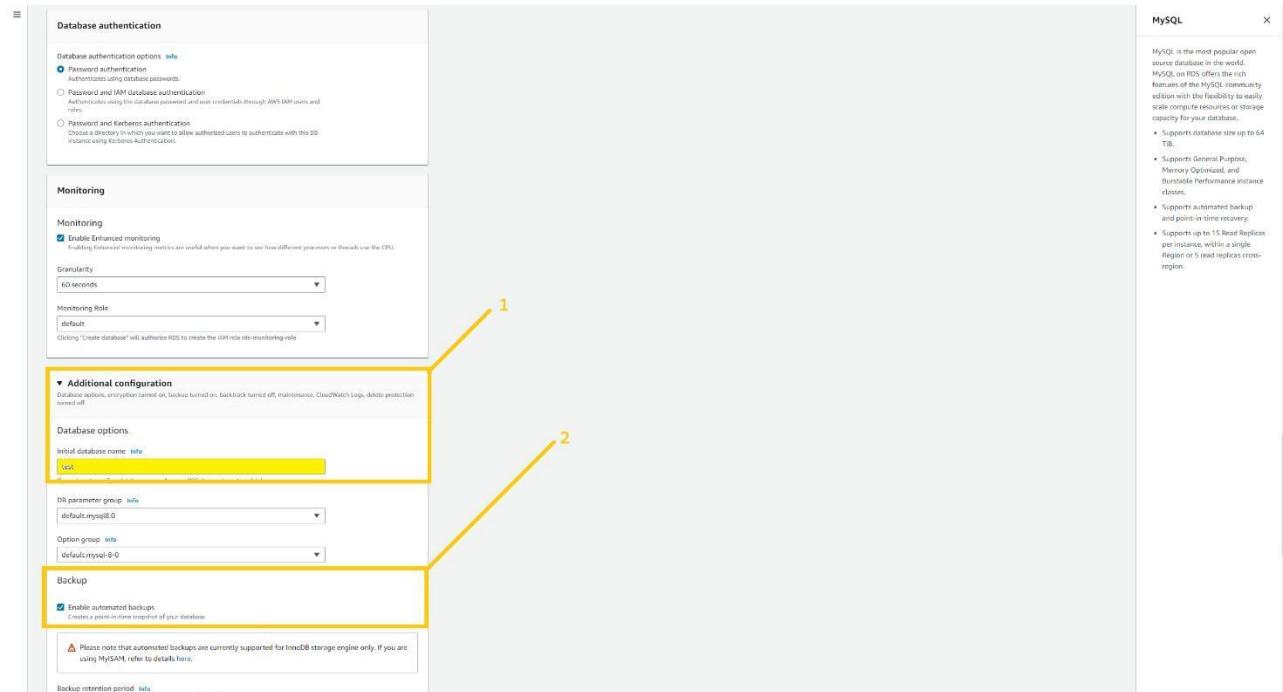
IPv4
All resources can communicate over the IPv4 network.

IPv6
All resources can communicate over IPv4, IPv6, or both networks.

In VPC, select VPC that we created earlier and in DB subnet group select the group that we just created, In the public access option please select No, choose existing security, and select security group **book-rds-db**.



Scroll down, click on Additional Configuration, and in the database option give the name **test** because we need a database with the name of the **test** in the application. Enable Automated Backup. Note: you have to enable automated backup otherwise you won't be able to create a read replica of the RDS instance.



Scroll down, mark on enable encryption checkbox to make the database bit more secure, and click on Create database button below.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB.
- Supports General Purpose, Memory Optimized, and Burstable Performance Instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB.
- Supports General Purpose, Memory Optimized, and Burstable Performance Instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB.
- Supports General Purpose, Memory Optimized, and Burstable Performance Instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB.
- Supports General Purpose, Memory Optimized, and Burstable Performance Instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

Create database

Note: RDStake 15-20 minute because it creates a database and then take a snapshot. So please have patience and wait for it to be ready.

After your database is completely ready and you see the status Available then select the database and click on the Action button. There you can see the drop-down list. Please click on created read-replica.

This screenshot shows the AWS RDS Databases page. A database named 'book_rds_db' is listed. A context menu is open over the database row, with three numbered arrows pointing to specific options:

1. Create read replica
2. Create Multi-AZ deployment - new
3. Create Blue/Green Deployment - new

This page is similar to creating a database. In the AWS region select the region where you want to create the read replica. In my case, It is **Oregon (us-west-2)**. Give a name to your read replica, and select all the necessary configurations that we did before while creating the database. For your reference, I have shown everything in the below images.

This screenshot shows the 'Create read replica' wizard step 1: Settings. It shows the 'Replica source' as 'book_rds_db' and the 'DB instance identifier' as 'book_rds_db'. A context menu is open over the 'DB instance identifier' field, with four numbered arrows pointing to:

1. 'db.t3.micro' instance class dropdown
2. 'Create read replica' button
3. 'Burstable classes (includes t classes)' checkbox
4. 'AWS Region' dropdown set to 'Oregon (us-west-2)'

This screenshot shows the 'Create read replica' wizard step 2: Storage. It shows the 'Storage type' as 'General Purpose SSD (gp2)', 'Allocated storage' as '22 GB', and 'Provisioning less than 100 GiB of General Purpose (SSD) storage for high-throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credits balance.' A context menu is open over the 'Allocated storage' field, with four numbered arrows pointing to:

1. 'Enable storage autScaling' checkbox
2. 'Multi-AZ DB instance' checkbox
3. 'Dual stack mode' checkbox
4. 'Existing VPC security group' dropdown set to 'book_rds_sg'

Amazon RDS

Database authentication

Additional configuration

Backup

Encryption

Monitoring

Log exports

IAM role

Maintenance

Deletion protection

Create read replica

Amazon RDS

Database authentication

Additional configuration

Backup

Encryption

Monitoring

Log exports

IAM role

Maintenance

Deletion protection

Create read replica

Once you click on the button create replica . It will start creating that.

DB identifier	Status	Role	Engine	Region & AZ	Size	Actions	CPU	Current activity	Maintainance	VPC	Multi-AZ
book-ids-db	Creating	Primary	MySQL, Community	us-east-1b	db.t3.micro	1 Action	1..... 3.22% 0 Connections	none	vpc-04422c5ab9f77fe0f	Yes

You can check your read replica on the specified region's RDS dashboard. So let me head over to **Oregon** and show you the read replica.

The screenshot shows the Amazon RDS console with the 'Oregon' region selected. The 'Databases' section displays a single database named 'book-rds-db'. The 'Actions' column for this database has a yellow box around it. At the top right of the page, there is a 'Create detail' button and a 'Restore from S3' button. A yellow arrow points to the 'Origin' dropdown menu.

Note: we can't write anything into a read replica. It is just read-only database. So, when a disaster happens, we just have to promote read replica so that it becomes the primary database in that region.

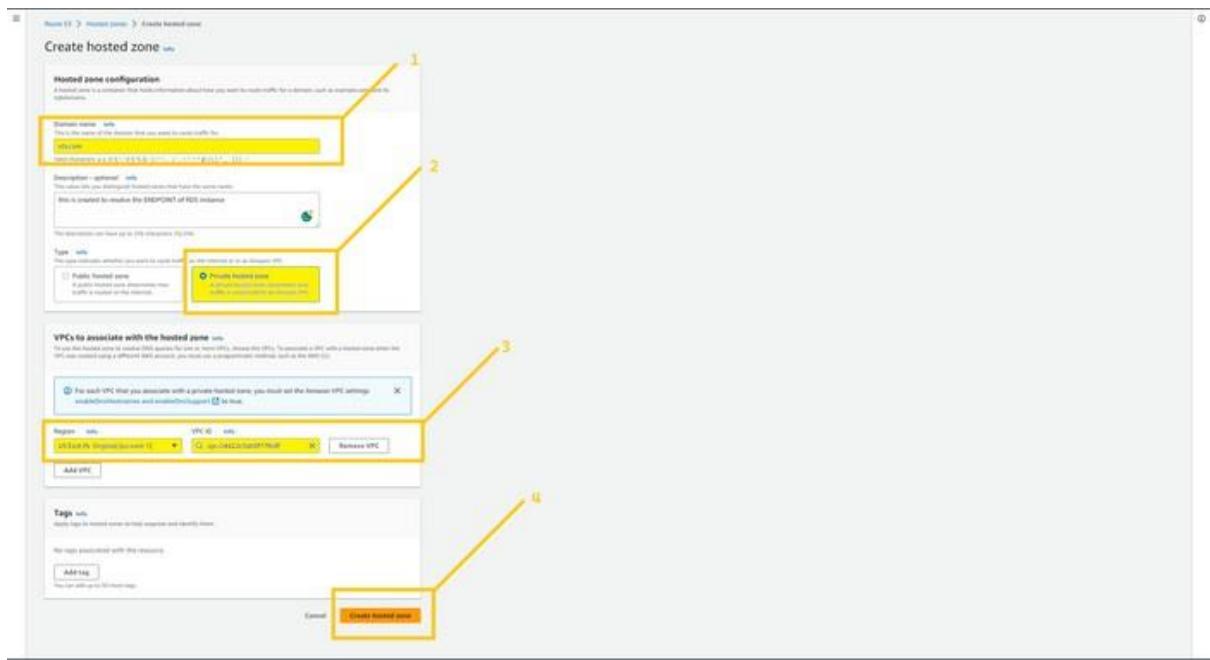
Now we are going to utilize route 53 service and create two private hosted zone. One for **north Virginia(us-east-1)** and another one for **Oregon region (us-west-2)** with the same name. you may think Why Two hosted zone with the same name? don't worry I will answer it later. So head over to Route 53. Type route 53 in the AWS console. And click on the service.

The screenshot shows the AWS Services page with 'Route 53' selected. The 'Services' section lists various AWS services, with 'Route 53' highlighted by a yellow box. To the right, there are sections for 'Welcome to AWS', 'AWS Health', and 'AWS Lambda'.

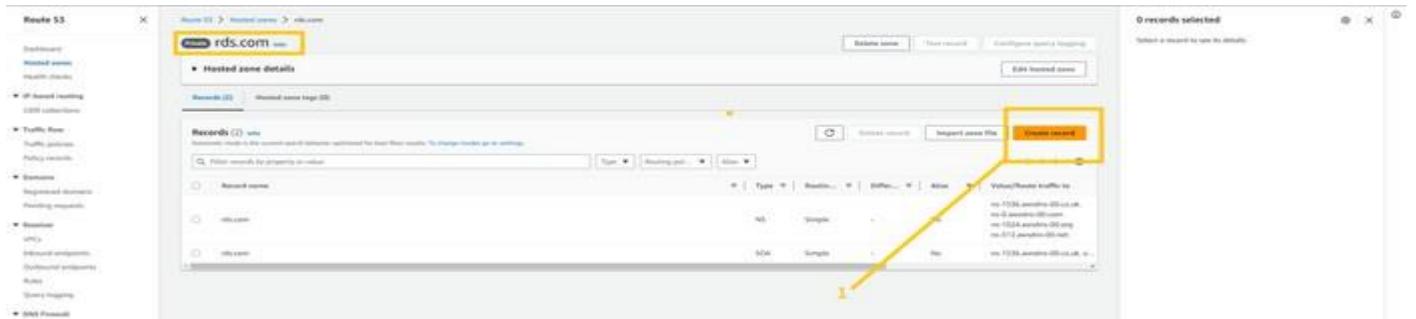
Firstly, we are going to create a hosted zone for **us-east-1**. Click on the Hosted Zones button on the left panel and click on the created hosted zone button on the top right corner.

The screenshot shows the 'Route 53 > Hosted zones' page. The left sidebar shows 'Hosted zones' selected. The main area displays a table of 'Hosted zones (4)'. At the top right of the table, there is a 'Create hosted zone' button highlighted by a yellow box.

Give any domain name because anyhow it will be private hosted zone but it would be great if you give the name same as mine (**rds.com**). Please select the private hosted zone and Select the region. In my case, it is **us-east-1**. And then select VPC ID. Make sure you select VPC that we created earlier. Because this hosted zone will resolve the record only in specified VPC. and then click on the Create hosted zone.



Now we are going to create a Record that points to our RDS instance which is in **us-east-1**. So click on create record button on the top right corner.

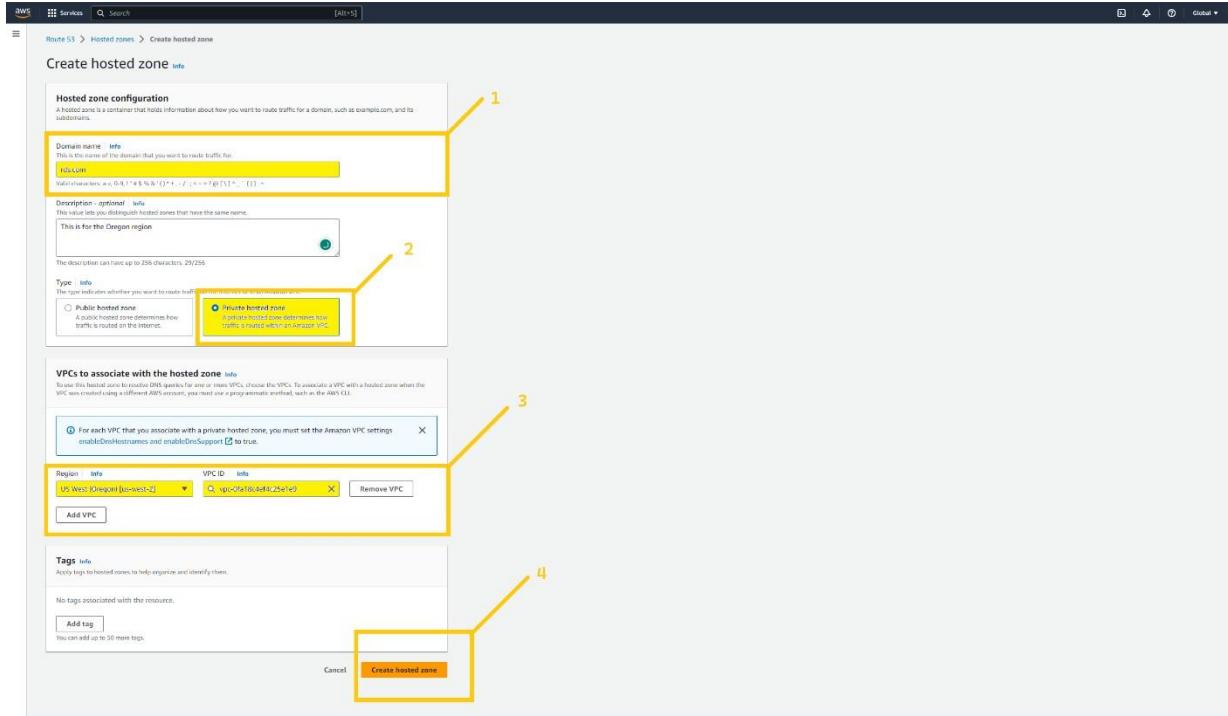


Select simple routing,

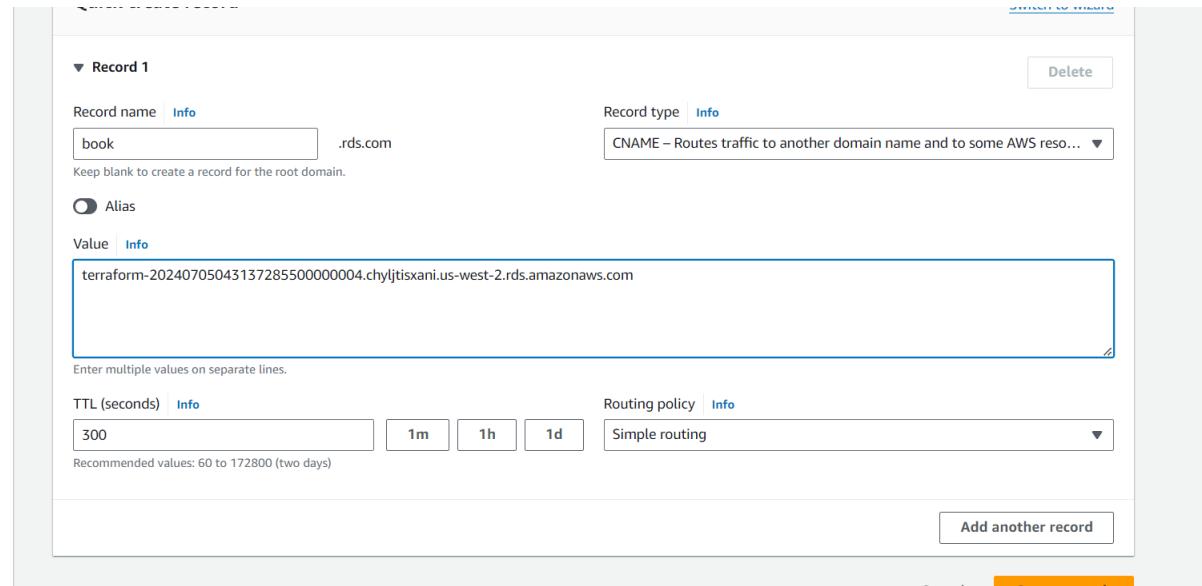
Here type book in the record name field. In the record type select CNAME. In the value field paste **endpoint of the RDS which is in us-east-1**. Then click on the defined record button.

The screenshot shows the 'Quick Create Record' dialog box. Record name: book.rds.com, Record type: CNAME, Value: terraform-20240705043137285500000004.chyljtxani.us-east-1.rds.amazonaws.com, TTL: 300, Routing policy: Simple routing. The 'Create records' button is highlighted.

Now we are going to create a new hosted zone with the same name, but for **disaster recovery region** and that is **us-west-2 (Oregon)**. While creating hosted zone please keep in mind that you need to choose the **us-west-2** region and select VPC that you have created in the **the us-west-2** region. Again you can utilize the below image for reference



Our next step is to set up a **simple record** that points to the **read replica** (database) which is in the **us-west-2 (Oregon)**. So select the hosted zone that was created for **us-west-2** and defined a simple record in that. Everything is the same as we defined the record in the **us-east-1** hosted zone.



After successfully completing the above steps your Route 53 console look like this.

The screenshot shows the AWS Route 53 'Create Record' page. A new CNAME record named 'book.rds.com' is being created. The 'Record type' is set to 'CNAME – Routes traffic to another domain name and to some AWS resources'. The 'Value' field contains the URL 'terraform-2024070504313728550000004.chyjxitxani.us-west-2.rds.amazonaws.com'. The 'TTL (seconds)' is set to 300. The 'Routing policy' is 'Simple routing'. An 'Add another record' button is visible at the bottom right.

You may think that We can connect two regions and VPC in one hosted zone then why two private hosted zone with the same name? And the answer is Endpoint of both databases will be different and we can't implement a health check coz we attached a security group that allow traffic from **3306 port from only backend SG**. So route 53 can't check the health of databases and because of that we can't implement a Failover record here but we will do that with the application server (backend-server).And here we successfully completed our RDS setup. Let's go ahead and explore more services.

◆ Certificate Manager

As you saw in previous screenshots, I have the domain name ankitjodhani.club in Route 53. Now I am going to use this domain name to create subdomains such as api.ankitjodhani.club and that will resolve **ALB-backend DNS**. Furthermore, we need an SSL certificate so that we can make the connection secure.

Note: we are going to create certificates in both regions us-east-1 and us-west-2.

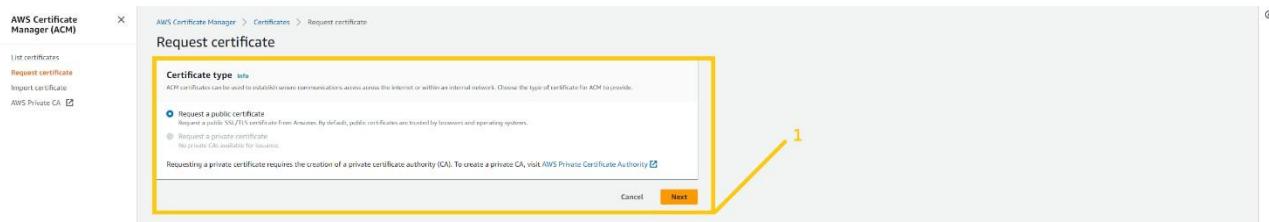
So, let's head over to ACM (AWS certificate manager). Type certificate manager in the AWS console search bar. And click on the service.

The screenshot shows the AWS search interface with 'certificate manager' typed into the search bar. The 'Services' section lists several options, with 'Certificate Manager' highlighted by a yellow box. Other listed services include AWS Health, AWS Private Certificate Authority, and AWS Marketplace.

Now click on the list certificates button on the left panel and then click on the request certificate on the top right corner.

The screenshot shows the AWS Certificate Manager (ACM) console. The left sidebar has 'List certificates' highlighted. The main area shows a table titled 'Certificates (0)'. At the top right of the main area, there is a large orange 'Request' button, which is also highlighted by a yellow box. Another yellow box labeled '1' points to the 'List certificates' button in the sidebar.

Select the option Request the public certificate and click on the next button.



In the domain name field please type ***.Your_Domain_Name.xyz** in my case it is ***ramakrishna.shop** DON'T DO ANY TYPO. In the validation method select DNS validation and click on the request certificate.

Domain names

Provide one or more domain names for your certificate.

Fully qualified domain name [Info](#)

`*.vardhan.live`

Add another name to this certificate

You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name.

Validation method [Info](#)

Select a method for validating domain ownership.

DNS validation - recommended
Choose this option if you are authorized to modify the DNS configuration for the domains in your certificate request.

Email validation
Choose this option if you do not have permission or cannot obtain permission to modify the DNS configuration for the domains in your certificate request.

Key algorithm [Info](#)

Here you can see the status pending validation. Now we need to add a **CNAME record** in our domain. If you are not using route 53 then you need to add this CNAME record manually by going to your DOMAIN REREGISTER. And if you are using route 53 then click on the button create record in route 53 and click on the create record button. That's it

Certificate with ID `a10deead-1c89-4c47-8f0a-95cc5374a3cf` status: Pending validation has been created. Further action is needed to complete the validation and approval of the certificate.

Certificates > a10deead-1c89-4c47-8f0a-95cc5374a3cf

`89-4c47-8f0a-95cc5374a3cf`

Status: Pending validation [Info](#)

Create records in Route 53

Status	Renewal status	Type	CNAME name	CNAME value
Pending validation	-	CNAME	<code>_4ed5d1155dc1b6077e0f5b4e45604f57b.ankitjpdhani.club</code>	<code>74771451f17efb0e545fb4066137088.gqבקגnp.acr-valiations.won</code>
Serial number	N/A	Requested at	June 11, 2023, 02:48:53 (UTC+05:30)	
Public key info	RSA 2048	Issued at	N/A	
Signature algorithm	SHA 256 with RSA	Not before	N/A	
Can be used with	CloudFront, Elastic Load Balancing, API Gateway and other integrated services.	Not after	N/A	

Value: No tags. No tags associated with this resource.

And in just a few minutes you will see the status issued.

The screenshot shows the AWS Certificate Manager (ACM) interface. A success message at the top states: "Successfully requested certificate with ID a1ddeea4-1c89-4c47-8f0a-95cc5374a3cf" and "Successfully created DNS records for certificate with ID a1ddeea4-1c89-4c47-8f0a-95cc5374a3cf". Below this, a certificate card for "a1ddeea4-1c89-4c47-8f0a-95cc5374a3cf" is displayed, showing it was issued by "amazonaws.com/us-west-2:138250738702" and is "Amazon issued". The status is "Issued" (highlighted with a yellow box). The "Domains" section lists "(1)". Buttons for "Create records in Route 53" and "Export to CSV" are visible at the bottom.

Note: I created a certificate in N.virginia(us-east-1) but you need to do the same thing for the Oregon region(us-west-2).

◆ Application Load balancer (ALB) and Route 53

Now it's time to set up an application load balancer. We need two load balancers, one point to the backend server, and another point to the frontend server.

Note: I am doing setup in N.virginia (us-east-1) but you have to do the same setup for Oregon (us-west-2) or whatever region you have chosen.

Type ec2 in the AWS console, and click on the EC2 service.

The screenshot shows the AWS EC2 service page. The search bar at the top has "EC2" typed into it. The main content area shows "Search results for 'EC2'" with a note to "Try searching with longer queries for more relevant results". Under the "Services" section, "EC2" is highlighted with a yellow box. Other services listed include "EC2 Image Builder" and "AWS Lambda". To the right, there's a "Welcome to AWS" sidebar with sections for "AWS Health info", "Getting started with AWS", "Training and certification", and "Scheduled changes".

Note: before we created ALB we need to create a Target group(TG). So first we will create TG for ALB-frontend and then create TG for ALB-backend.

Click the target group button on the bottom of the left panel. And click on the create target group button in the middle of the page.

The screenshot shows the "Target groups" page under the EC2 service. The left sidebar has a "Load Balancing" section with "Target Groups" highlighted with a yellow box. The main content area shows a table for "Target groups" with columns for Name, ARN, Port, Protocol, Target type, Load balancer, and VPC ID. A button labeled "Create target group" is highlighted with a yellow box. A callout arrow points from the "Target Groups" link in the sidebar to the "Create target group" button. Another callout arrow points from the "Create target group" button to the "Create target group" button itself.

Here we can configure our TG. Select the instance in the target type. You can give any name to TG but try to give some relevant name such as **ALB-frontend-TG** because we are creating TG for ALB-frontend. In the VPC section select VPC that we created earlier.

Step 2:
Register targets

Basic configuration
Settings in this section can't be changed after the target group is created.

Choose a target type

- Instances
 - Targets individual AWS Lambda functions or Amazon EC2 instances in your VPC.
 - Facilitates the use of Amazon CloudWatch Metrics Metrics to manage and scale your Lambda functions.
- IP address
 - Facilitates routing to multiple IP addresses and network interfaces on the same instance.
 - Offers flexibility with microservice-based architectures, simplifying inter-application communication.
 - Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 failover.
- Lambda function
 - Facilitates routing to a single Lambda function.
 - Accessible to Application Load Balancers only.
- Application Load Balancer
 - Offers the flexibility for a Lambda Load Balancer to accept and route TCP requests within a specific VPC.
 - Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name
ALB-frontend-TG
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol Port
HTTP | 80 | 44355

VPC
Select the VPC with the instances that you want to include in the target group.
Three-tier-ppg-VPC
vpc-04422c5ab77f70ff
Subnet: 172.0.0.0/16

Protocol version

- HTTP1
 - Sends requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or gRPC.
- HTTP2
 - Sends requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
- gRPC
 - Sends requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol
HTTP

Keep everything as it is, scroll down, and click on the Next button.

Health check protocol
HTTP

Health check path
Use the default path of "/" or specify a custom path if preferred.
/

Advanced health check settings

Health check port
The port the load balancer uses when performing health checks on targets. By default, the health check port is the same as the target group's traffic port. However, you can specify a different port as an override.

- Traffic port
- Override

Healthy threshold
The number of consecutive health checks successes required before considering an unhealthy target healthy.
2

Unhealthy threshold
The number of consecutive health check failures required before considering a target unhealthy.
2

Timeout
The amount of time, in seconds, during which no response means a failed health check.
5 seconds

Interval
The approximate amount of time between health checks of an individual target.
30 seconds

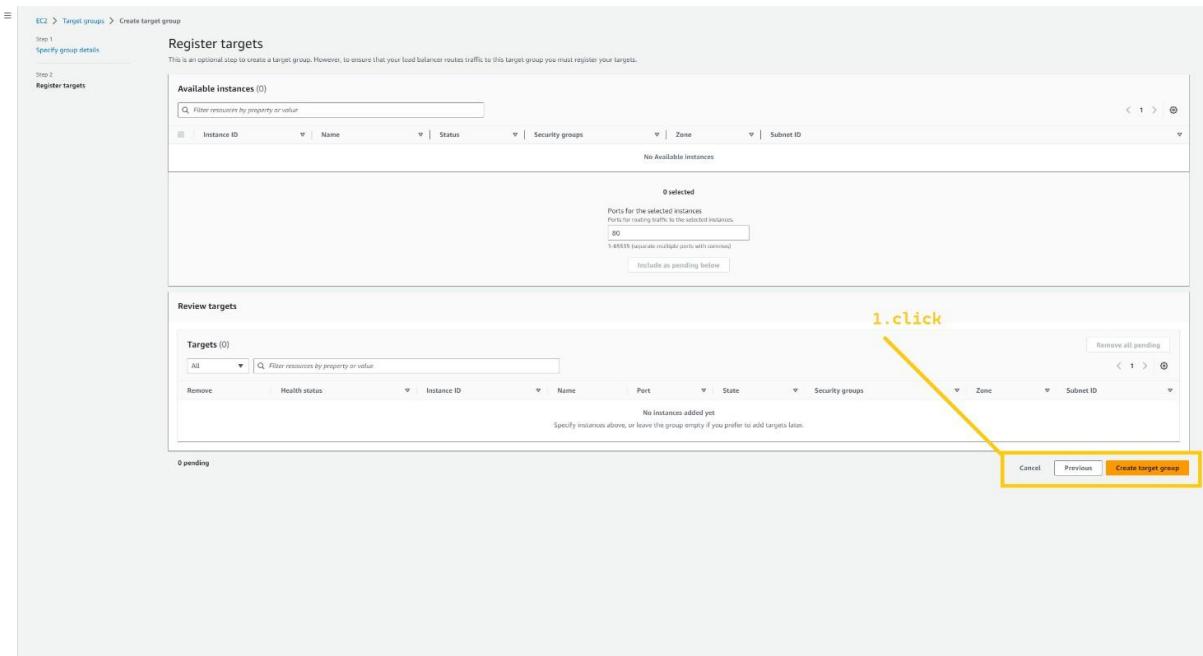
Success codes
The HTTP codes to use when checking for a successful response from a target. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-209").
200

Attributes

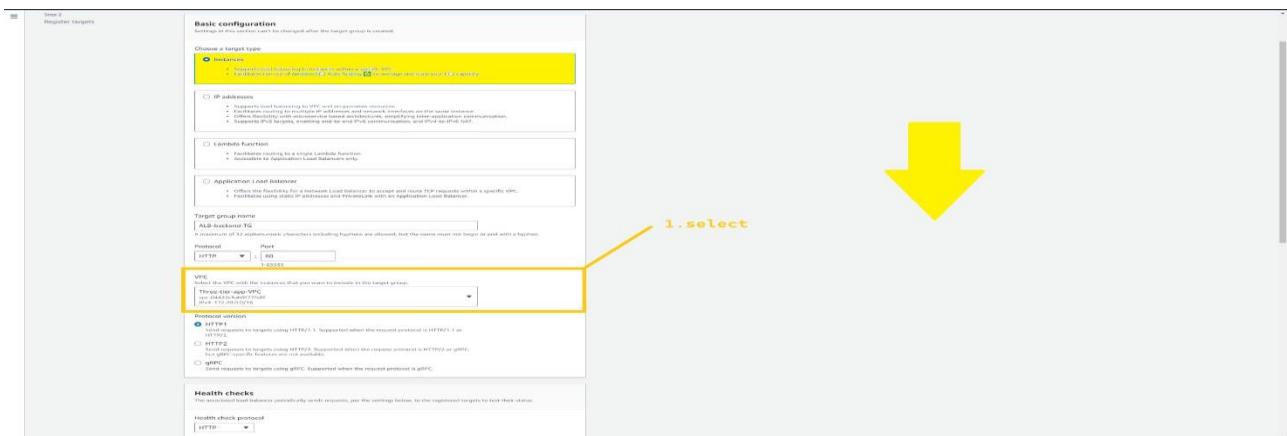
Tags - optional
Console allows tags to your target group. Tags enable you to categorize your AWS resources so you can more easily manage them.

Next

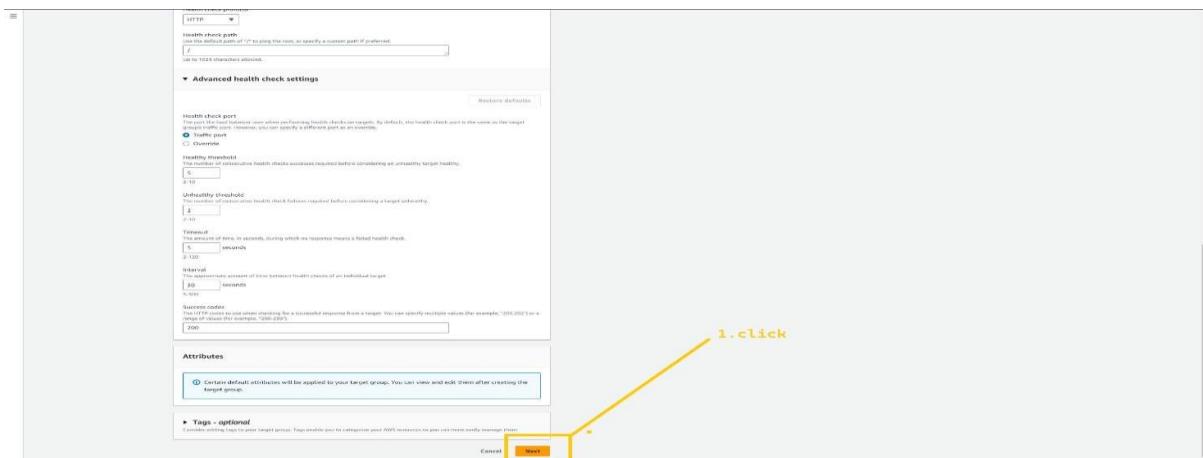
Click on the create target group button.



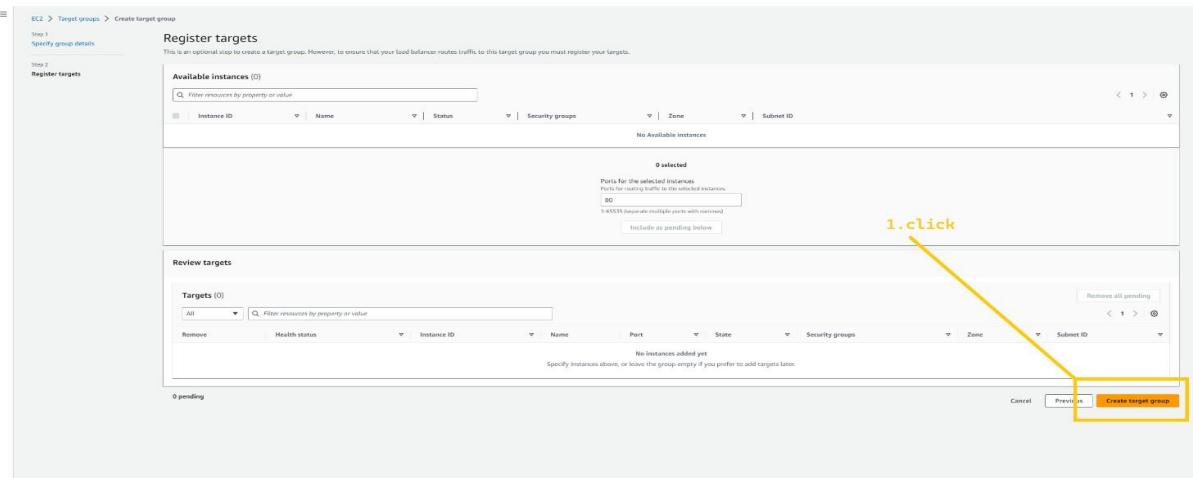
Let's create TG for **ALB-backend**. Click on the create target group button. Select the target type Instance. Again give some meaning full name such as **ALB-backend-TG**. Select VPC that we have created.



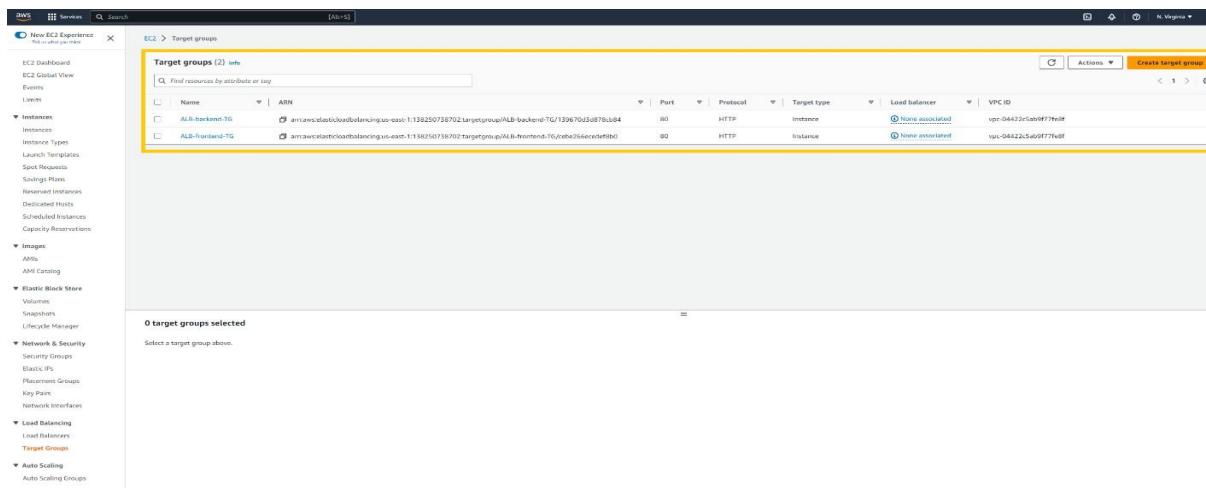
Scroll down and click on the next button.



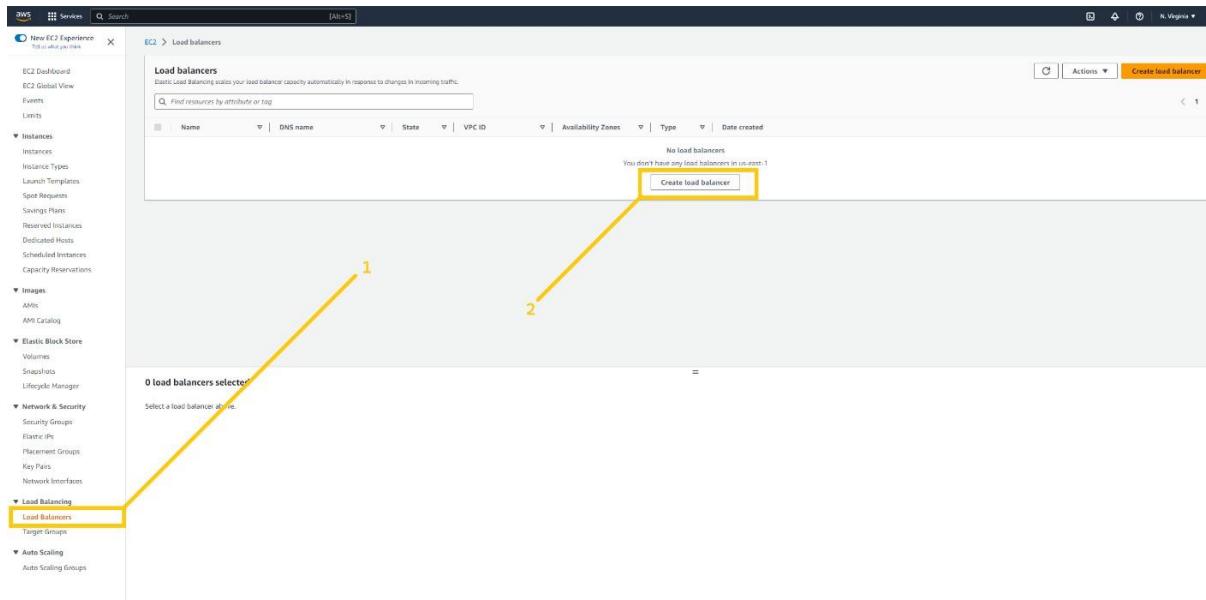
And click on the creatd target group. That's it.



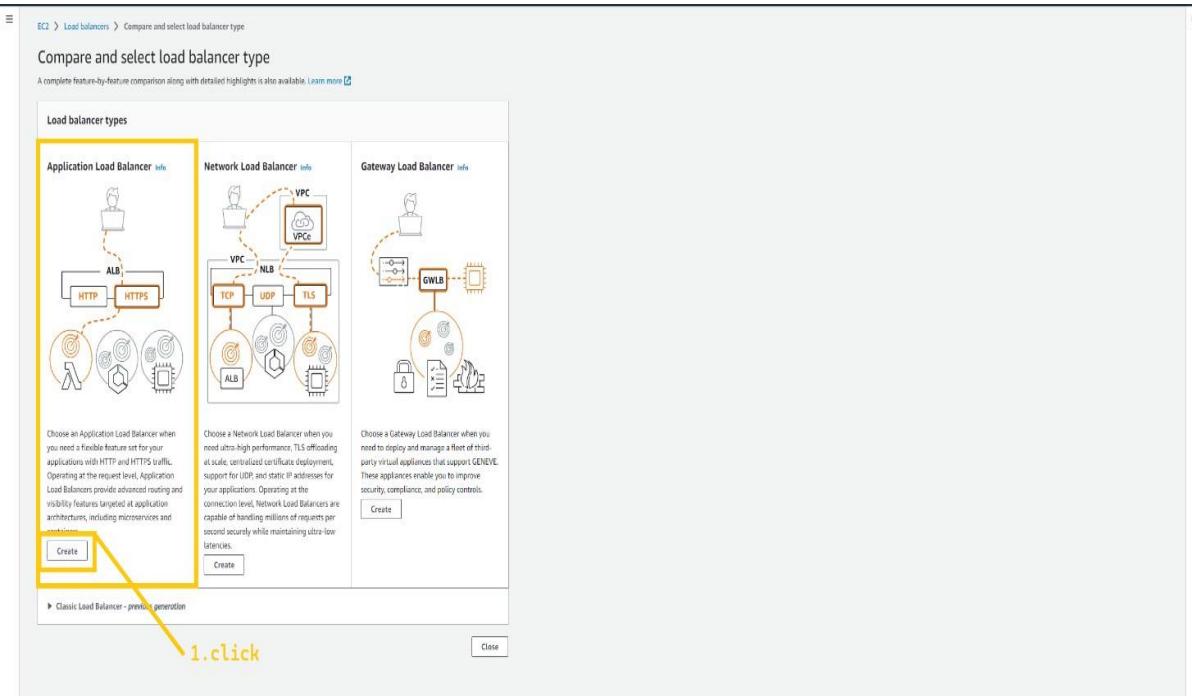
So we have two TG. **ALB-frontend-TG** and **ALB-backend-TG**.



Now let's associate these TG with the load balancer. So click on the Load Balancer button at the bottom of the left panel and click on the create load balancer button. First, we will create ALB for frontend.

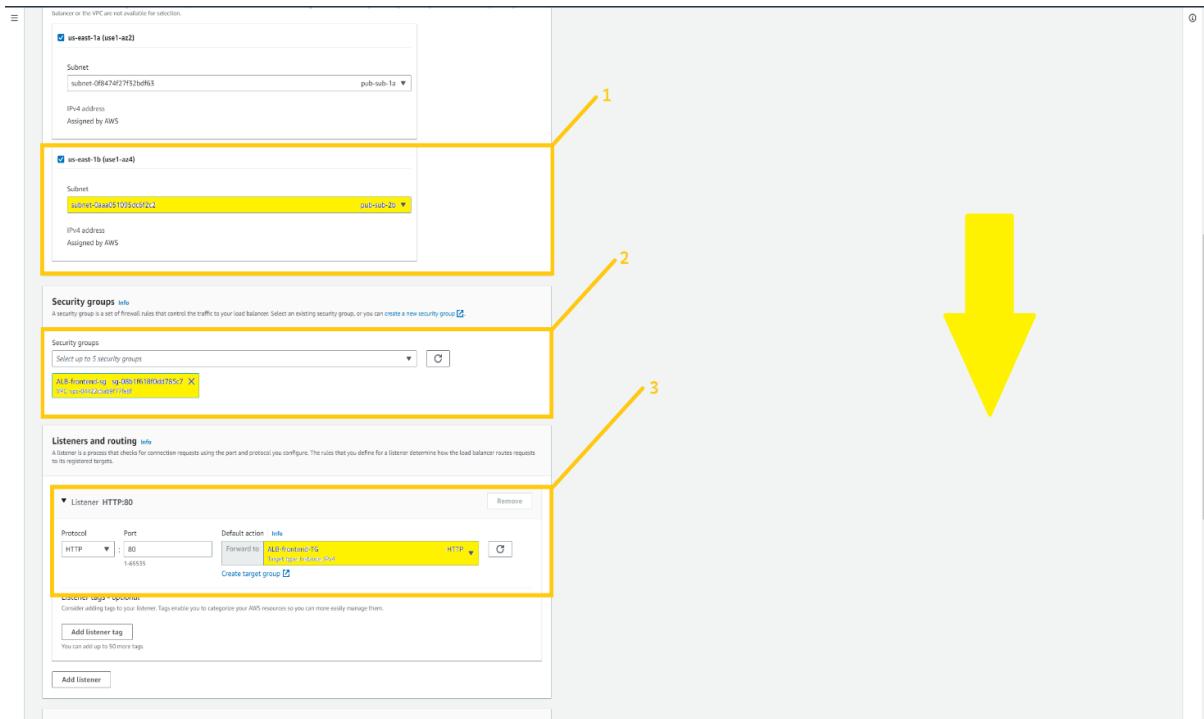


Choose Application load balancer and click on create button.

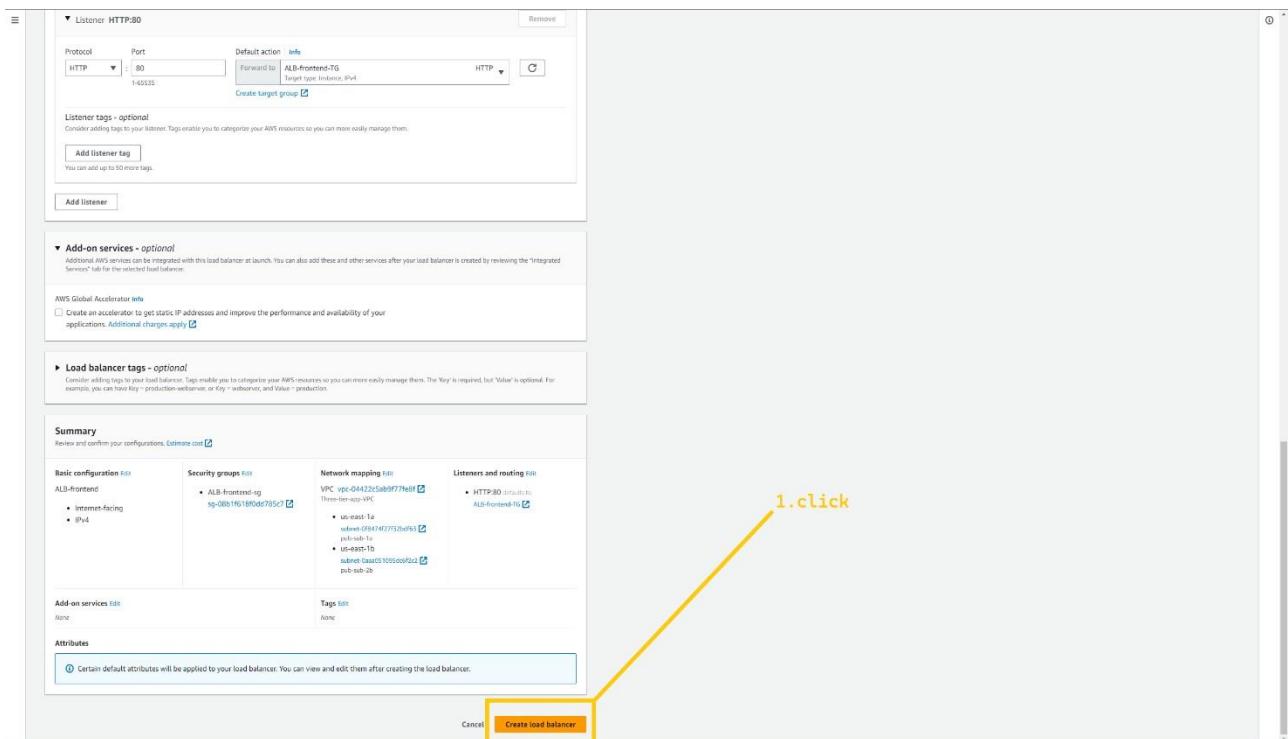


here we can configure our ALB. First, give the relevant name to ALB such as **ALB-frontend**. Select the internet-facing option. In Network mapping select VPC that we have created. Select both availability zone **us-east-1a** and **us-east-1b**. and select subnet **pub-sub-1a** and **pub-sub-1b** respectively.

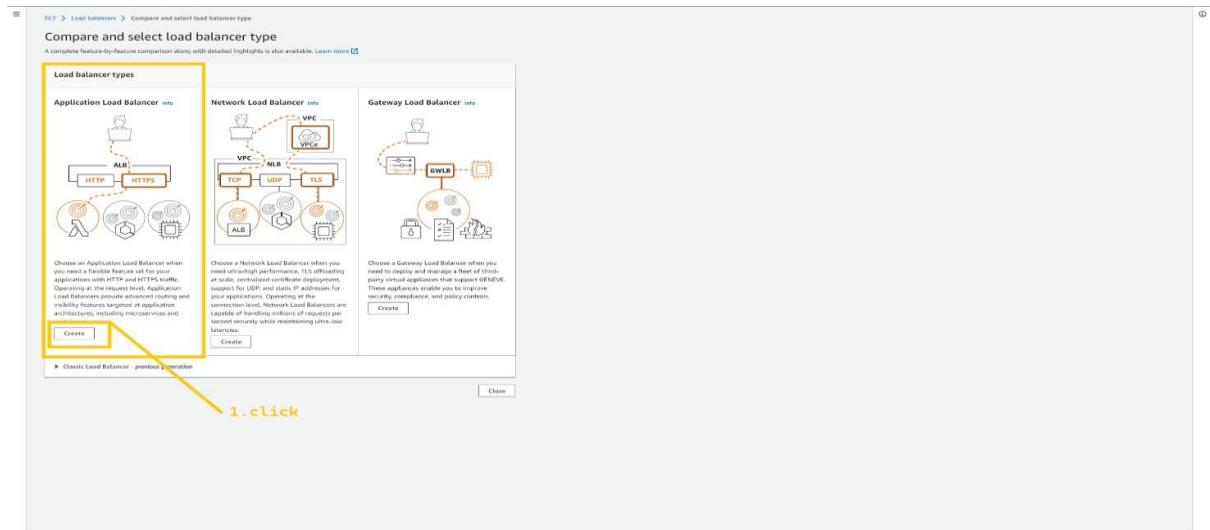
Select security group **ALB-frontend-sg**. This SG we have created for ALB-frontend. In the listener part select TG that we have just created **ALB-frontend-TG**.



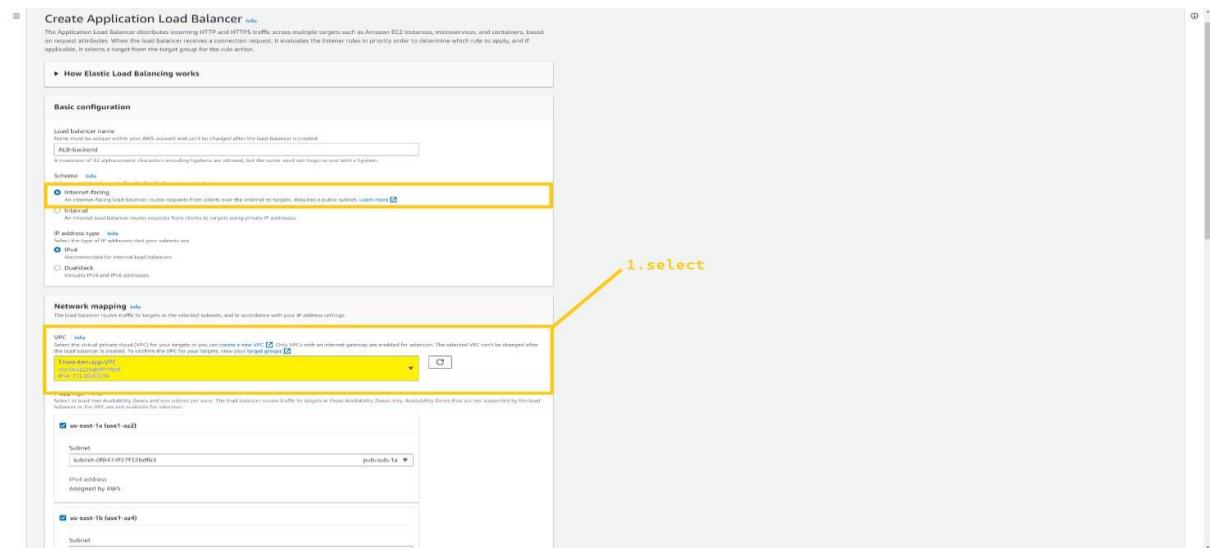
Scroll down and click on the create load balancer button.



Now, lets create ALB for backend. Again choose Application load balancer option and click on the create button.



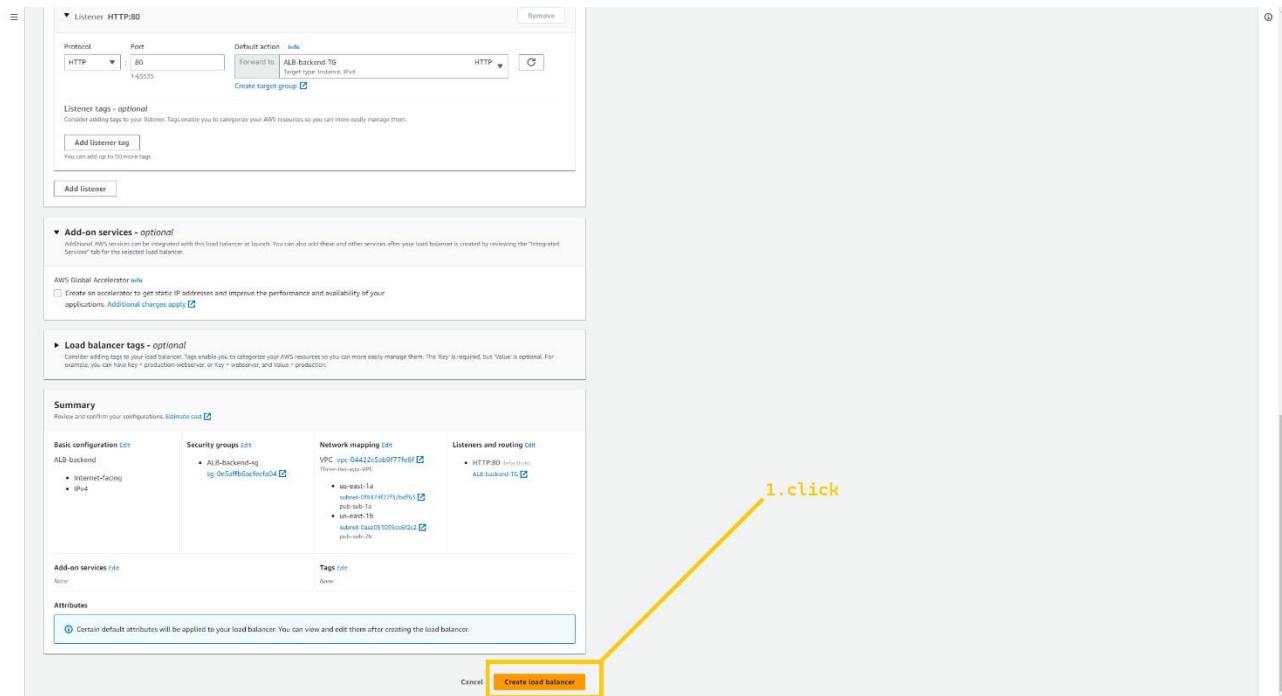
Select Internet facing option. And select VPC that we have created.



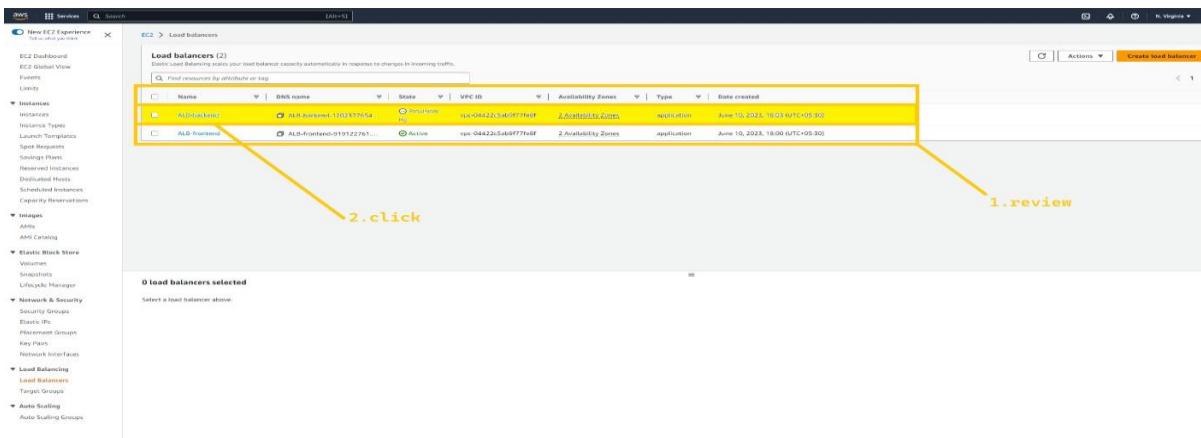
Select both availability zone **us-east-1a** and **us-east-1b**. and select subnet **pub-sub- 1a** and **pub-sub2b**. select security group **ALB-backend-sg** that we created for ALB- backend. And in the listner part select TG that we just created **ALB-backend-TG**.



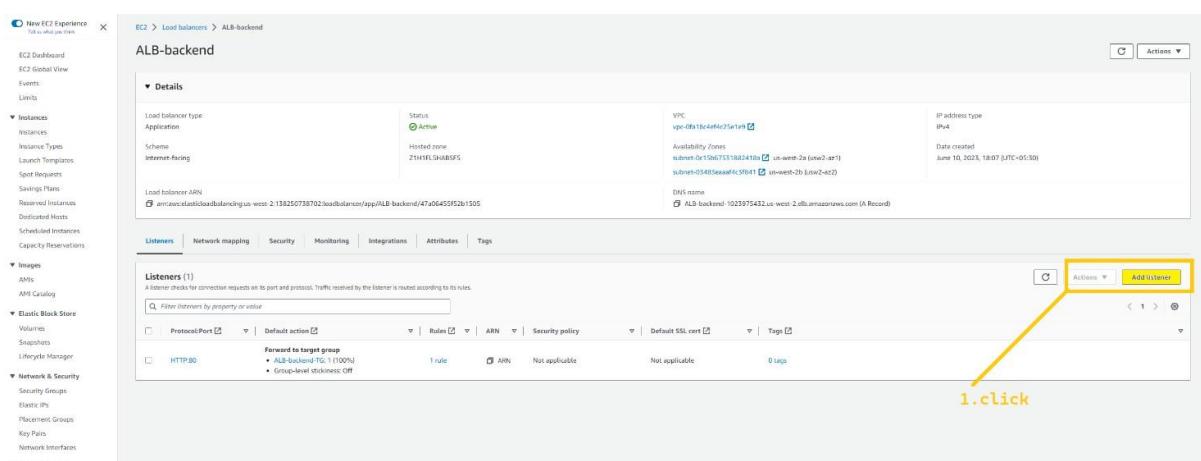
Scroll down as click on the Created Load balancer button



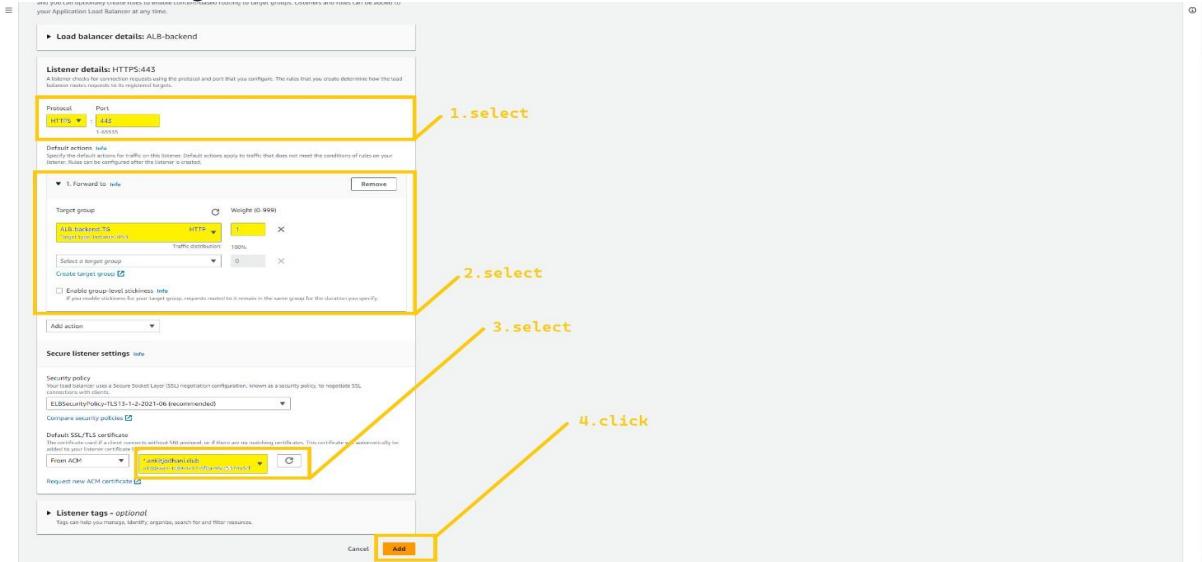
Now we have two load balancers, **ALB-frontend** and **ALB-backend**. But we need to add one more listener in **ALB-backend**. So click on **ALB-backend**.



Click on add listener the button that is located on the right side.



Here In listener details select HTTPS. Default Action should be Forward and select ALB-backend-TG. Now we need to select the certificate that we have created. So in the Secure Listener setting select the certificate. And click on the add button below.



So here we successfully completed the ALB setup for the **N.virginia region (us-east- 1)**, and your task is to set up the same ALB for the **Oregon region (us-west-2)**.



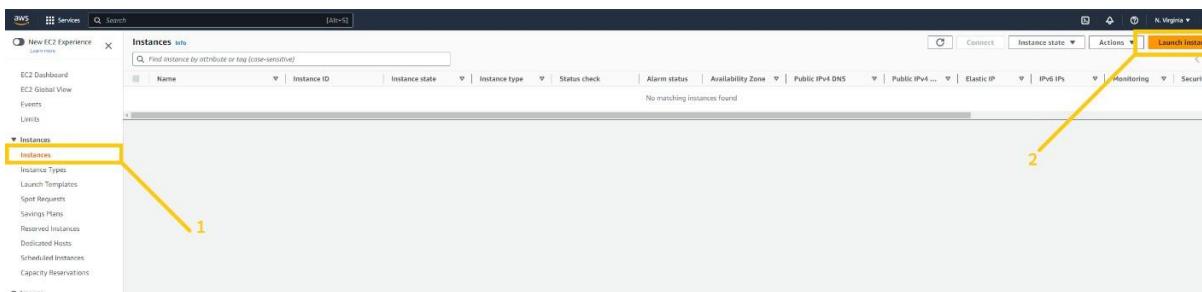
Same process do for frontend loadbalancer in both regions to add another listener rule that is https for security.

◆ EC2

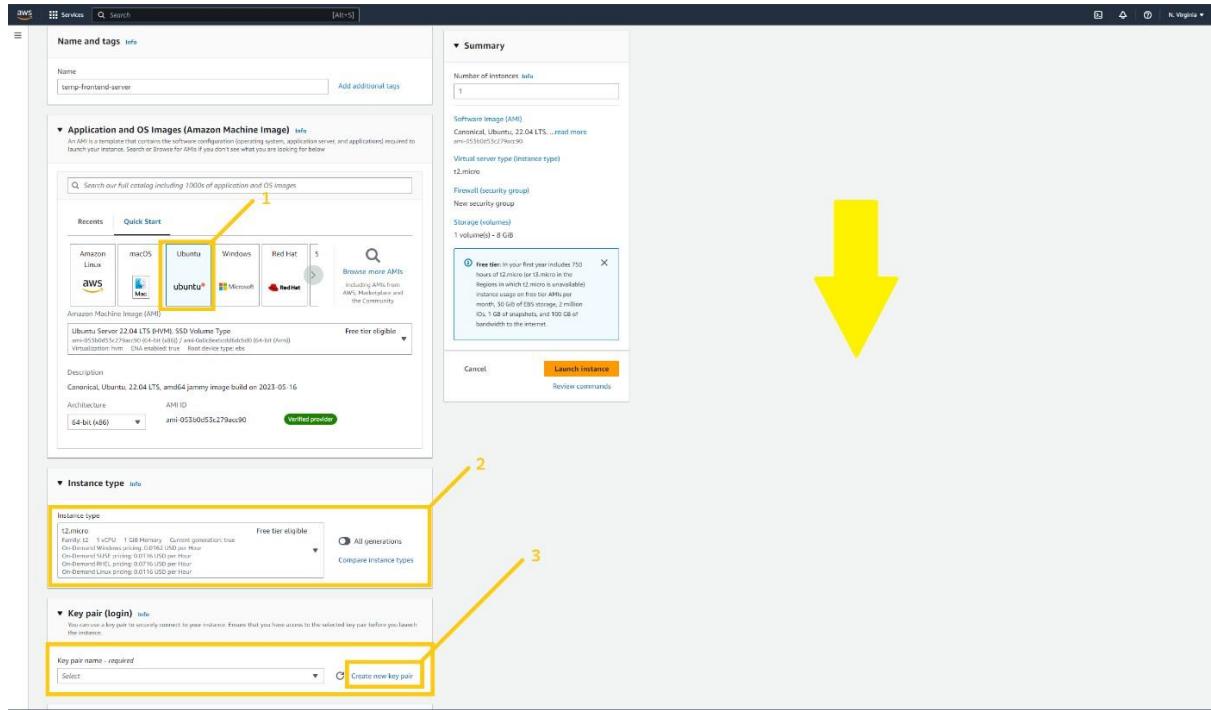
Now we are going to create a temporary frontend and backend server to do all the required setup, take snapshots and create Machine images from it. So that we can utilize it in the launch template. It is a long process so bear with me.

Note: we are doing this setup in the us-east-1 region and we don't have to do this in the us-west-2 because we are going to leverage AWS backup service and copy it in the us-west-2 region.

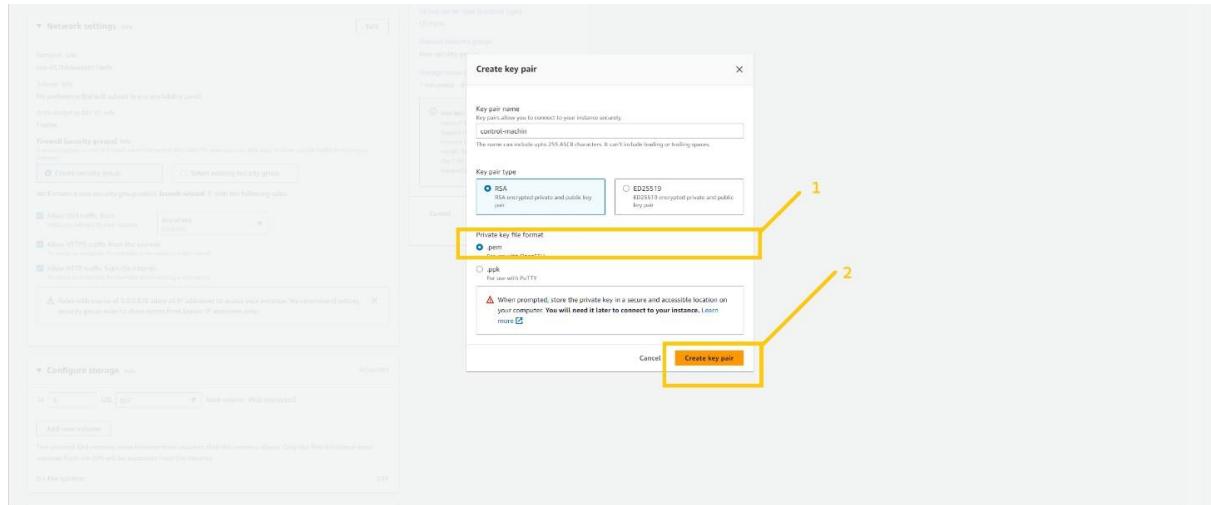
First, click on the instance button and then click on the Launch Instance button on the top right corner.



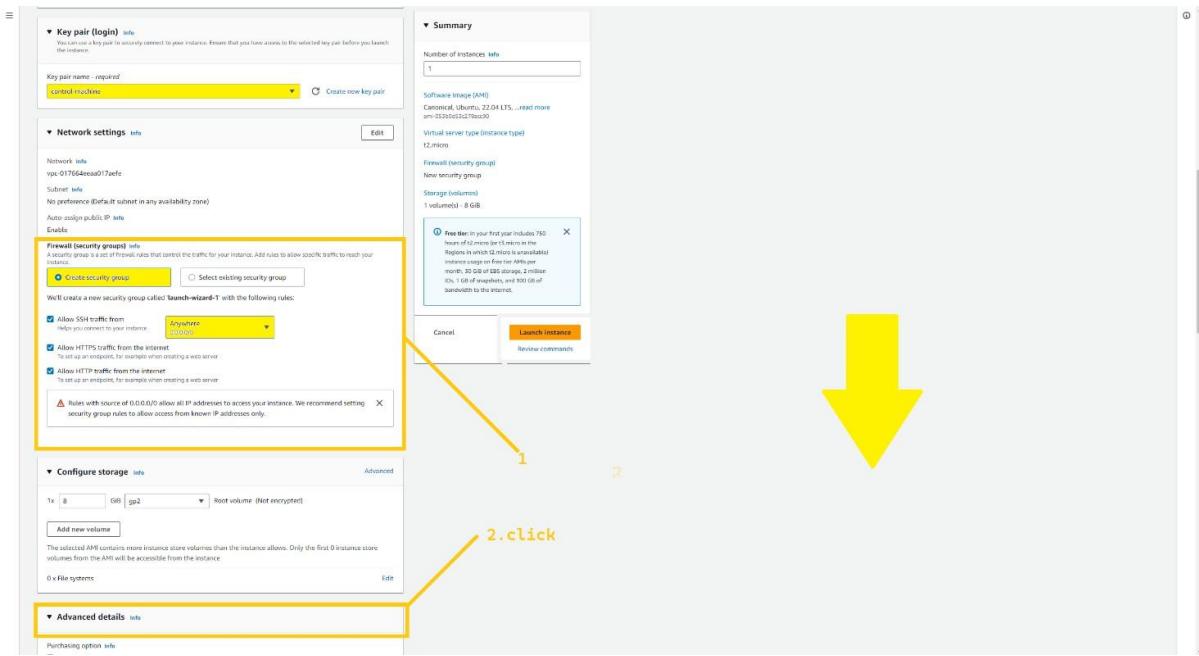
First, we are going to set up a frontend server. Give a name to your instance (**temp-frontend-server**). Select Ubuntu as the operating system. Choose the instance type as t2.micro. click on Create key pair if you don't have it.



If you are creating key pair make sure you select .PEM file format as I have shown in the below image. Because we are going to use Git bash to do the login NOT putty and give any name to your key. And save it somewhere safe location on your computer.



Here we are doing a temporary setup so we don't use our OWN VPC. we can use the default VPC given by AWS. In short, keep the Network setting as it is. In the firewall setting select all the fields as I shown in the below image to keep things simple. And lastly, click on the Advance details option.



Scroll down to the bottom of the page, here we can see one text box with the name USER DATA. Here in this text box, you can write your bash script file and that will be executed during the launch of the instance. I have given the bash script below. so please copy that script and paste it here. And lastly, click on the launch instance button.

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

```
sudo apt update -y
```

```
sudo apt install apache2 -y
```

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash - && \sudo apt-get install -y nodejs -y
```

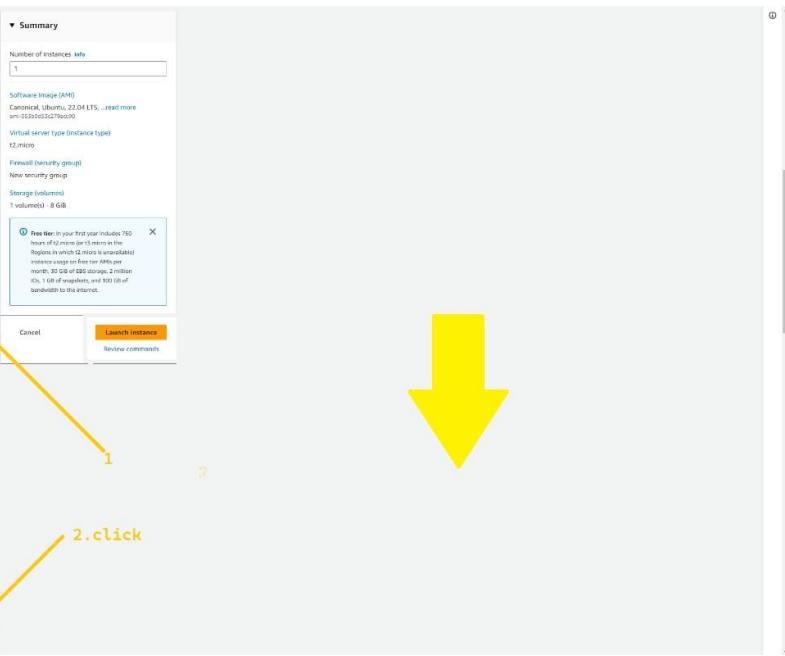
```
sudo apt update -y
```

```
sudo npm install -g corepack -y
```

```
corepack enable
```

```
corepack prepare yarn@stable --activate
```

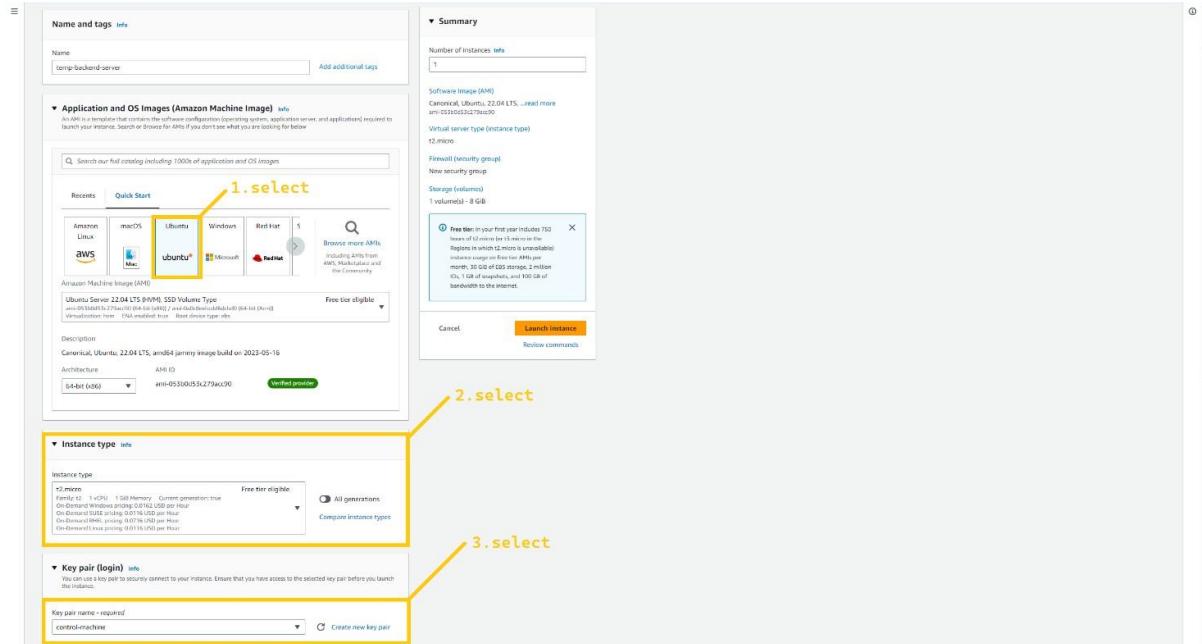
```
sudo npm install -g pm2
```



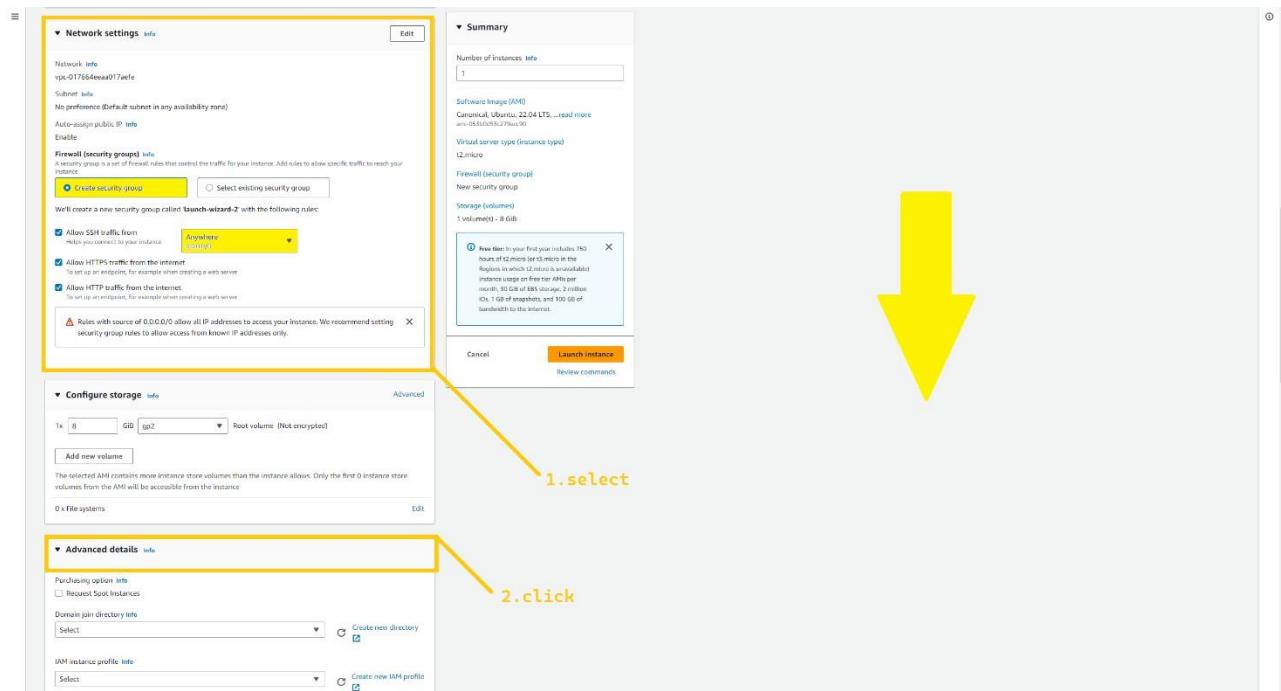
2.click

1.write

we have successfully launched **temp-frontend-server**. so now let's launch a temporary backend server. give a name to your instance (**temp-backend-server**). select ubuntu as the operating system. And select t2.mirco as instance type. Here we don't have to create a new key, we can utilize the previous key that we have created while launching the frontend instance.



In the network setting, we don't have to change anything just select whatever I have shown below image to keep things simple. And lastly please click on the advance details option.



Scroll down to the bottom of the page, and copy the bash script that I have given below. and paste it in the USER-DATA text box. This bash scripting installs some packages so that we don't have to install them manually. And click on the launch instance.

```

#!/bin/bash

sudo apt update -y

curl -fsSL https://deb.nodesource.com/setup_18.x |
sudo -E bash - && sudo apt-get install -y nodejs -y

sudo apt update -y

sudo npm install -g corepack -y

corepack enable

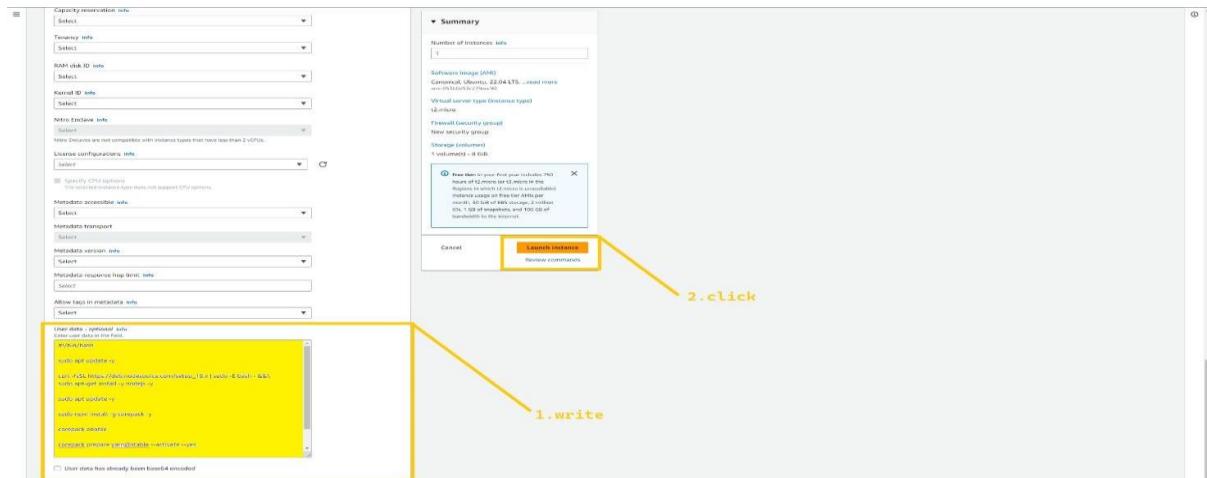
corepack prepare yarn@stable --activate

sudo npm install -g pm2

sudo pm2 startup

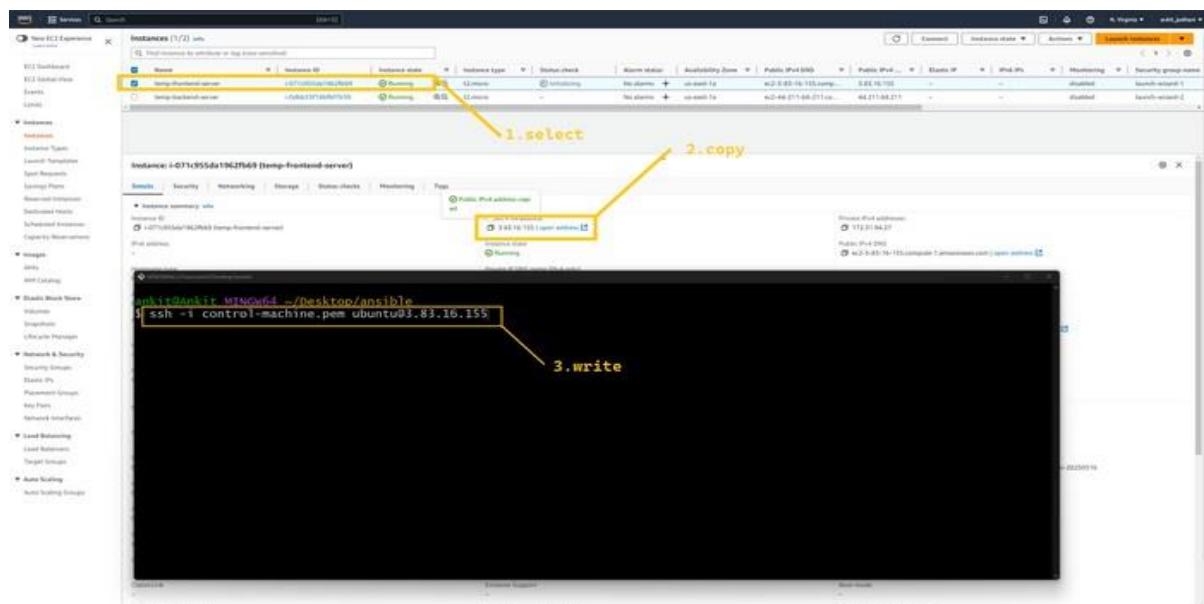
sudo pm2 save

```

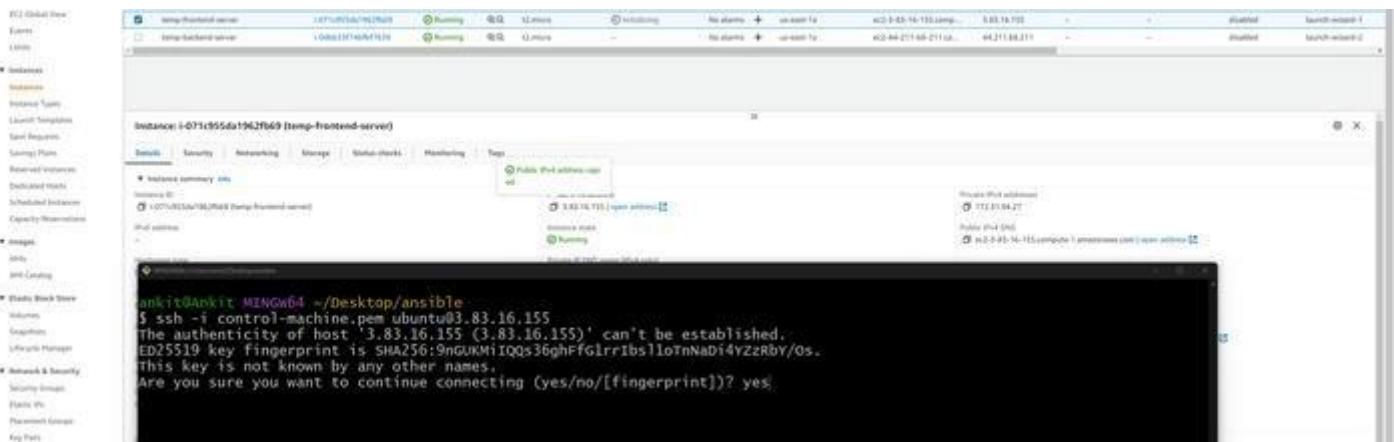


Please wait for 5-8 minutes so that the instance comes in a running state. and then we will utilize instances for further steps.

Select **temp-frontend-server**. and copy the IP address of the instance. Now open Gitbash where you have downloaded your **YOUR_KEY.pem** file. And type the command.
`ssh -i <name_of_key>.pem ubuntu@<Public_IP_add_of_Instance>`



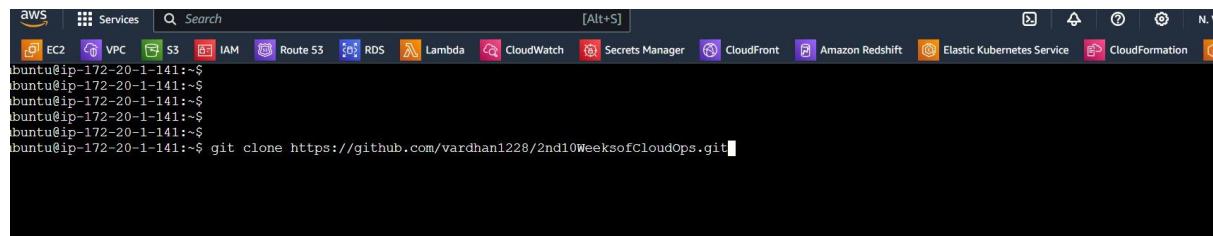
It will prompt you for your permission just type yes



Now you are successfully logged your remote **temp-frontend-server**. now our first task is to clone my git repo. If you are working on your own project then clone your repo. So type the command in the terminal.

The Github repository link is

git clone : <https://github.com/RKVankini/Highly-Available-AWS-Multi-Region-3-Tier-Architecture.git>



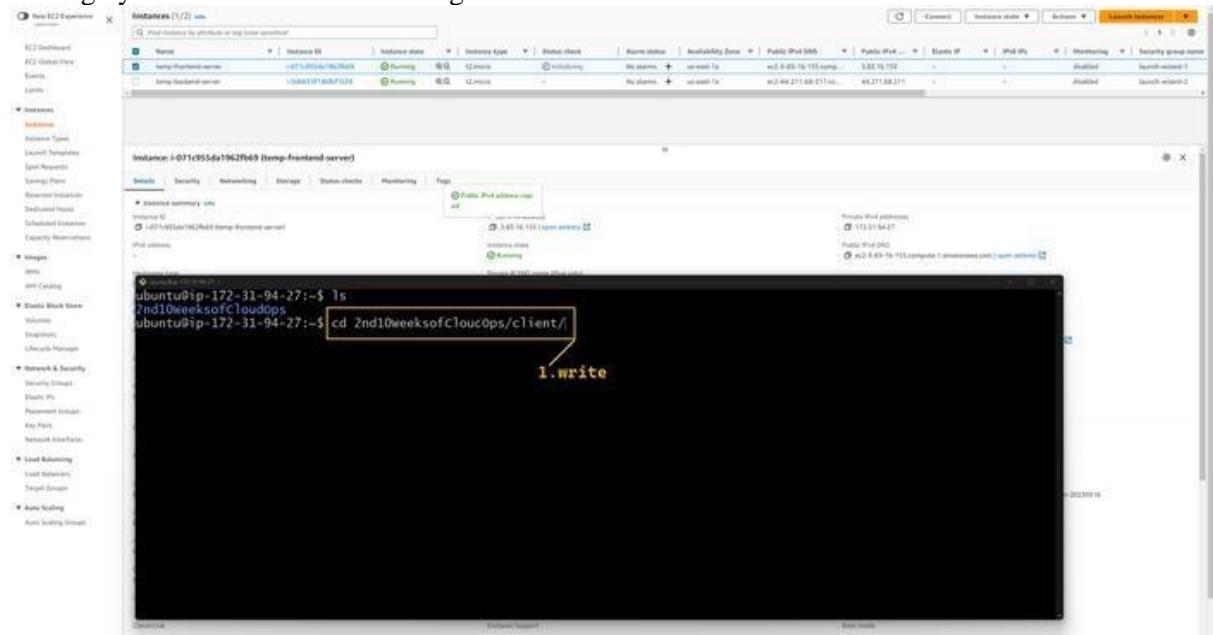
Go inside the directory. Cd

/ Highly-Available-AWS-Multi-Region-3-Tier-Architecture cd

/client

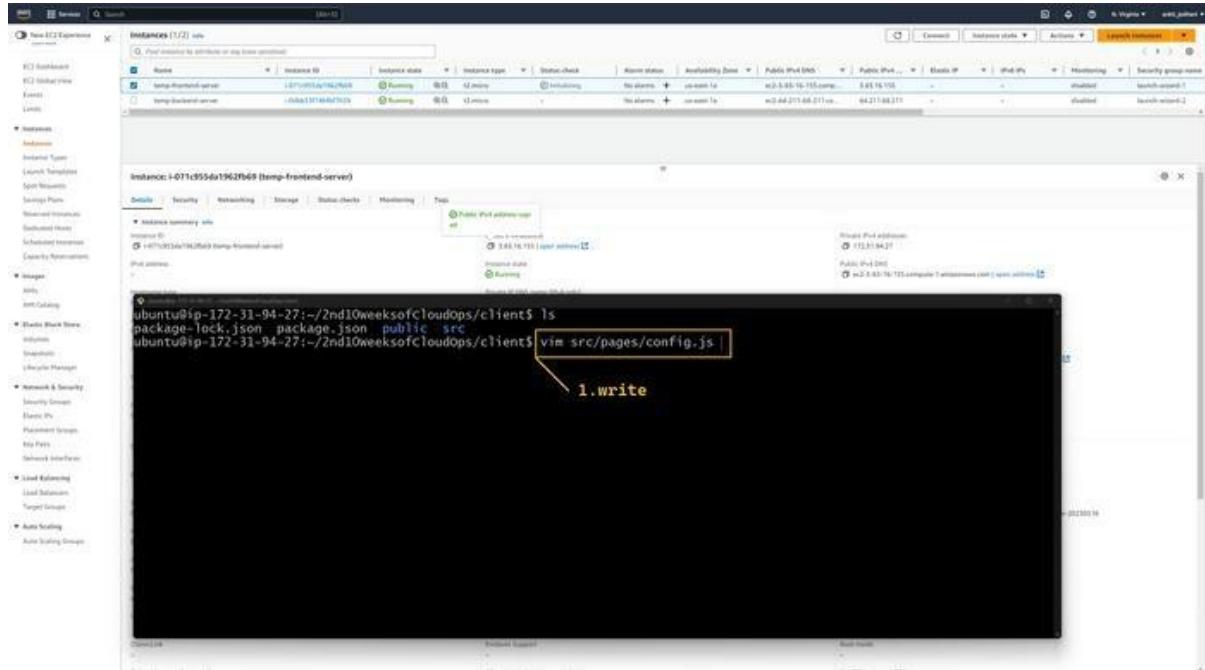
Or

Cd Highly-Available-AWS-Multi-Region-3-Tier-Architecture/client

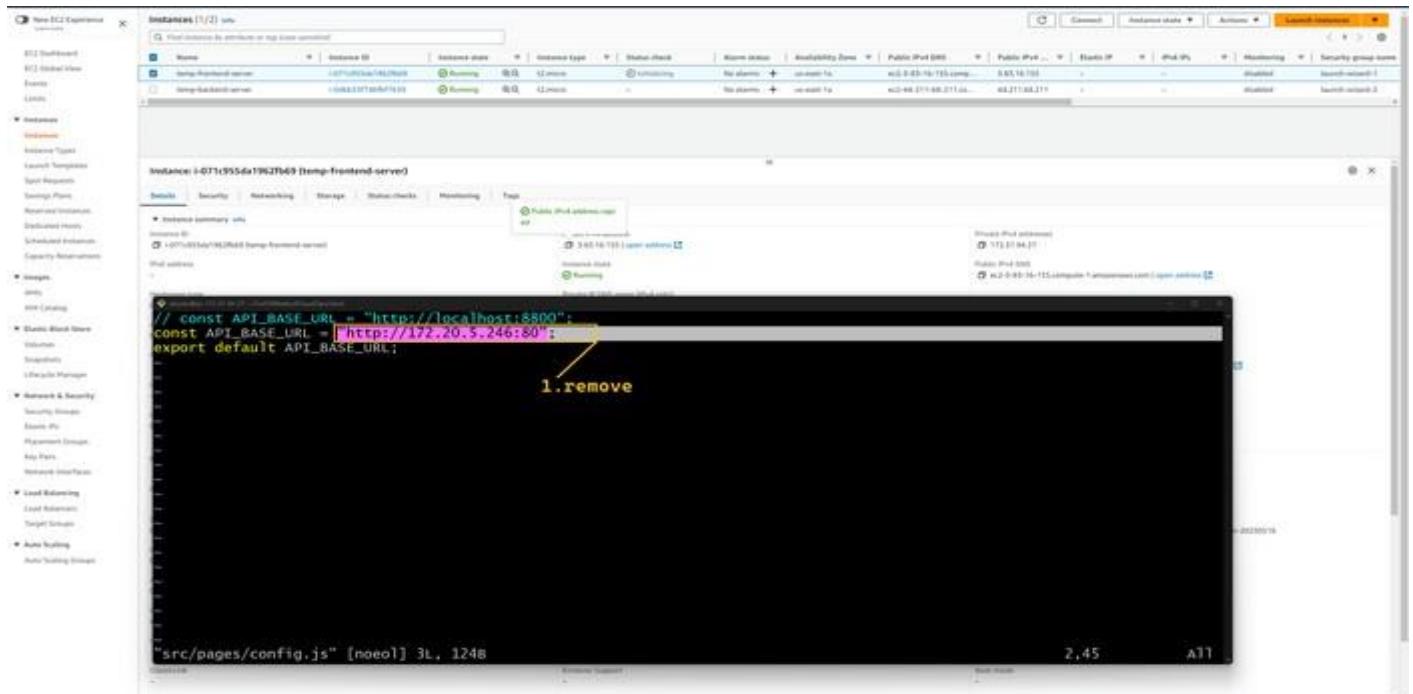


Now, we need to change just one line in our frontend application that is built in React.

So type the command
vim src/pages/config.js



The above command opens the file in a text editor. Now press I the button on your keyboard to edit the file. In this file, we have to change API_BASE_URL. So remove whatever is present in the API_BASE_URL variable.



And add <https://api.ramakrishna.shop>, In my case I have added this URL but in your case it is different. This means you need to use your OWN domain name. so your API_BASE_URL should be like https://api.<YOUR_DOMAIN_NAME>.XYZ I hope it makes sense. After updating the variable press ESC key on your keyboard and then type :wq and hit the Enter button.

API_BASE_URL = <https://api.ramakrishna.shop>

```

// const API_BASE_URL = "http://localhost:8800";
const API_BASE_URL = "https://api.vardhan.live";
export default API_BASE_URL;

```

After making these changes our frontend of the application will send all the API calls on the domain name <https://narni.co.in> And lastly, that will point to our backend server.

Now type the command `npm install` in the terminal to install all the required packages.
`npm install`

```

ubuntu@ip-172-31-94-27:~/2nd10WeeksofCloudOps/client$ ls
package-lock.json package.json public src
ubuntu@ip-172-31-94-27:~/2nd10WeeksofCloudOps/client$ vim src/pages/config.js
ubuntu@ip-172-31-94-27:~/2nd10WeeksofCloudOps/client$ npm install

```

1.hit command

Type the command `npm run build` to create the optimize static pages.

`npm run build`

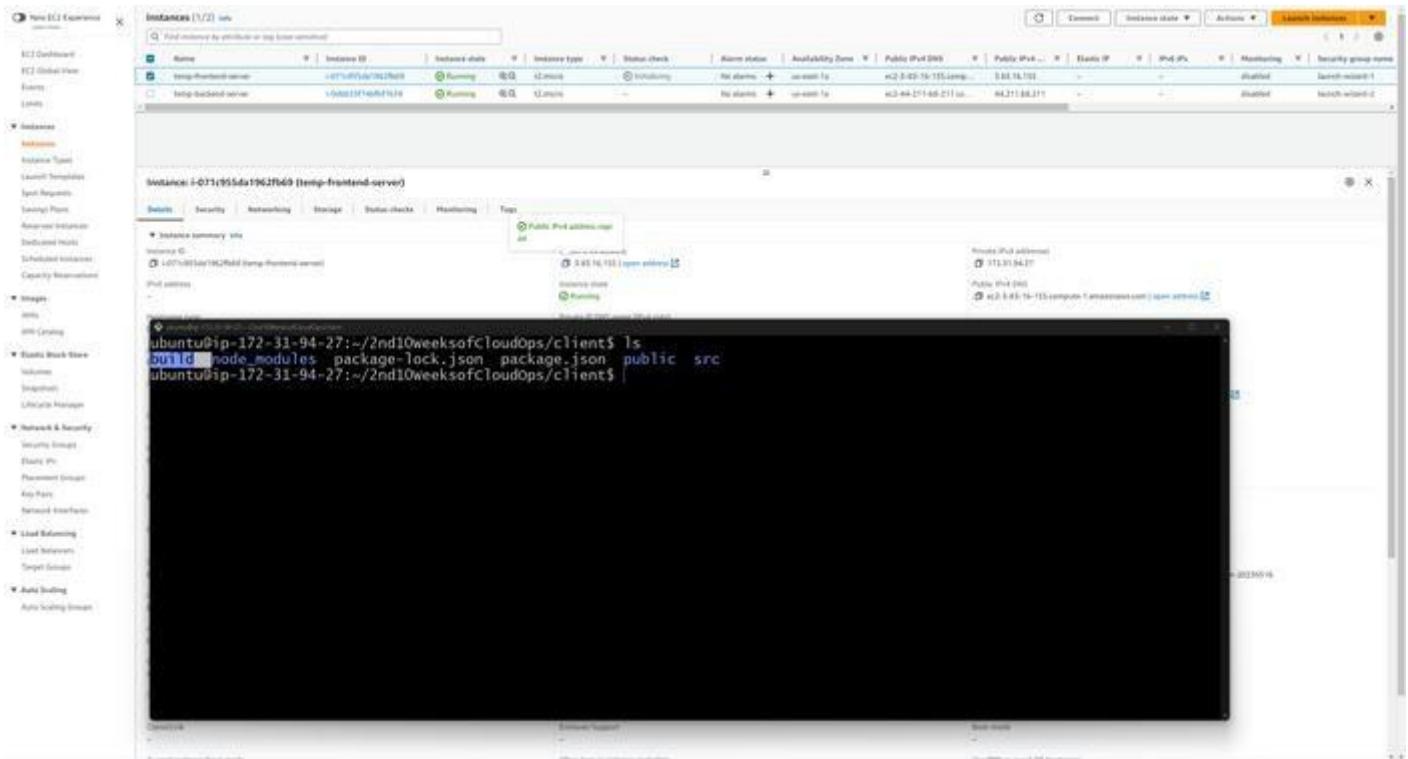
```

ubuntu@ip-172-31-94-27:~/2nd10WeeksofCloudOps/client$ npm run build

```

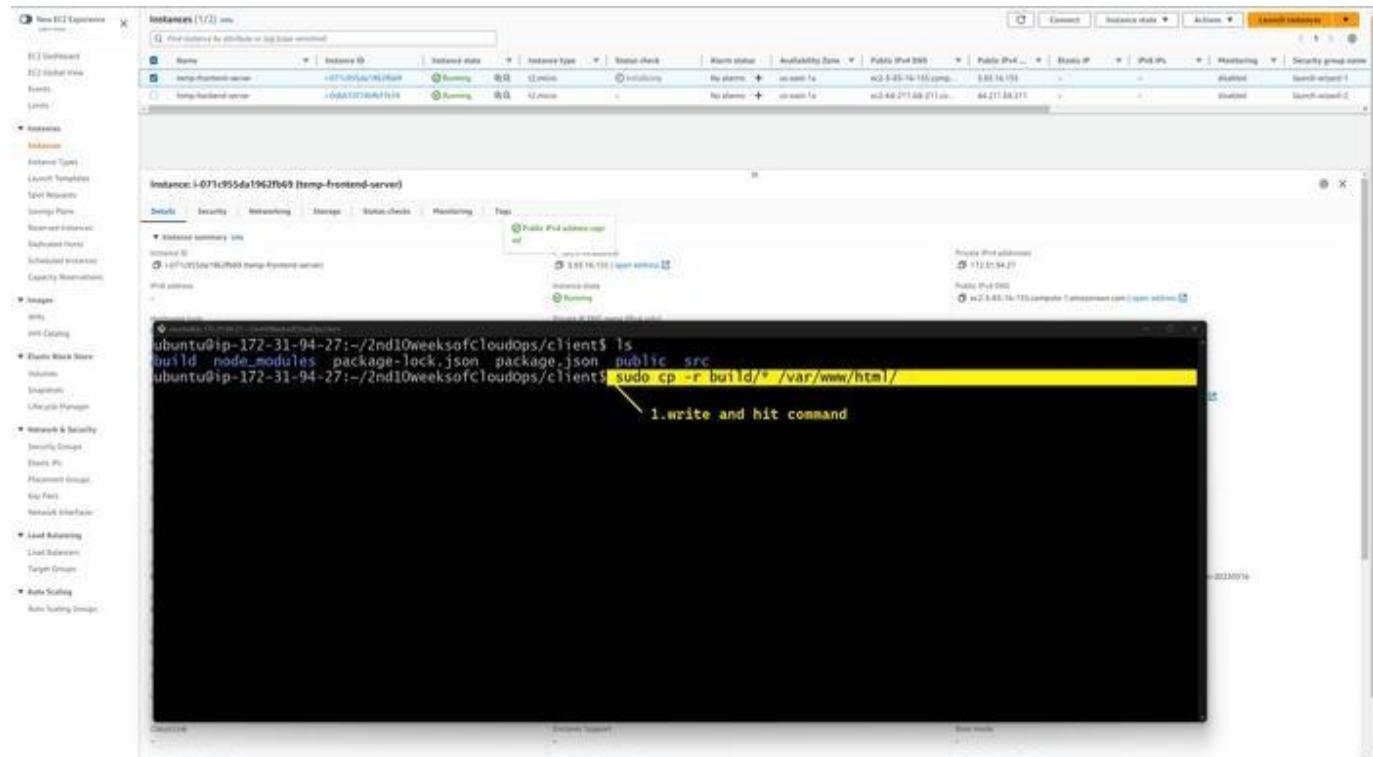
1.write and hit command

Now you have one more folder in the directory called build. You can verify that by typing ls command



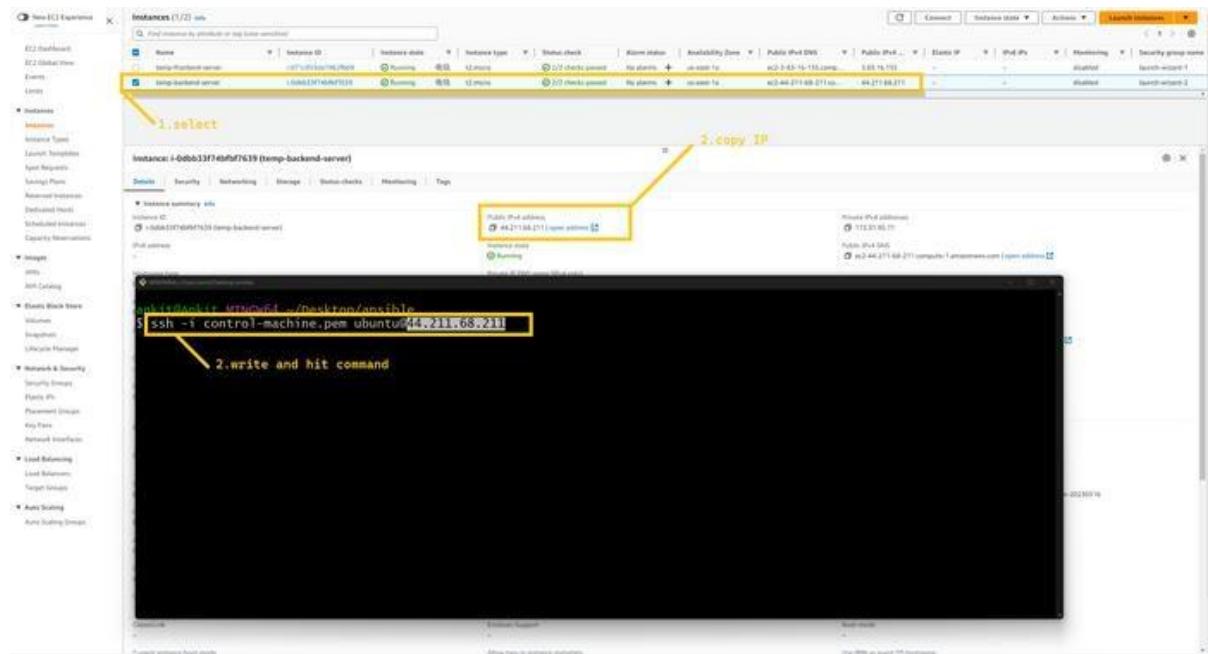
Now type the very essential command **sudo cp -r build/* /var/www/html/**

The above command takes all the static files from the build folder and stores them in /var/www/html so that Apache can serve them.



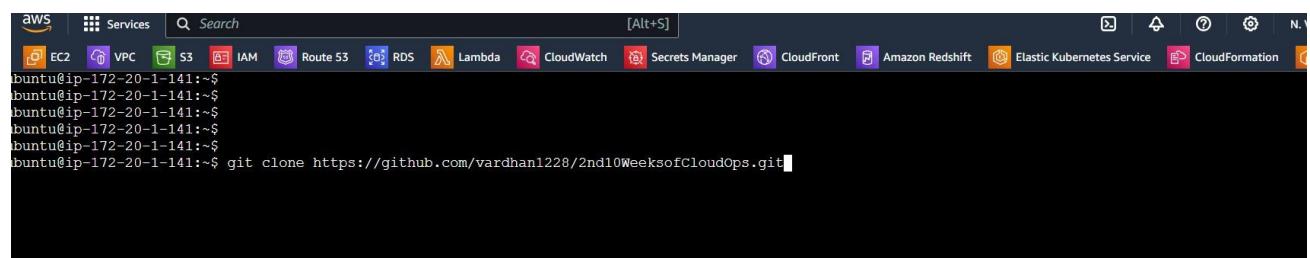
Here our temp-frontend-server configuration is completed. Now let's set up the temp-backend-server. So select the temp-backend-server and copy the IP address of the instance. Again please open Git bash in the same directory where your stored key.pem file. And type the below command

```
ssh -i name_of_your_key>.pem ubuntu@<Public_IP_addr>
```



We are successfully logged in inside the backend server. first, we will clone the repo.

git clone: <https://github.com/RKVankini/Highly-Available-AWS-Multi-Region-3-Tier-Architecture.git>

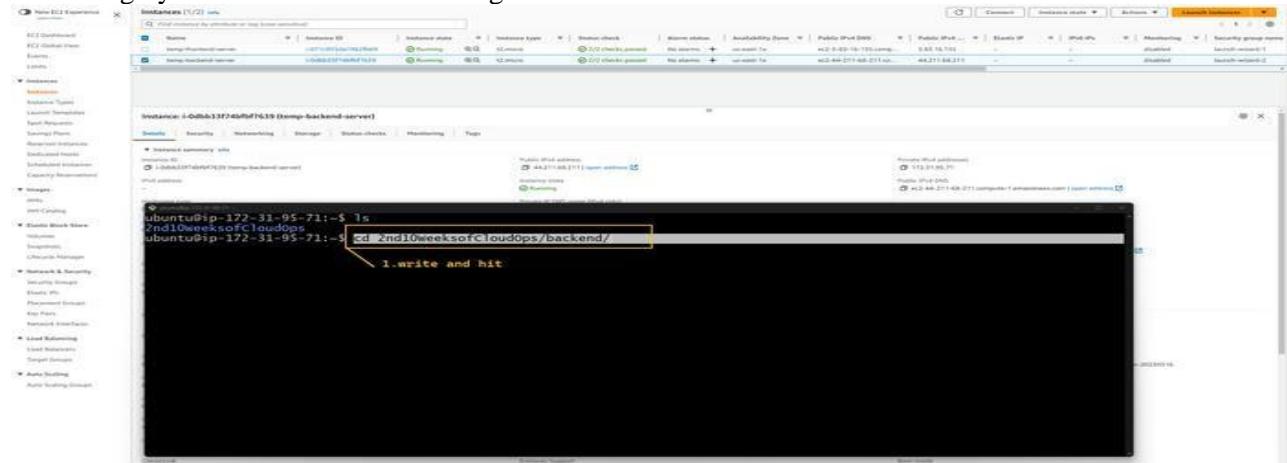


Go inside the directory. Cd

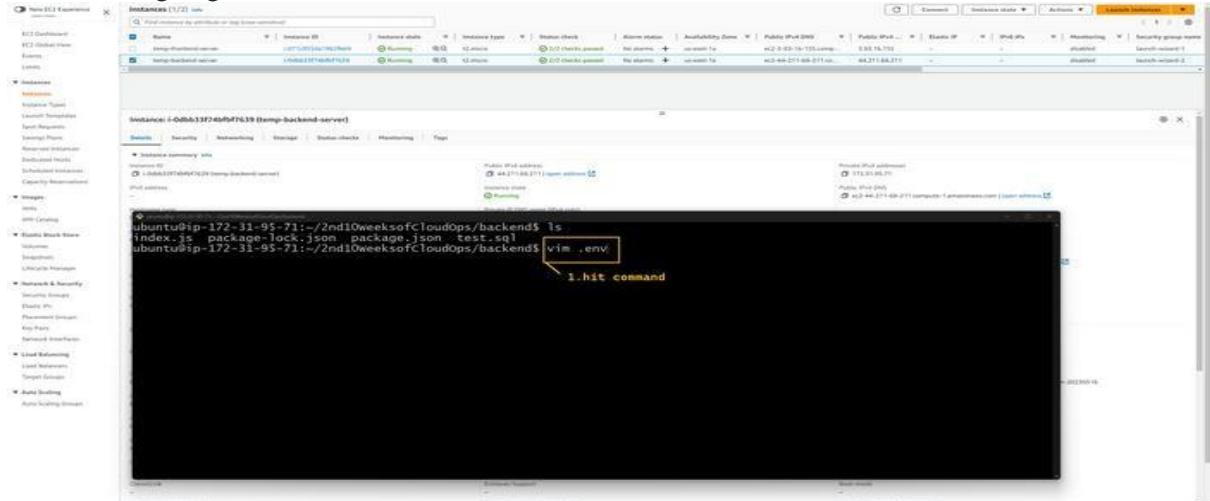
/ Highly-Available-AWS-Multi-Region-3-Tier-Architecture

cd backend

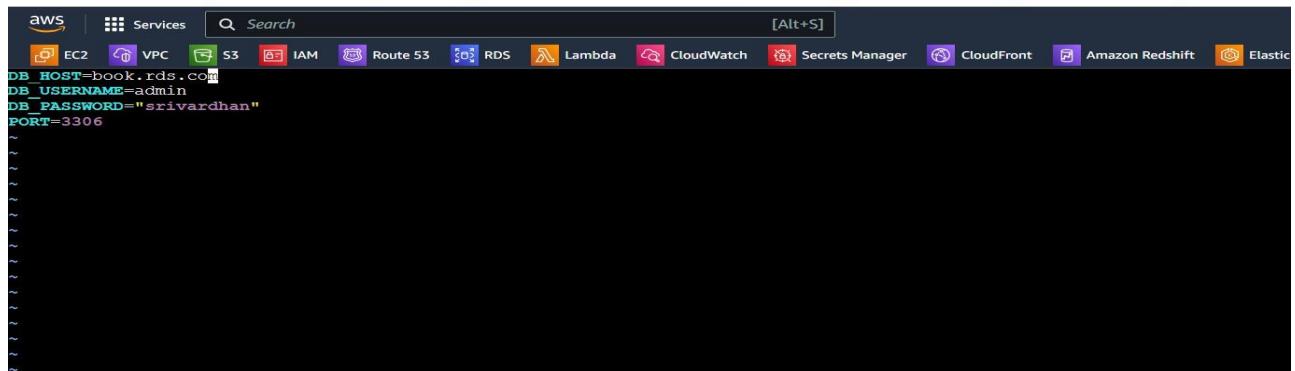
Or Cd Highly-Available-AWS-Multi-Region-3-Tier-Architecture/ backend



Here we are going to create one file with the name .env vim .env



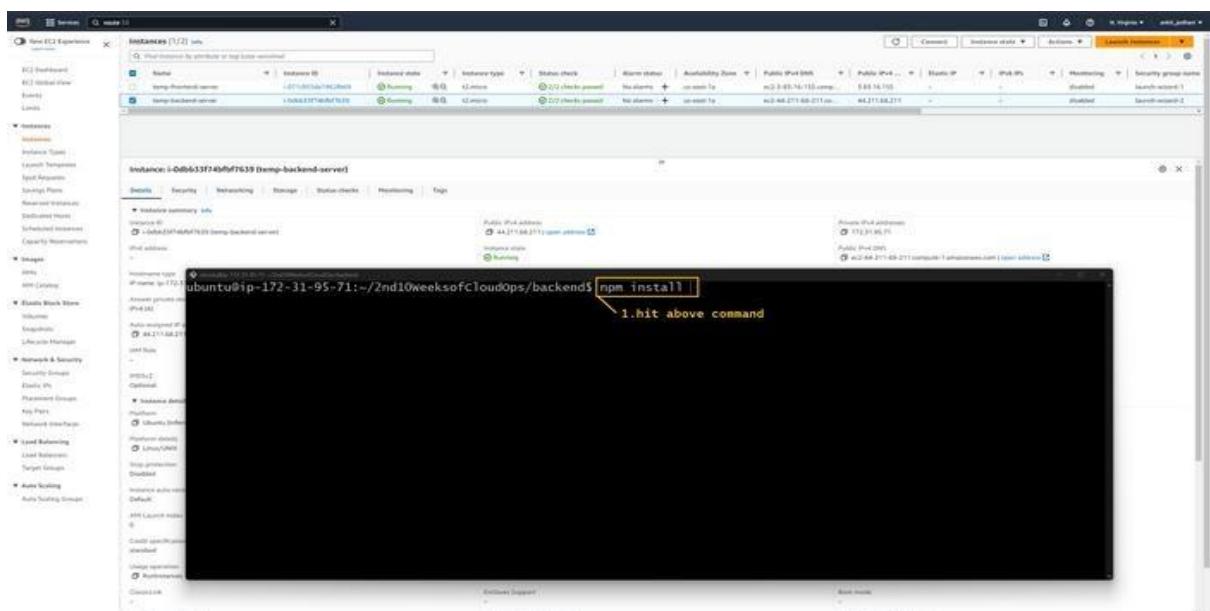
Press the I button on your keyboard. And copy the code given below and paste the snippet into the code editor. This code contains information about the RDS instance. Please change your username and password according to whatever you kept while creating a database. And then click on the ESC button and type :wq and hit the enter button

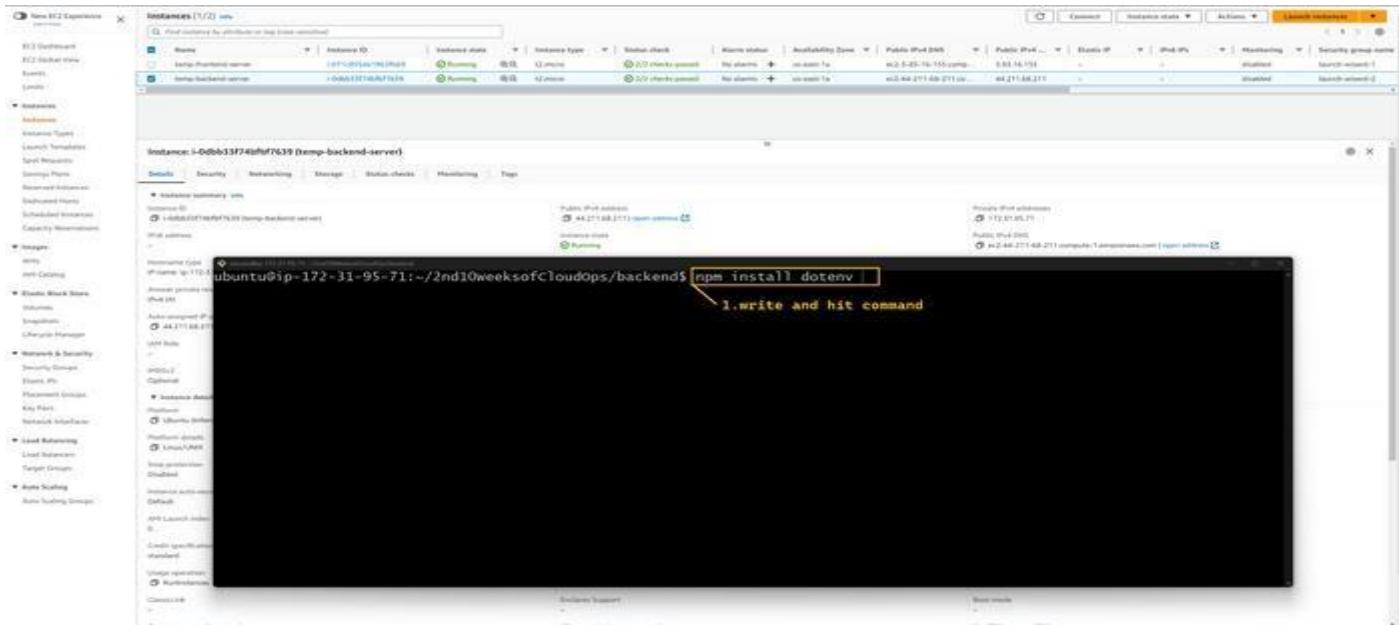


Now type the below commands in terminal

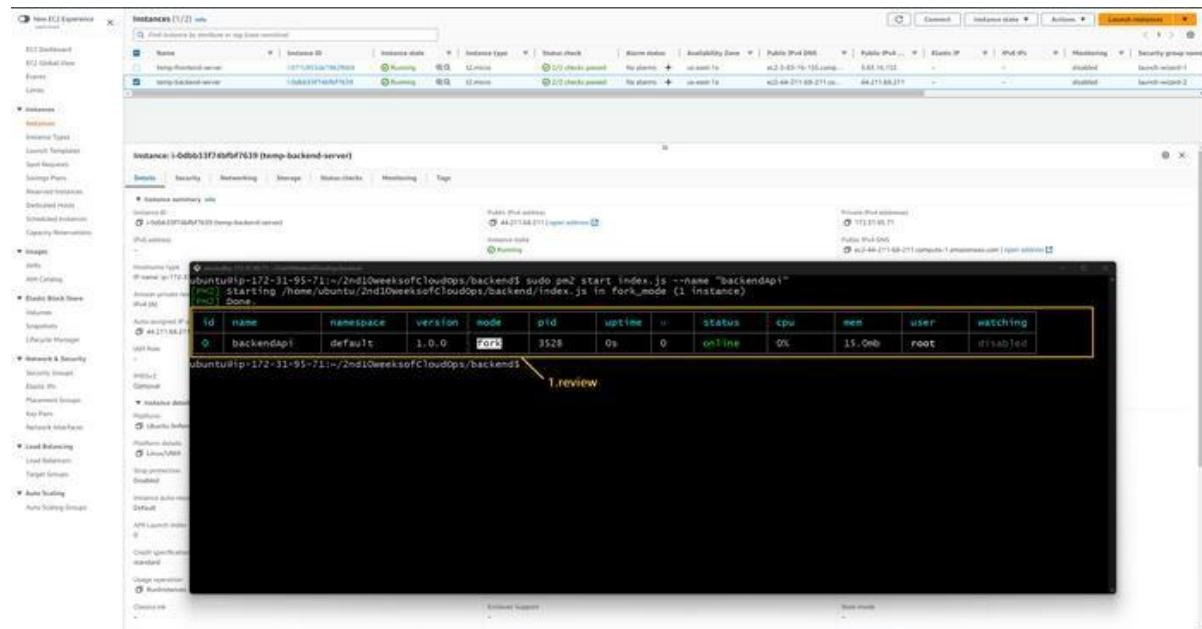
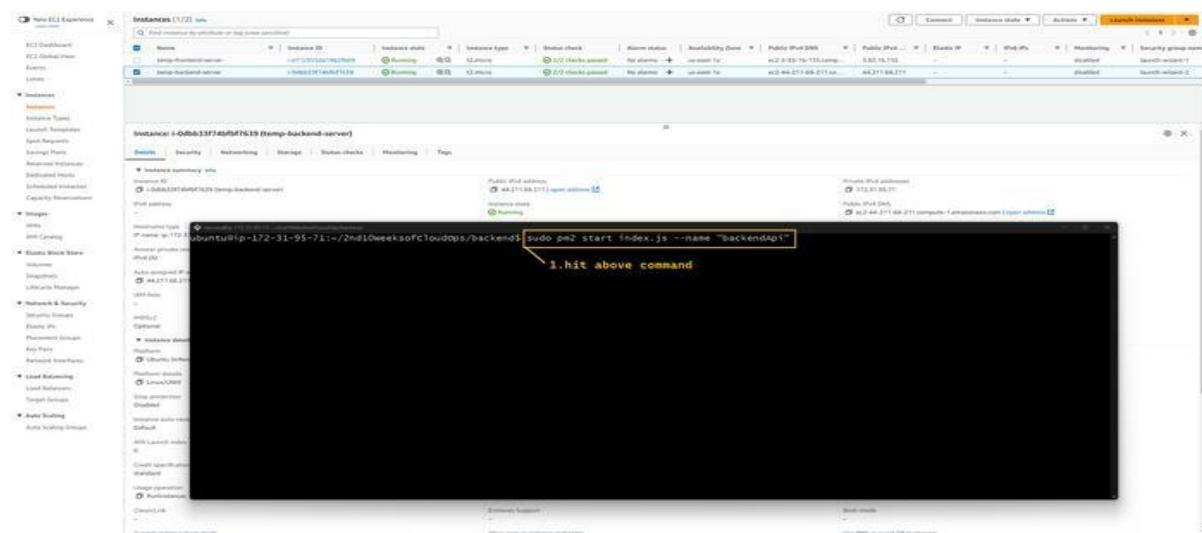
npm install

npm install dotenv





Now, let's start the backend server. (*very IMP*)
 sudo pm2 start index.js --name "backendApi"



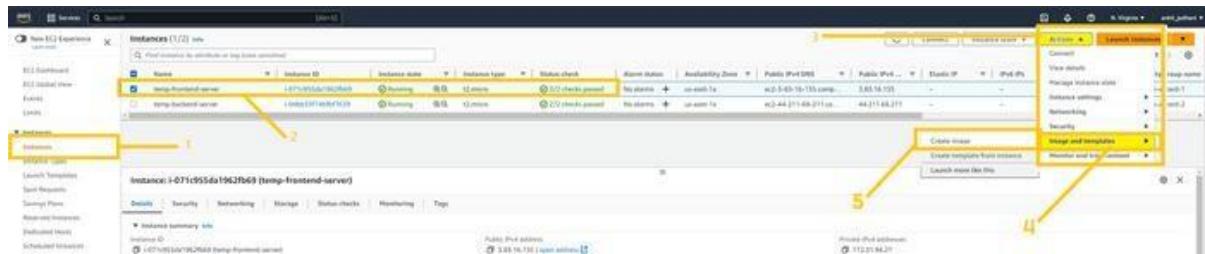
you can verify that by typing the command

```
sudo pm2 list
```

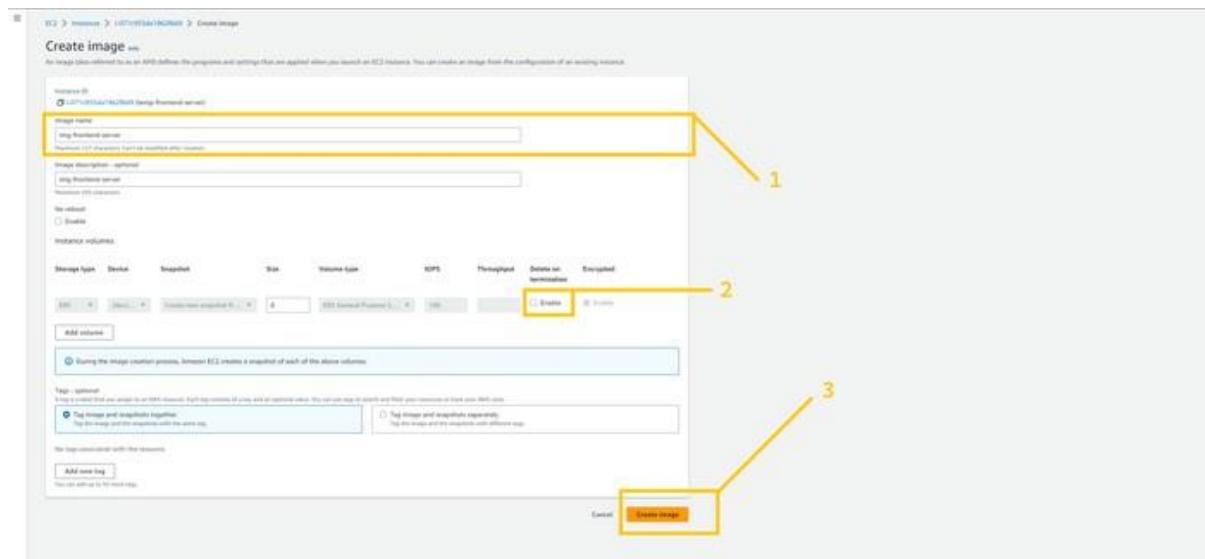
Yeah!!! 😊 Successfully completed our backend server configuration. You can close the terminal

But before we end this section we need to do a few more steps. We have to create Machine images of these servers so that we can create a launch template. *these steps are optional because anyhow we will take a backup from the AWS backup service and that will do the same thing. but that takes time. so it would be better if you follow the steps.*

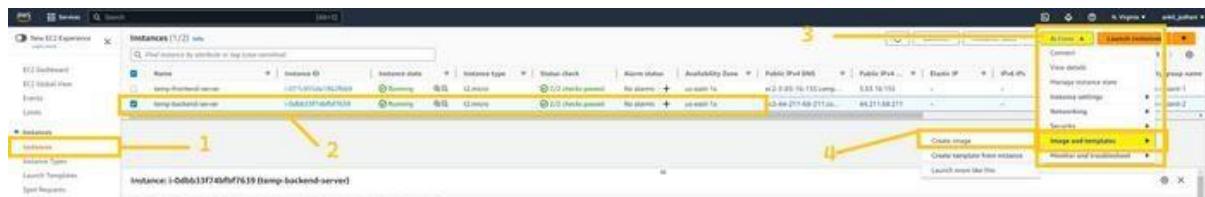
So please select temp-frontend-server and click on the Action button in the top right corner. One drop-down menu will open. You have to select the images and template option and that will give one more drop-down menu from which we need to click on create image button.



Give the name you your image (**img-frontend-server**). just deselect that delete on the termination button and click on the create image button.



You have to do the same thing for the temp-backend-server as well. I have shown you each and every step in the below images.





After a couple of minutes (10-15) you can see those images. Click on the AMIs button on the left panel and you can see both images here.

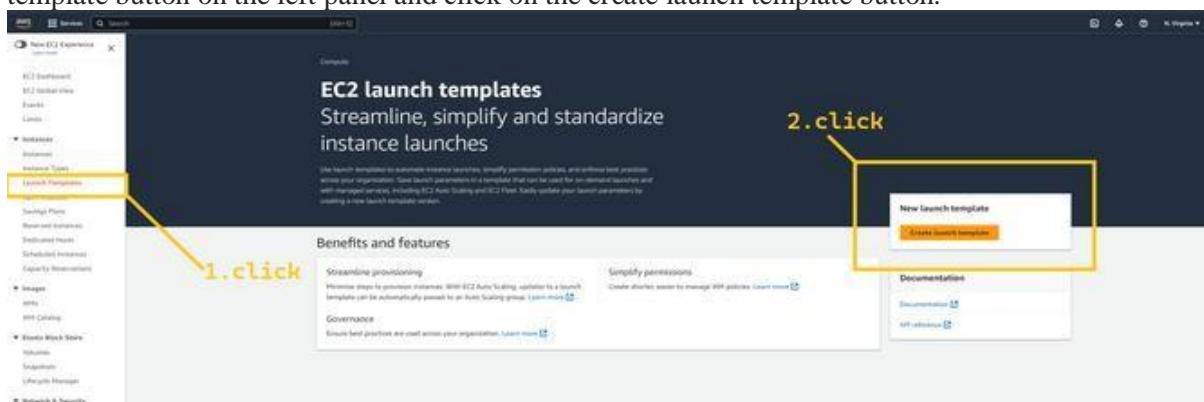


Note: Again we did the above setup in us-east-1 and we don't have to do this setup for us-west-2 we will leverage aws backup service or copy ami from one region to another region to copy these machine images in us-west-2.

◆ Launch Template

Take note that we need to create a launch template in both regions primary and disaster recovery (secondary) **us-east-1** and **us-west-2**. And now we have machine images in both regions.

First I will create a launch template in **N.virginia (us-east-1)**. So click on the launch template button on the left panel and click on the create launch template button.



Give the name to your launch template such as template-frontend-server as we are creating a launch template for frontend-server. let's give the version 1 in the version field. Here we need to select AMI so click on My AMIs tab and select the option owned by me. So now it will show you all the images that are present in your current region. If you are following the blog from starting then you will have a total of 4 images in **N.virginia**. coz two we created manually and two were created by backup service. Here you have to select the image that contains the frontend application. Either you can select the manual or the one created by the backup service. both are okay coz it contains the same data. Select instance type t2.micro

Launch template name and description

Launch template name - required/
template-frontend-server
Must be unique to the account. Max 128 chars. No spaces or special characters like %, ^, @.

Template version description
1
Max 255 chars

Auto Scaling guidance info
Select this if you intend to use this template with EC2 Auto Scaling
 Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► Template tags
► Source template

Launch template contents
Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

Application and OS Images (Amazon Machine Image) info
An AMI is a template that contains the software configuration (operating system, applications, services, and applications) required to launch your instance. See the Amazon Machine Image (AMI) Overview for more information about what you are launching and how to use it.

Q Search our full catalog including 1000s of application and OS images

Recent: My AMIs Quick Start

Don't include in launch template
 Shared with me

Browse more AMIs including AMIs from AWS Marketplace and the Community

Amazon Machine Image (AMI)
img-frontendserver
ami-0b26ca509a9fb2f20
Description: img-frontendserver
Architecture: x86_64
AMI ID: ami-0b26ca509a9fb2f20

Summary

Software image (AMI)
img-frontendserver
ami-0b26ca509a9fb2f20
Virtual server type (instance type)
t2.micro
Firewall security group
frontendsg
Storage (volumes)
1 volume(s) - 8 GB

Free tier: In your first year includes 750 hours of compute usage per month in the Regions in which Lambda is available
instance usage on free tier: 400 per region per month (100 hours, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet)

Create launch template

Scroll down, attach the key pair, and in the network setting just select the security group that we created for the frontend server. in my case the name **SG is frontend-sg**. And click on the advance details section at the bottom of the page.

Key pair (login) info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name
control-machine
 Create new key pair

Network settings info
Subnet info
 Don't include in launch template
 Create new subnet

Firewall security group info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
 Select existing security group
 Create security group

Security group info
Select security groups
frontendsg
 Frontend security group
 Frontend backup security group

Storage (volumes)
EBS Volume
Volume 1 (Amazon EBS) (8 GiB, EBS, General purpose SSD (gp2))
AMI volumes are not included in the template unless modified
Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

Resource tags info
No resource tags are currently included in this template. Add a resource tag to include it in the launch template.
Add new tag

You can add up to 50 more tags:

Advanced details info

Summary

Software image (AMI)
img-frontendserver
ami-0b26ca509a9fb2f20
Virtual server type (instance type)
t2.micro
Firewall security group
frontendsg
Storage (volumes)
1 volume(s) - 8 GB

Free tier: In your first year includes 750 hours of compute usage per month in the Regions in which Lambda is available
instance usage on free tier: 400 per region per month (100 hours, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet)

Create launch template

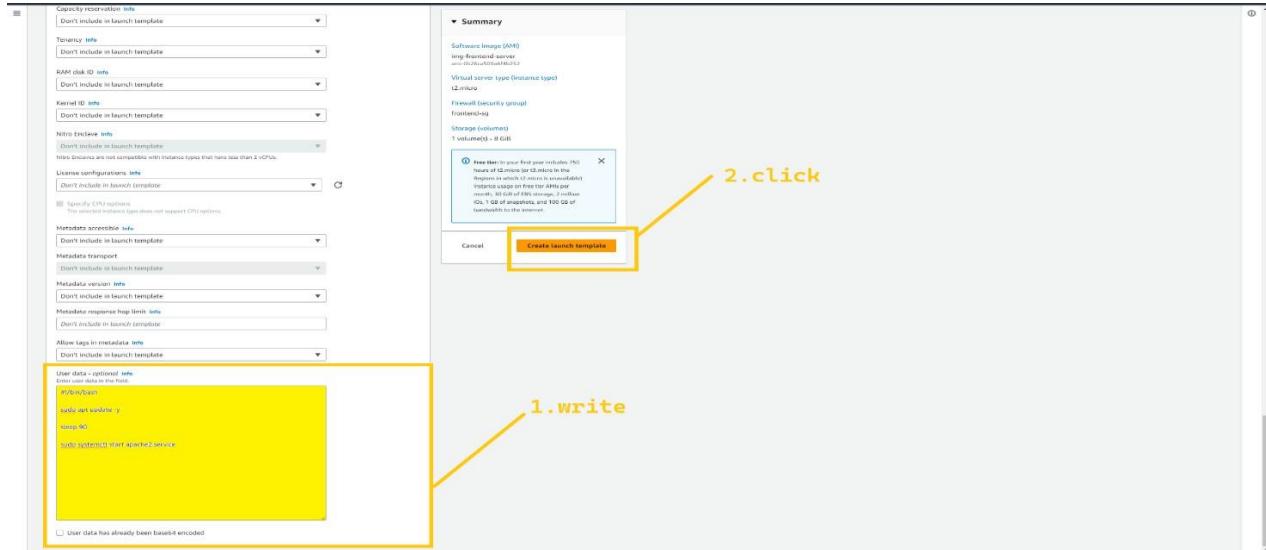
Scroll down to the bottom, and in the USER-DATA text box paste the code that I have given below. And then click on the Create launch template button.

```
#!/bin/bash
sudo apt update
```

```
-y sleep 90
```

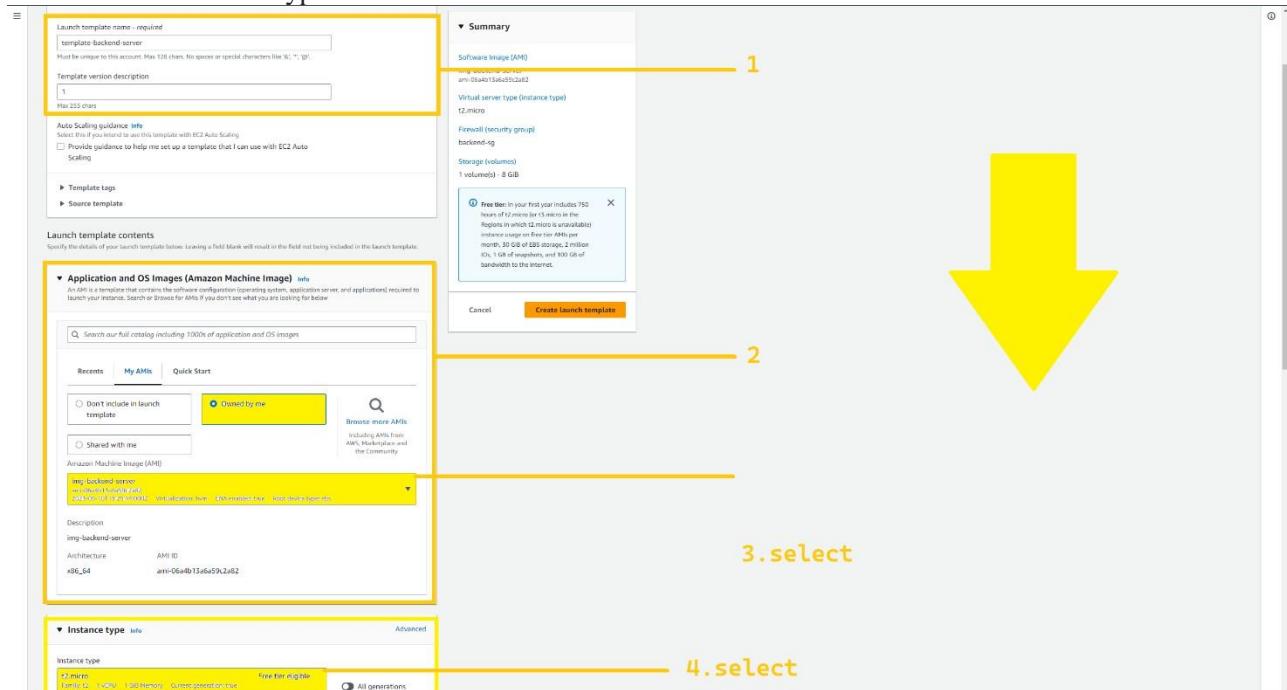
```
sudo systemctl start apache2.service
```

```
sudo systemctl enable apache2
```

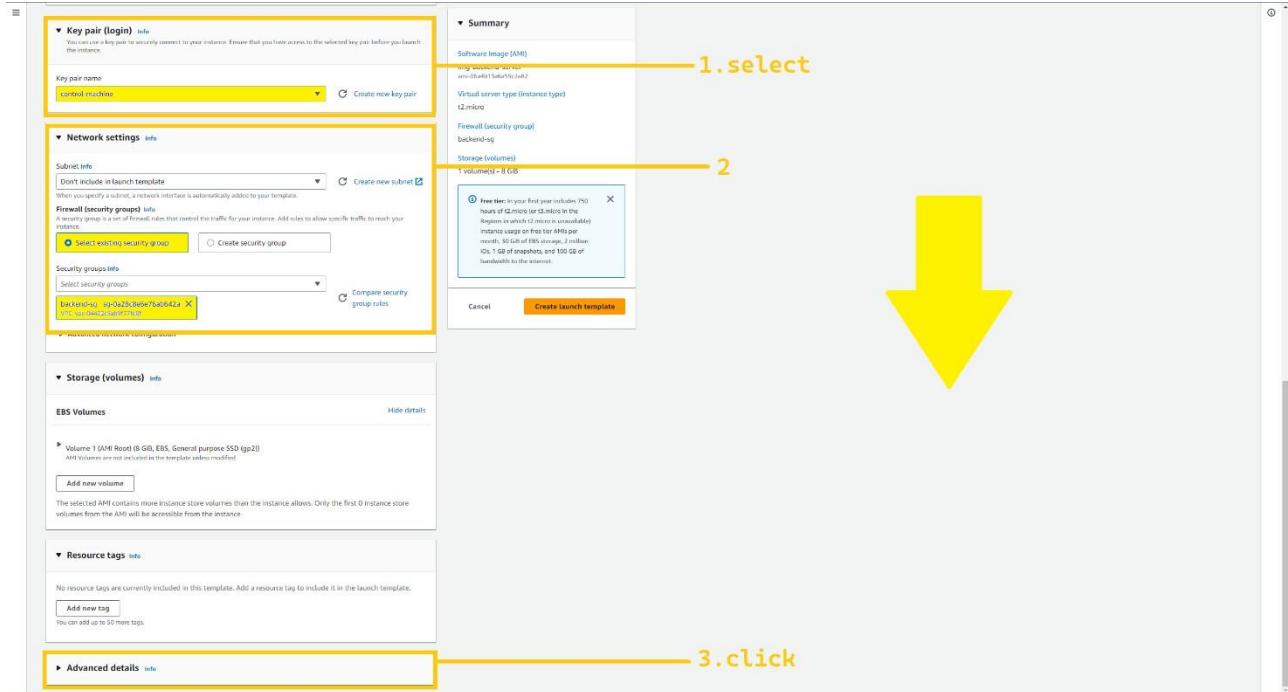


we successfully created a launch template for the frontend-server. now let's create a launch template for the backend server.

Give a name to your launch template (template-backend-server). Give version 1 in the version field, but make you select the correct AMI that holding your backend application. And select an instance type t2.micro



Select the key pair, and in the network setting just select the security group that we have created for the backend server. in my case name of the SG is [backend-sg](#). And click on the advance details section at the bottom.



Scroll down to the bottom, and in the USER-DATA text box paste the code that I have given below. And then click on the Create launch template button.

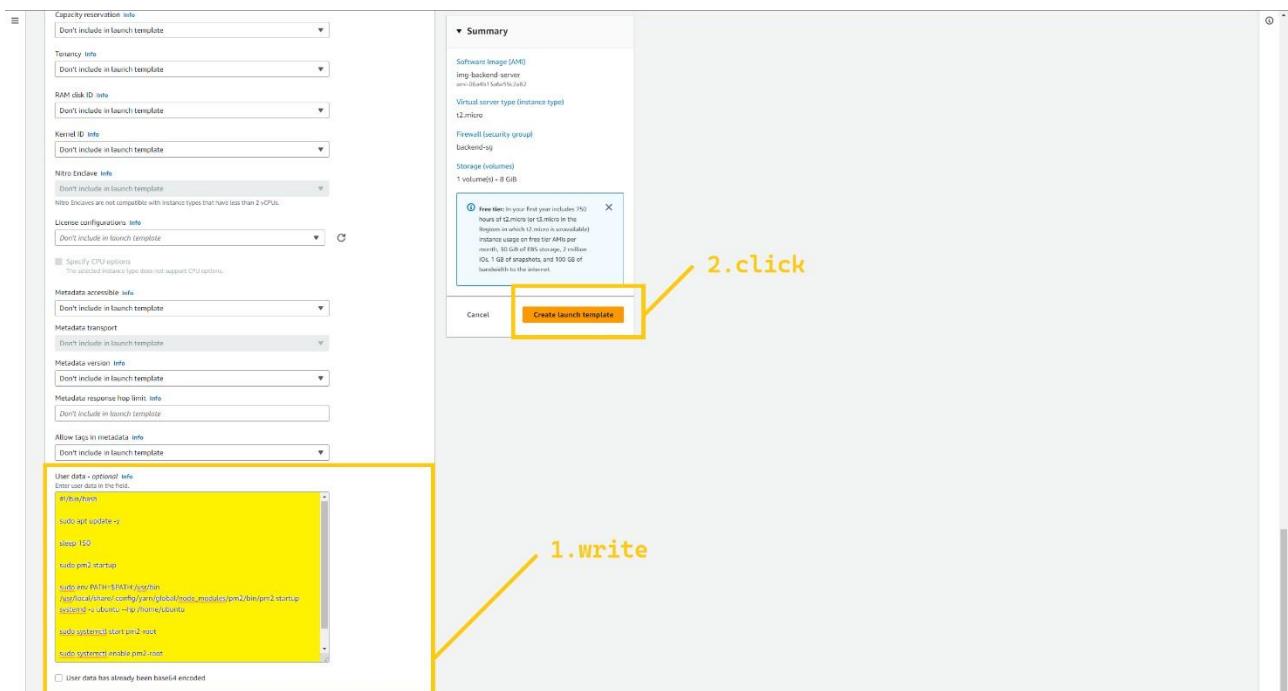
```
#!/bin/bash
```

```
sudo apt update -y
```

```
sleep 150
```

```
sudo pm2 startup
```

```
sudo pm2 save
```



We have created two launch templates, template-frontend-server and template-backend-server in **N.virginia**.

Note: Your task is to set up two launch templates in the **Oregon** region (disaster recovery region). Everything is completely similar just you have to change AMIs. Please select the correct AMI for the frontend and backend. If you have difficulties finding AMIs you can compare the instance_id with temp-frontend-server and temp- backend-server. this will definitely help you.

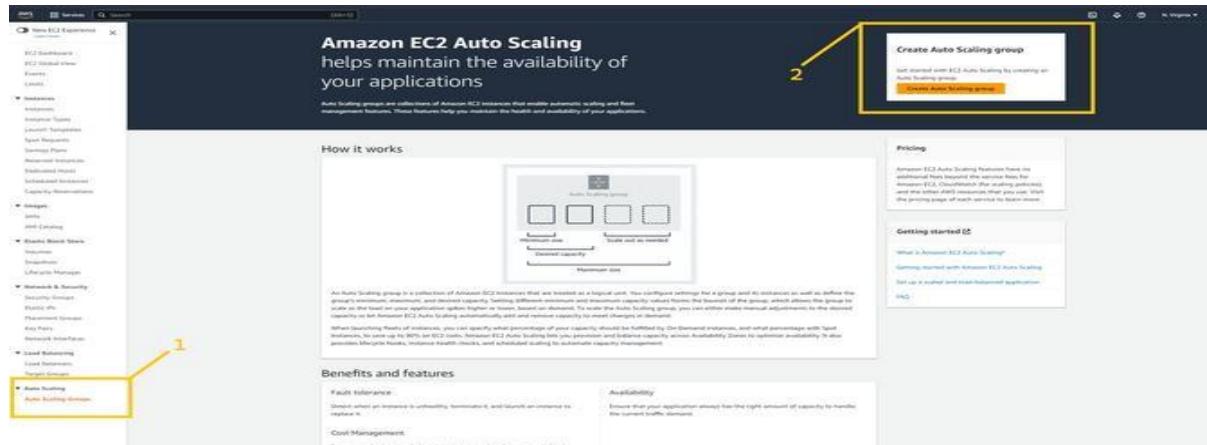
Now we can terminate those temp-servers to save the bills.

◆ Auto scaling group (ASG)

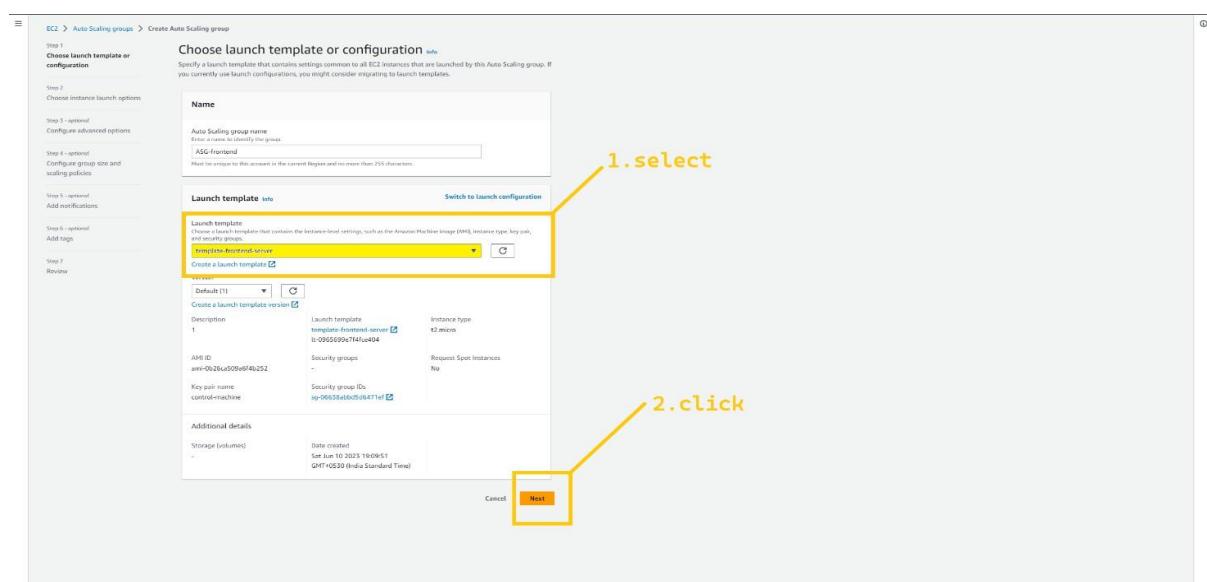
The auto-scaling group is the functionality of EC2 service that launches instances depending on your network traffic or CPU utilization or parameter that you set. It launches instances from the launch template.

Note: we need to set up an Auto Scaling group in both regions **us-east-1 (N.virginia)** and **us-west-2 (Oregon, Disaster recovery region)**. First, we are going to set up ASG in **us-east-1**.

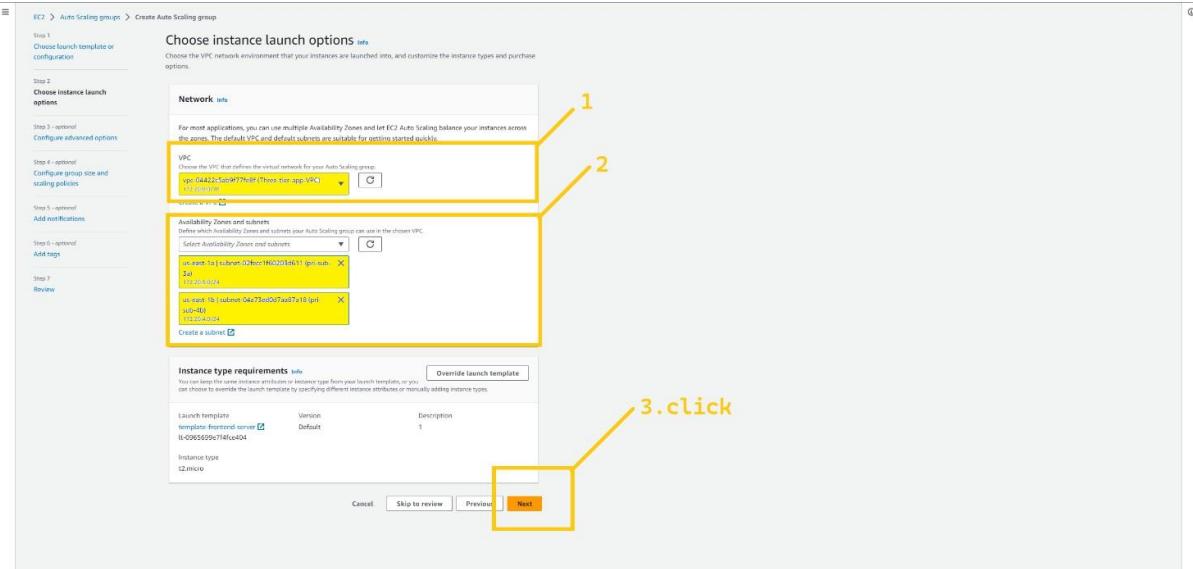
Click on the Auto scaling groups button which is located at the bottom of the left panel. And then click on the Create auto scaling group button.



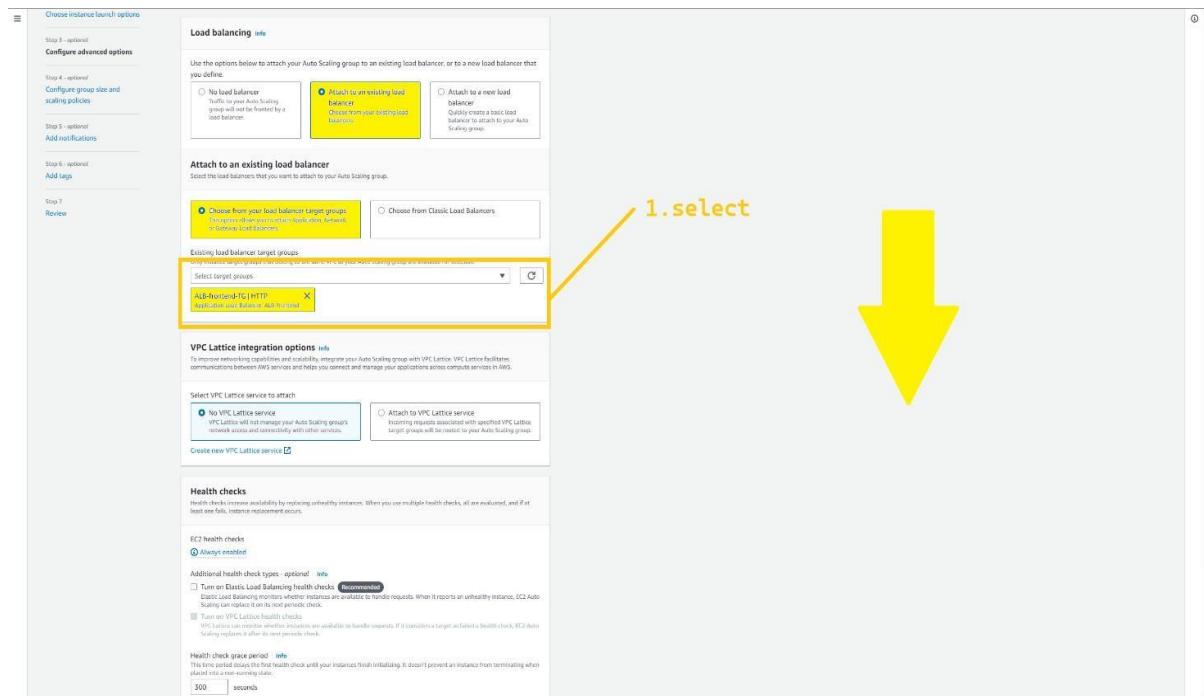
Give a name to your ASG. E.g ASG-frontend . And select the launch template that we have created for frontend (e.g template-frontend-server) in the launch template field. And click on the next button.



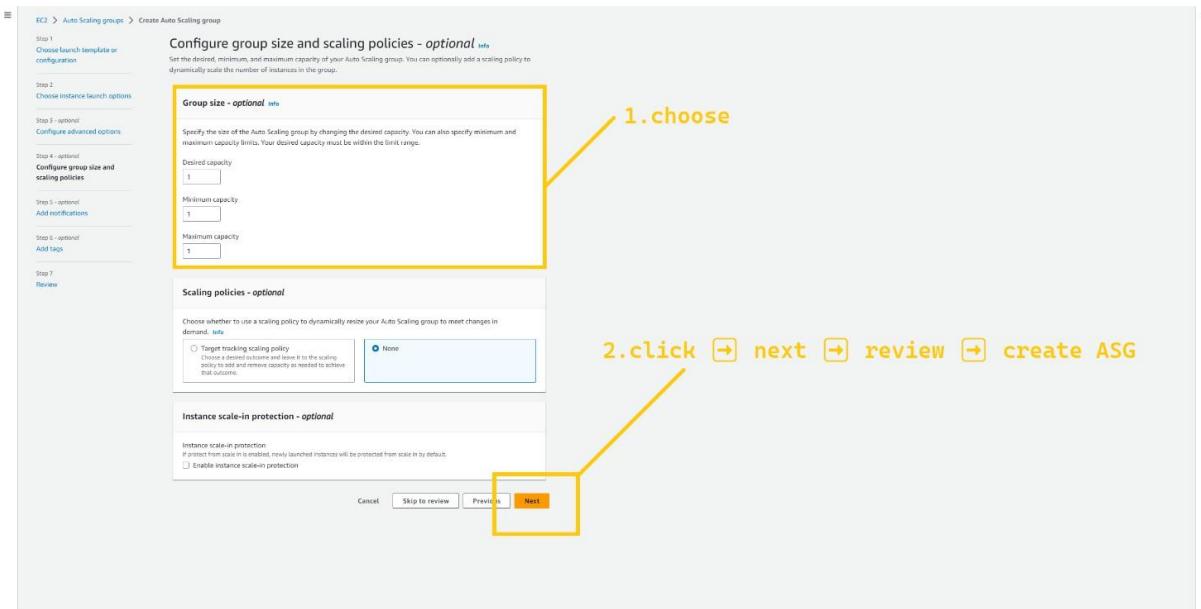
In the network field, you have to choose VPC that we created earlier. And in AZs and subnet filed choose pri-sub-3a and pri-sub-4b. these subnets we have created for frontend servers. And click on the next button.



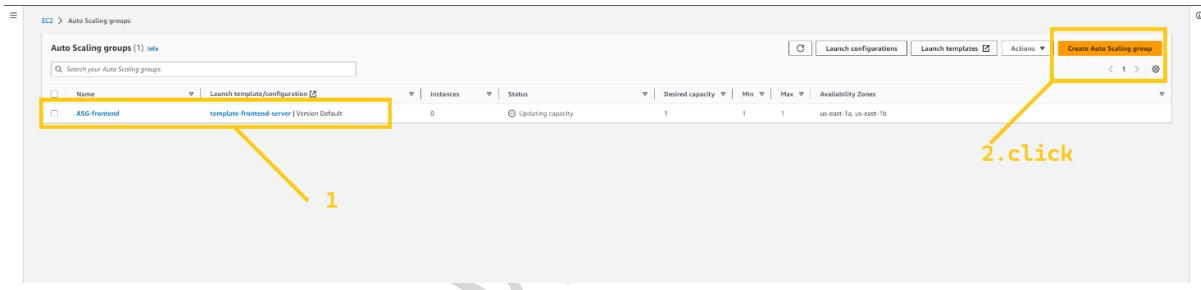
On this page we need to attach ASG with ALB so select the Attach existing ALB option and select TG that we have created for frontend e.g ALB-frontend-TG. And then scroll down and click on the NEXT button



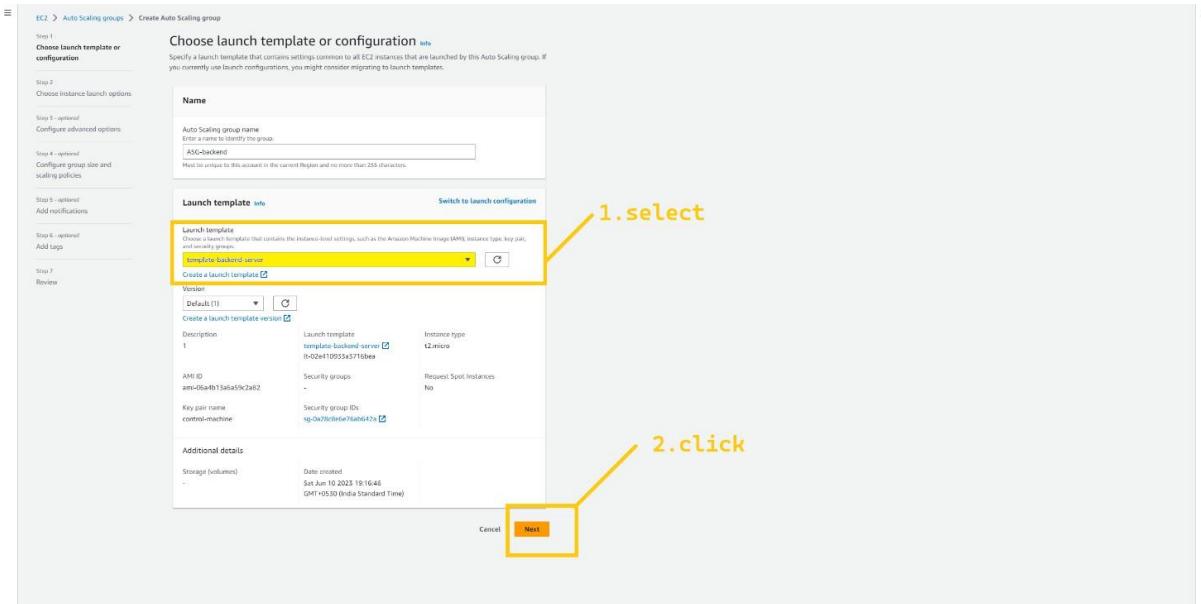
Here you can set the capacity and scaling policy but I'm keeping 1,1,1 to save cost but in real projects, it depends on the traffic. Click on the NEXT->next->next-> and create ASG button.



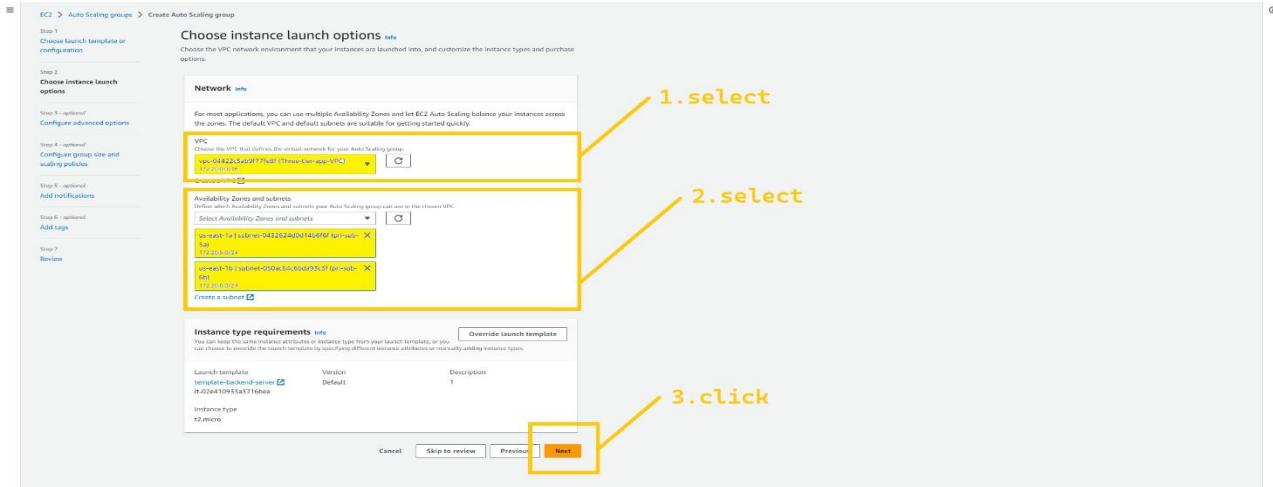
Let's set up ASG for the backend.



Give a name to your ASG. E.g ASG-backend. And select the launch template that we have created for the backend (e.g template-backend-server) in the launch template field. And click on the next button.



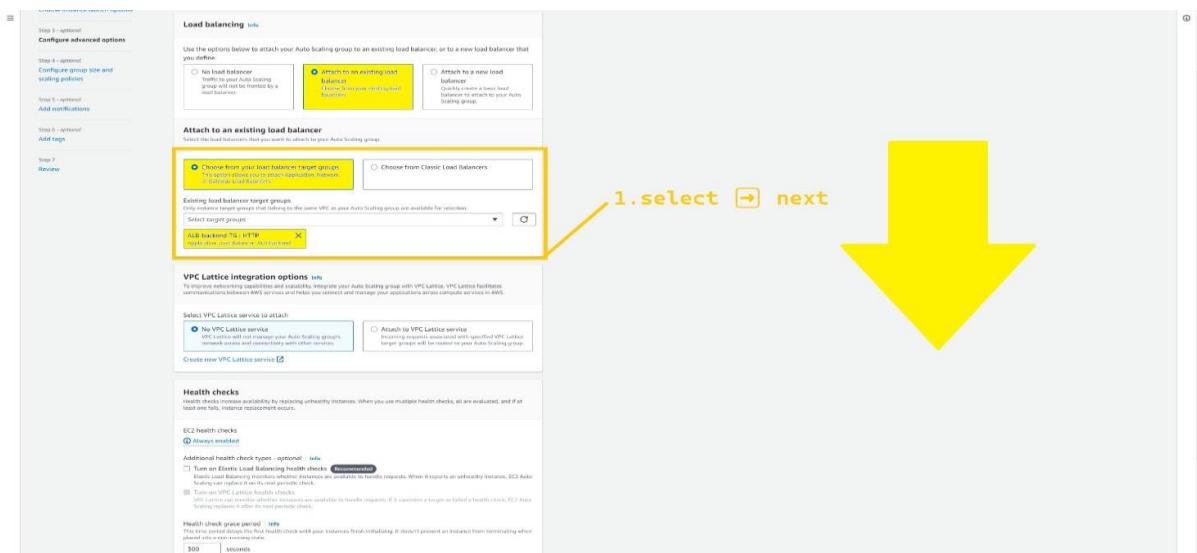
In the network field, you have to choose VPC that we created earlier. And in AZ and subnet field choose pri-sub-5a and pri-sub-6b. these subnets we have created for backend servers. And click on the next button.



On this page we need to attach ASG with ALB so select the Attach existing ALB option and select TG that we have created for the backend e.g ALB-backend-TG. And then scroll down and click on the NEXT button.



Here you can set the capacity and scaling policy but I'm keeping 1,1,1 to save cost but in real projects, it depends on the traffic. Click on the NEXT->next->next-> and create ASG button.



Now, We have two ASGs, ASG-frontend will launch frontend servers and ASG-backend will launch backend servers. we have successfully set up ASG in the N.virginia region and we need to do the same setup in the **Oregon region** as well.

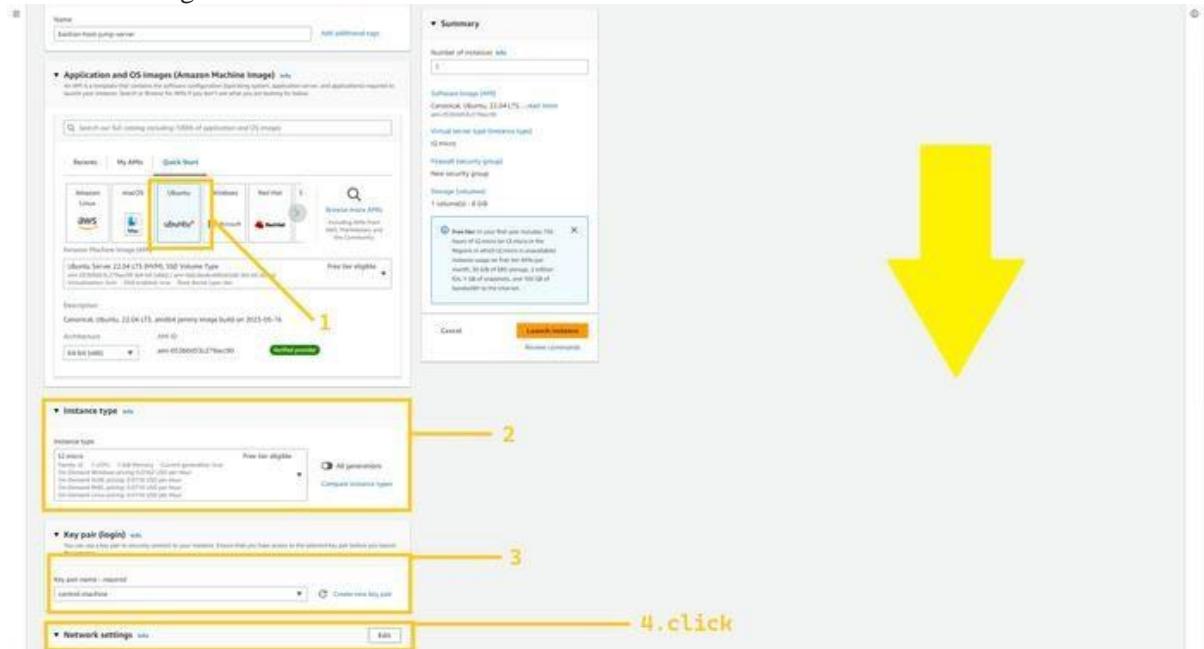
So your task is to set up ASGs in the **Oregon region (Disaster recovery region)**. I hope you successfully completed the given task.

Now before we go further one more thing we need to do. We need to initialize our database and need to create some tables. But we can't access the RDS instance or backend server directly coz they are in a private subnet and the security group won't let us login into it. So we need to launch an instance in the same VPC but in the public subnet that instance is called bastion host or jump-server. And through that instance, we will log in to the backend server, and from the backend server we will initialize our database.

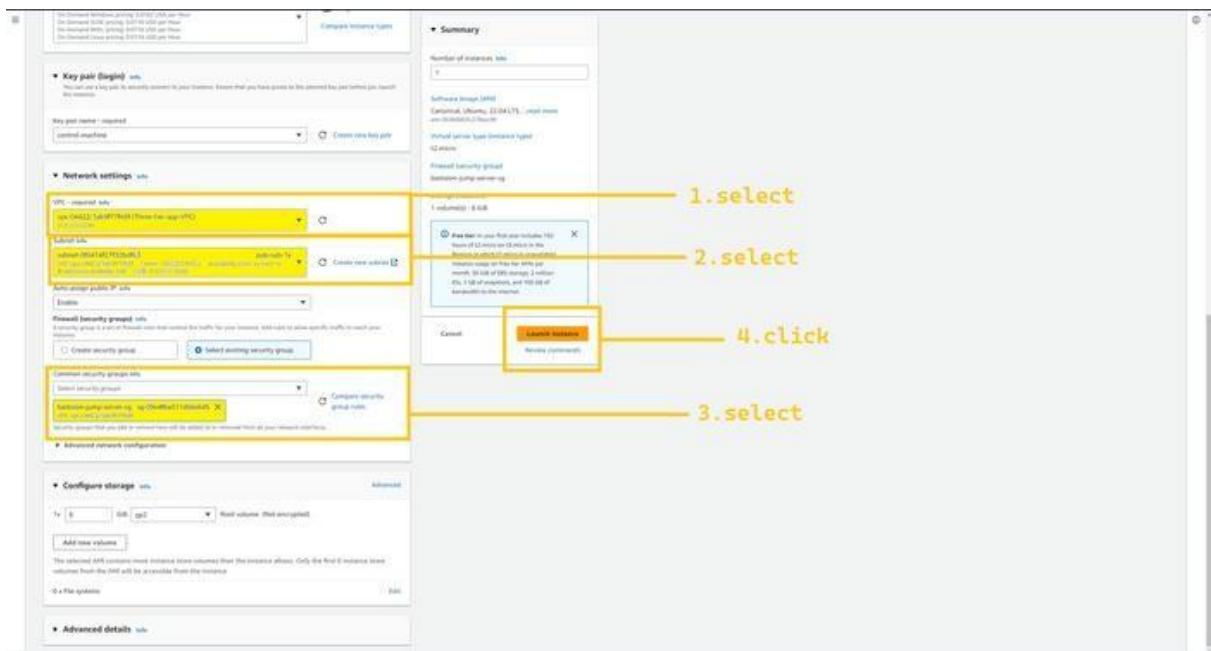
Click on the instance button on the left panel and click on the launch instance button in the top right corner. *Please terminate those temp-servers if you haven't*



Give a name to the instance (bastion-jump-server). Select Ubuntu as OS, instance type t2.micro, and select Key pair. In all the instance and launch template we have used only **one key** so it will be easy to login in any instance. And then click on the Edit button of the Network setting.



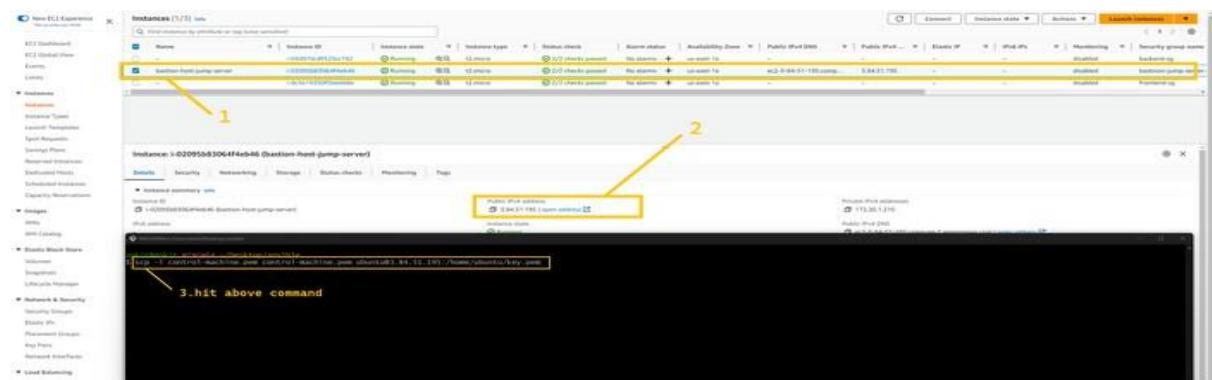
In the network setting select VPC that we have created and in the subnet select pub- sub-1a, you can select any public subnet from the VPC. and then select security group. We already have a security group with the name bastion-jump-server-sg and click on the launch instance.



Once the instance becomes healthy, we can SSH into it. so select the instance and copy its public IP. Open Git bash or terminal in which folder your key.pem file is present and hit the below command.

`scp -i <name_of_your_key>.pem`

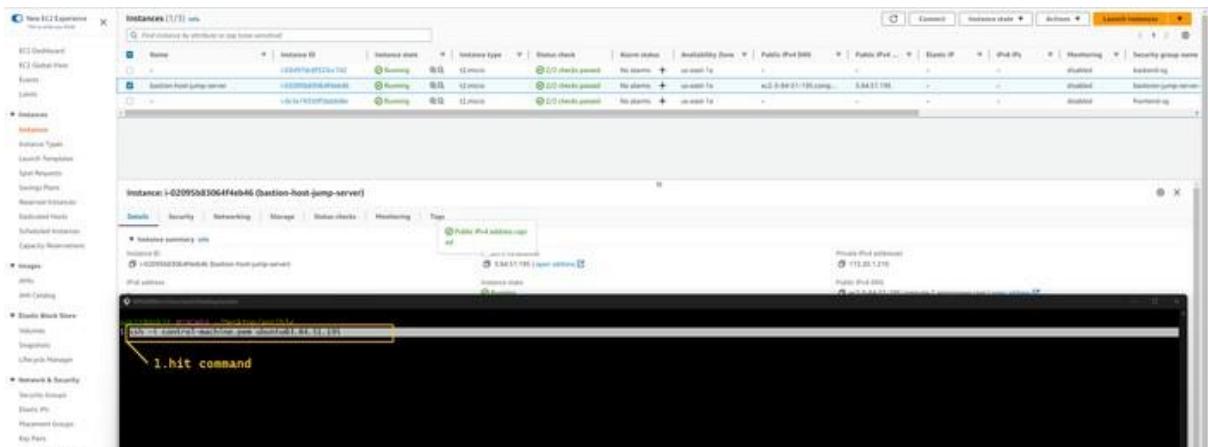
```
<name_of_your_key>.pem
ubuntu@<Public IP add of instance>:/home/ubuntu
/key.pem
```



The above command will copy our login secret key file into the bastion host.

Now type the below command to login into the Bastion host. And copy the public IP of the Bastion host.

```
ssh -i <name_of_your_key>.pem ubuntu@<Public IP add of instance>
```



We are successfully logged in inside the bastion host.



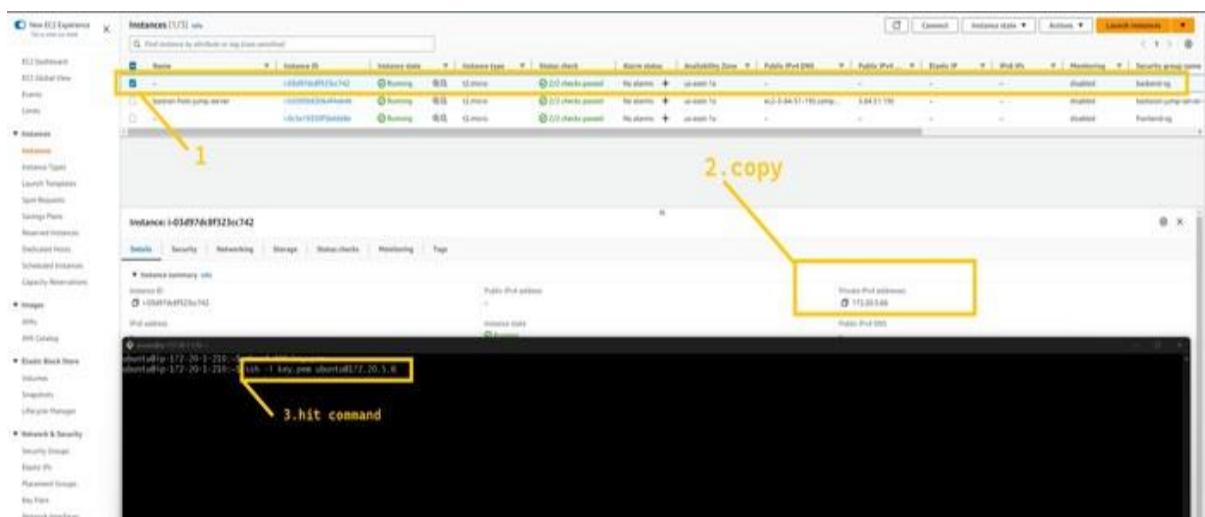
Hit the below command to change the permission of the key.pem file

```
chmod 400 key.pem
```



Now we want login into the backend server so select backend server and copy its private IP address. You can identify the backend server by the security group attached to the instance. Type the below command to log in to the backend server.

```
ssh -i key.pem ubuntu@<Private_IP_add_backend_server>
```



Cd Highly-Available-AWS-Multi-Region-3-Tier-Architecture/ backend



We need to install one package type below the command

```
sudo apt install mysql-server -y
```



And type the below command to initialize the database.

```
mysql -h book.rds.com -u <user_name_of_rds> -p<password_of_rds> test < test.sql
```

connect to your database by using

```
mysql -h book.rds.com -u <user_name_of_rds> -  
p<password_of_rds mysql -h book.rds.com -u admin -  
psrivardhan -----this is my url
```

```
create  
databas  
e test;  
exit
```

again give first command in project backend directory.

```
mysql -h book.rds.com -u <user_name_of_rds> -p<password_of_rds> test < test.sql
```



Now you can close the terminal.

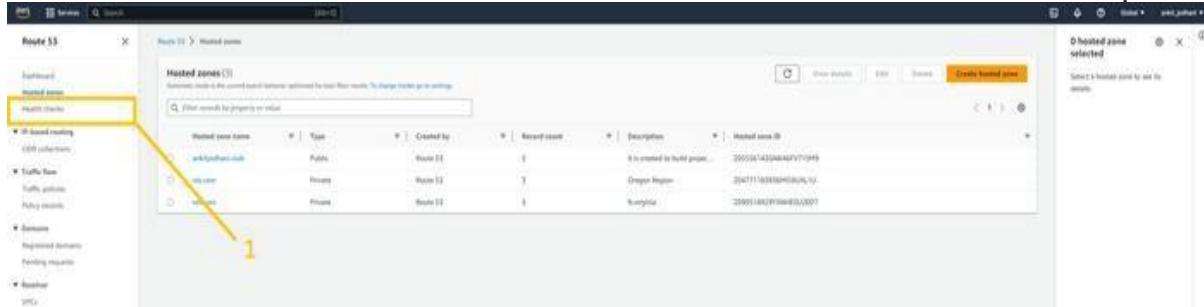
◆ Route 53

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service.

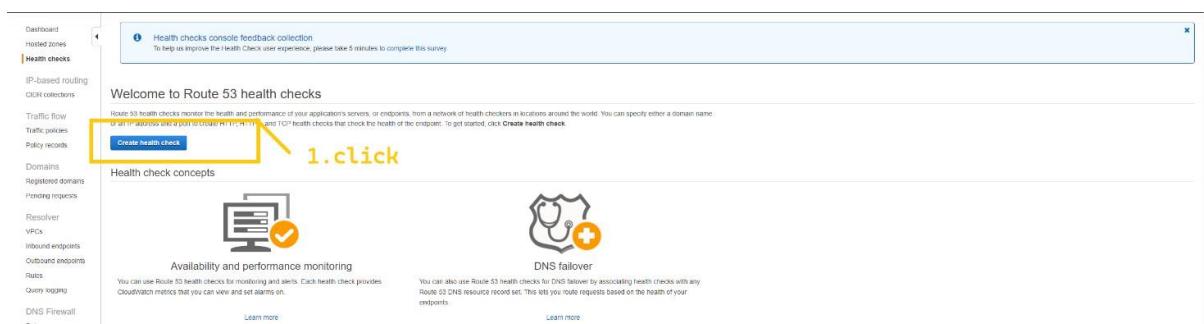
If you try to access the web app using ALB-frontend DNS then you won't see the website in functional mode because our frontend or loaded static pages try to call the API from your browser on the domain name https://api.<Your_Domain_name>.xyz. In my case, <https://api.vardhan.live>. And that record we didn't add yet in our domain name. so let's do that.

We are going to set up a Health check in route 53. So route 53 checks the health of the backend servers and if it is unhealthy (hits by disaster) then it will transfer the traffic to another region's (**disaster recovery region, Oregon**) backend server.

Head over to route 53 service. And click on the health check button on the left panel.



Click on the create health check button.



On this page, we can configure our Health check. Give a name to your health check, and select the endpoint to monitor it. select HTTP and in the Domain name field give the DNS of the ALB-backend which is in US-EAST-1 because **us-east-1 is our primary region**. And fill in all the details as I have shown you in the below image. And then click on the next button.

Configure health check

Route 53 health checks let you track the health status of your resources, such as web servers or mail servers, and take action when an outage occurs.

Name: backend_health_check **What to monitor:** Endpoint

Protocol: HTTP **Domain name:** ALB-backend-1202377654.us-east-1.elb.amazonaws.com **Port:** 80 **Path:** /images

Request interval: Standard (30 seconds) **Failure threshold:** 3 **String matching:** No **Latency graphs:** Off **Invert health check status:** Off **Disable health check:** By default, disabled health checks are considered healthy. Learn more

Health checker regions: US East (N. Virginia), US West (Oregon), US West (N. California)

URL: http://ALB-backend-1202377654.us-east-1.elb.amazonaws.com:80/ **Health check type:** Basic + additional options: Fast Interval (View Pricing)

Next

Our health check is set up successfully. it takes a few minutes to show whether our specified resource is healthy or not.

Name	Status	Description	Alarms	ID
backend_health_check	Healthy	http://ALB-backend-1202377654.us-east-1.elb.amazonaws.com:80/	No alarms configured.	c0701ec-ad3e-4d5e-a140-59ab073478e

In the next few minutes, it starts showing you the health of the ALB-backend (backend-servers).

Record name	Type	Value
arkitjodhani.club	NS	ns1.arkitjodhani.club

Now let's create a record in our domain name. click on the hosted zone and select your public hosted zone or your domain. I already have one. And click on the Create record button in the top right corner.

Hosted zones

Hosted zones

1

Records (5) **Create record**

Record name	Type	Value
arkitjodhani.club	NS	ns1.arkitjodhani.club
_Se5d133d1b6077e0f584e43604f67b.arkitjodhani.club	CNAME	arkitjodhani.club
share.arkitjodhani.club	Simple	-
www.share.arkitjodhani.club	A	Simple

2.click

Select failover record. And click on the next button.

Here in the record name field write api so that our record name becomes api.<Your_Domain_name>.xyz in my case, it is api.vardhan.live . in the record type field select “A” and then click on the define failover record button.

The screenshot shows the 'Create Record Set' wizard step 2 of 3. The 'Record name' field contains 'api'. The 'Record type' dropdown is set to 'A – Routes traffic to an IPv4 address and some AWS resources'. Under 'Route traffic to', 'Alias' is selected, and 'Alias to Application and Classic Load Balancer' is chosen. A search bar shows 'dualstack(ALB-backend-579315607.us-east-1.elb.amazonaws.com)'. Below it, 'dualstack(ALB-backend-579315607.us-east-1.elb.amazonaws.com' is highlighted. The 'Evaluate target health' dropdown is set to 'Yes'. The 'Health check ID' field contains 'a3d09aa1-6e2a-4233-b4b3-d4b3b7e69077'. The 'Record ID' field contains 'US West load balancer'. A note at the bottom says 'one failover record or each type.' with a dropdown set to 'Primary'.

Firstly Select Alias to application and classic Load balancer from the drop-down list, secondly, select us-east-1 as a region. And in the below drop-down list select DNS of the ALB-backend. As you know that **us-east-1 is our primary region** so select primary in failover type. And in the health check ID select the health check that we have created just now. And click on the Define failover record button. Follow the below image for more clarity.

This screenshot is identical to the one above, showing the configuration for the primary region (us-east-1). All fields and settings are the same, including the record name 'api', record type 'A', alias selection, health check ID 'a3d09aa1-6e2a-4233-b4b3-d4b3b7e69077', and failover type 'Primary'.

Click on create record button.

Now we need to set up one more failover record with the same domain name but for a secondary region. Firstly Select Alias to application and classic Load balancer from the drop-down list, secondly, select us-west-2 as the region. And in the below drop-down list select the DNS of the ALB-backend. As you know that **us-west-2 is our secondary region** so select secondary in failover type. **Make sure you don't select anything in health check ID.** And click on the Define failover record button. Follow the below image for more clarity. For second region

aws Services Search [Alt+S]

EC2 VPC S3 IAM Route 53 RDS Lambda CloudWatch Secrets Manager CloudFront Amazon Redshift Elastic Kubernetes Service CloudFormation Elastic Container R

Global ▾ vardhan ▾

Record name **Info**
ap1 .vardhan.live
Keep blank to create a record for the root domain.

Record type **Info**
A – Routes traffic to an IPv4 address and some AWS resources

Alias

Route traffic to **Info**
Alias to Application and Classic Load Balancer
US West (Oregon)
Q dualstack.Alb-backend-1061937887.us-west-2.elb.amazonaws.com X
Alias hosted zone ID: Z1H1FL5HABSF5

Routing policy **Info**
Failover
Failover record type
Choose Primary to route traffic to the specified resource by default or Secondary to route traffic to the specified resource when the primary resource is unavailable. You can create only one failover record of each type.
Secondary

don't select any keep empty for second region

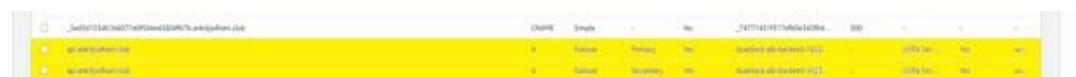
Health check ID - optional **Info**
Q Choose health check C

Evaluate target health
 Yes

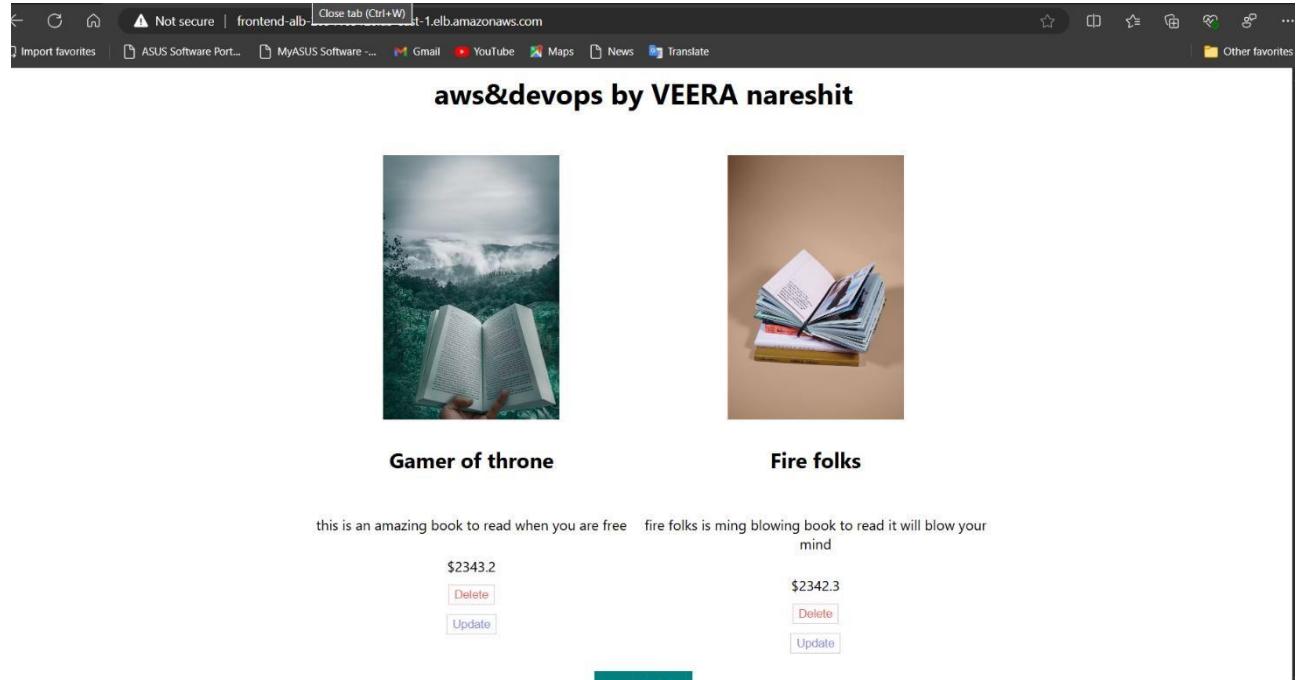
Record ID **Info**
US West load balancer

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

So we have two failover type records with same name but one is pointing to the backend load balancer which is in **us-east-1 region** and the second one is pointing to the backend load balancer of the **secondary region Oregon(us-west-2)**.

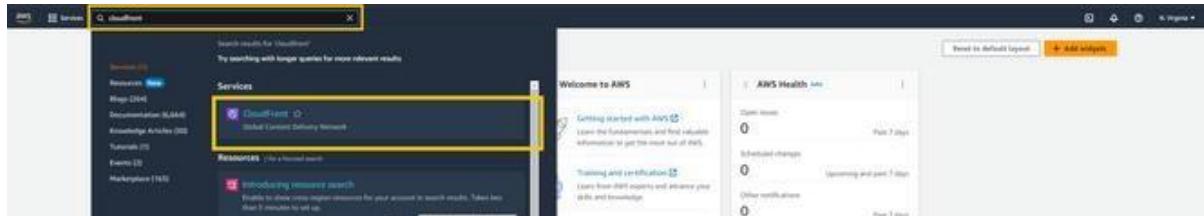


Now it's time to access our website so take the DNS of the **frontend load balancer** (ALB-frontend) and paste it into the browser. I am sure that you will see the website in fully functional mode. You can add and remove books.



◆ CloudFront

AWS CloudFront is a CDN service provided and fully managed by AWS. By utilizing CloudFront we can do the caching of our website at every edge location of the world. The user of the website faces less latency and gets high performance.
let's create Cloudfront distribution for our website. Head over to CloudFront.



Click on the distribution button on the left panel and then click on the create distribution button on top right corner.



In the origin name field select ALB-frontend (**us-east-1 primary region**). Select Match Viewer in the protocol field. And scroll down

The screenshot shows the "Create distribution" wizard. In the "Origin" step, the "Origin domain" field contains "ALB-frontend-919122761.us-east-1.elb.amazonaws.com". The "Protocol" dropdown has "Match viewer" selected. A large yellow arrow points downwards from the "Protocol" field towards the bottom of the page, indicating where to scroll.

Origin

Origin domain
Origin domain must be a public DNS name or IP address.
ALB-frontend-919122761.us-east-1.elb.amazonaws.com

Protocol Match viewer
 HTTP only
 HTTPS only

HTTP port
Enter your origin's HTTP port. The default is port 80.
80

HTTPS port
Enter your origin's HTTPS port. The default is port 443.
443

Minimum origin SSL protocol Info
The minimum SSL protocol that CloudFront uses with the origin.
 TLSv1.2
 TLSv1.1
 TLSv1
 SSLv3

Origin path - optional Info
Enter a URL suffix to append to the origin domain name for origin requests.
Enter the origin path
ALB-frontend-919122761.us-east-1.elb.amazonaws.com

Name
Enter a name for this origin.
ALB-frontend-919122761.us-east-1.elb.amazonaws.com

Add custom header - optional
CloudFront includes this header in all requests that it sends to your origin.

Disable Origin Shield Info
Origin Shield is an additional caching layer that can help reduce the load on your origin and help protect its availability.
 No
 Yes

► Additional settings

Default cache behavior

In viewer policy select Redirect **HTTP to HTTPS** and allow all the methods. But please make sure that you select Caching Disabled and in cache policy and select AllViewr in origin request policy.

Default cache behavior

Path pattern: **info**

Component objects automatically: **info**

Yes

Viewer

Viewer protocol policy:

- HTTP and HTTPS
- Redirect HTTP to HTTPS
- HTTPS only

Allowed HTTP methods:

- GET, HEAD
- GET, HEAD, OPTIONS
- GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Cache key and origin requests

We recommend using a Cache policy and origin request policy to control the cache key and origin requests.

Cache policy and origin request policy (recommended)

Legacy cache settings

Cache policy

CachingDisabled

Origin request policy - optional

AWS WAF

Create origin request policy View policy

Response headers policy - optional

Select response headers policy Create response headers policy

Additional settings

Click on the add item button and add an alternative domain name (frontend.ramakrishna.shop) and select the certificate that we have created in the Custom SSL certificate field.

Enable security protections

Keep your application secure from the most common web threats and security vulnerabilities using AWS WAF. Blocked requests are stopped before they reach your web servers.

Do not enable security protections

Select this option if your application does not need security protections from AWS WAF.

Settings

Price class [Info](#)

Choose the price class associated with the maximum price that you want to pay.

- Use all edge locations (best performance)
- Use only North America and Europe
- Use North America, Europe, Asia, Middle East, and Africa

Alternate domain name (CNAME) - optional

Add the custom domain names that you use in URLs for the files served by this distribution.

frontend.vardhan.live

[Remove](#)

[Add item](#)

To add a list of alternative domain names, use the [bulk editor](#).

Scroll down and click button create distribution.

Add the custom domain names that you use in URLs for the files served by this distribution.

RemoveAdd item

To add a list of alternative domain names, use the [bulk editor](#).

Custom SSL certificate - *optional*

Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).



*.vardhan.live Request certificate

Legacy clients support - \$600/month prorated charge applies. Most customers do not need this.

CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS.

Enabled

Security policy

The security policy determines the SSL or TLS protocol and the specific ciphers that CloudFront uses for HTTPS connections with viewers (clients).

- TLSv1.2_2021 (recommended)
- TLSv1.2_2019
- TLSv1.2_2018
- TLSv1.1_2016
- TLSv1_2016
- TLSv1

Supported HTTP versions

Add support for additional HTTP versions. HTTP/1.0 and HTTP/1.1 are supported by default.

- HTTP/2
- HTTP/3

Default root object - *optional*

The object (file name) to return when a viewer requests the root URL (/) instead of a specific object.

Standard logging

Get logs of viewer requests delivered to an Amazon S3 bucket.

- Off
- On

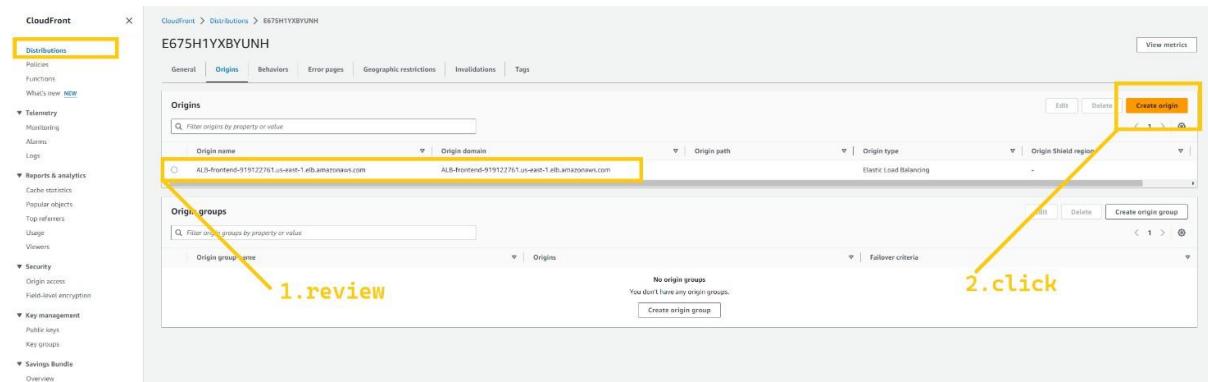
IPv6

- Off
- On

Description - *optional*

CancelCreate distribution

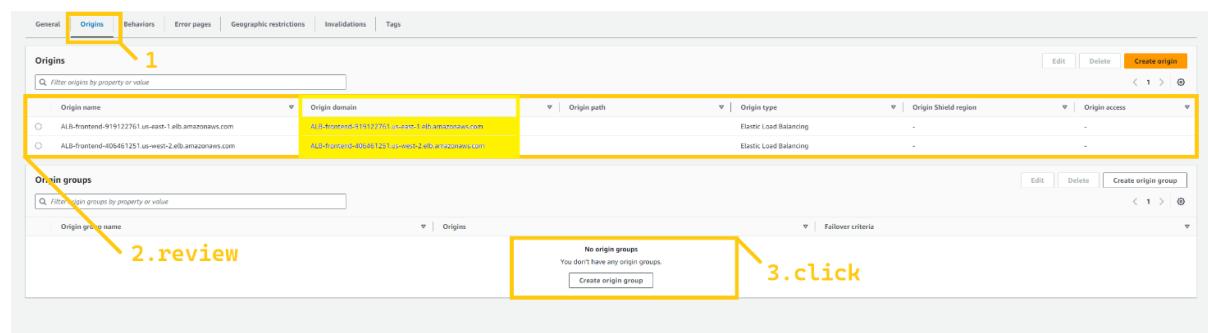
Now, click on the distribution that we have created just now and click on the Origin tab. Here you need to select create origin the button in the top right corner.



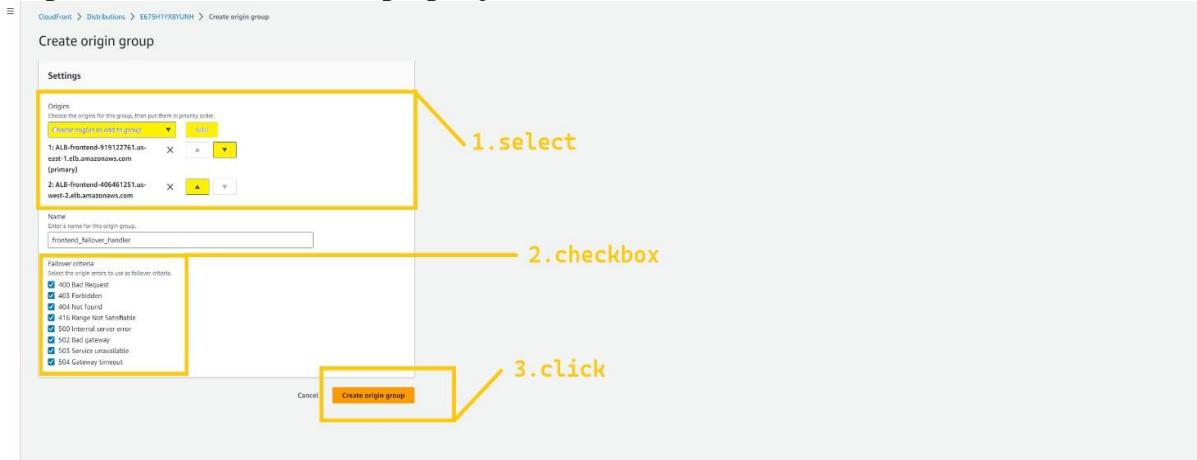
Click on the origin domain field and select the ALB-frontend (**us-west-2 secondary region**), select math view in protocol and the rest of the parameters are all the same so click on the create origin button.



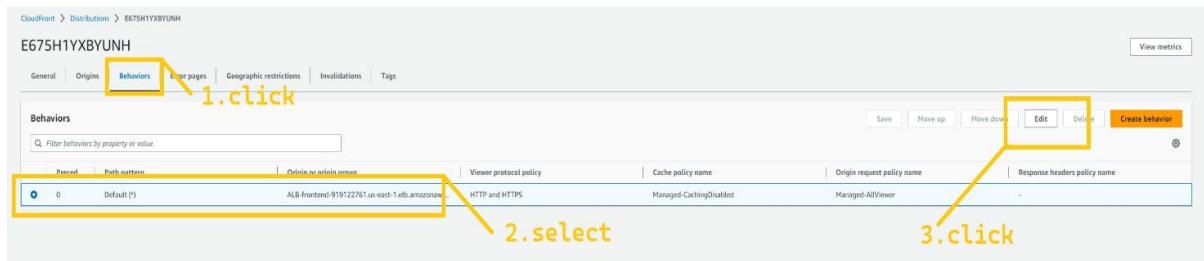
So now we have two Origins one is points to ALB-frontend which is in us-east-1 and the second one is pointing to ALB-frontend which is in the secondary region Oregon (us- west-2). Now click on the create origin group button.



Here, In the Origins field select the first origin that is associated with us-east-1 and click on the add button. And again click on the origin field and select the origin that is associated with us-west-2 and click on the add button. Give any name to the origin group (frontend_failover_handler) and select all the failover criteria as I have shown in the below image. Hit the button created origin group.



Now, click on the behavior tab. And select the behavior and click on the edit button.



Here we need to change the origin and origin group. Select the origin group that we have just created (frontend_failover_handler). Scroll down and click on the save button.



We need to wait till the distribution become available. It takes around 5-8 minutes. And then we can access our website through the DNS name generated by CloudFront. But we want to access the web app custom domain name. so again head over to Route 53 and select your public hosted zone. or your domain name hosted zone. and click on create record button.

The screenshot shows the AWS Route 53 console. The left sidebar is collapsed, showing navigation links like 'Route 53', 'Dashboard', 'Hosted zones', 'Health checks', 'Profiles', 'IP-based routing', 'Traffic flow', 'Domains', 'Resolver', 'VPCs', and 'Inbound endpoints'. The main area displays the 'vardhan.live' hosted zone details. At the top right are buttons for 'Delete zone', 'Test record', and 'Configure query logging'. Below that is a 'Hosted zone details' section with an 'Edit hosted zone' button. Underneath is a table titled 'Records (6)' with columns for Record, Type, Routine, Differ., Alias, Value/Route traffic to, TTL, and Health. The table lists six records: one NS record for 'vardhan.live' pointing to four nameservers (ns-1560.awsdns-03.co.uk, ns-1189.awsdns-20.org, ns-858.awsdns-45.net, ns-68.awsdns-08.com) with a TTL of 172800, and two SOA records for 'vardhan.live' with a TTL of 900.

Record	Type	Routine	Differ.	Alias	Value/Route traffic to	TTL	Health
vardhan.live	NS	Simple	-	No	ns-1560.awsdns-03.co.uk. ns-1189.awsdns-20.org. ns-858.awsdns-45.net. ns-68.awsdns-08.com.	172800	-
vardhan.live	SOA	Simple	-	No	ns-1560.awsdns-03.co.uk. a...	900	-

Select simple record, and click on the button defined record. In the record name, add name threetier so our domain name becomes threetier.<Your_Domain_name>.XYZ, in my case, it is threetier.ankitjodhani.club. Select record type “A”. Select Alias to CloudFront distribution from the drop-down list in value/route traffic to field. And select the distribution that we have created just now. Lastly, hit the define simple record button. Route 53 takes sometime around 5-10 minutes to route traffic on the newly created record so please wait.

Screenshot of the AWS CloudFront 'Create Record' wizard.

Record name: frontend .vardhan.live

Record type: A – Routes traffic to an IPv4 address and some AWS resources

Alias: Alias

Route traffic to: Alias to CloudFront distribution

Region: US East (N. Virginia)

CloudFront distribution ID: Q_ d1bz4yek9va0e2.cloudfront.net

Routing policy: Simple routing

Evaluate target health: No

Buttons: Add another record, Create records

Now, let's check the final endpoint. Please hit the record name that you have set up. In my case that is <https://frontend.ramakrishna.shop>.

I am sure you can see the website in a running state.

We are almost done before we taste our application one small service but very essential service we want to utilize and that is WAF.

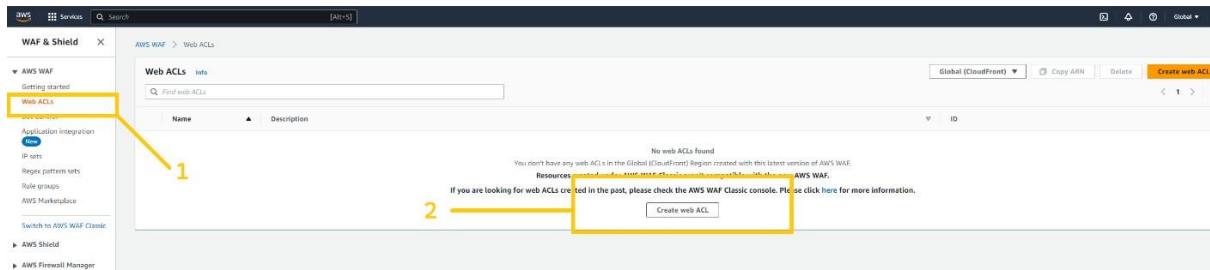
◆ AWS WAF (Web application firewall)

AWS WAF is a web application firewall that helps protect apps and APIs against bots and exploits that consume resources.

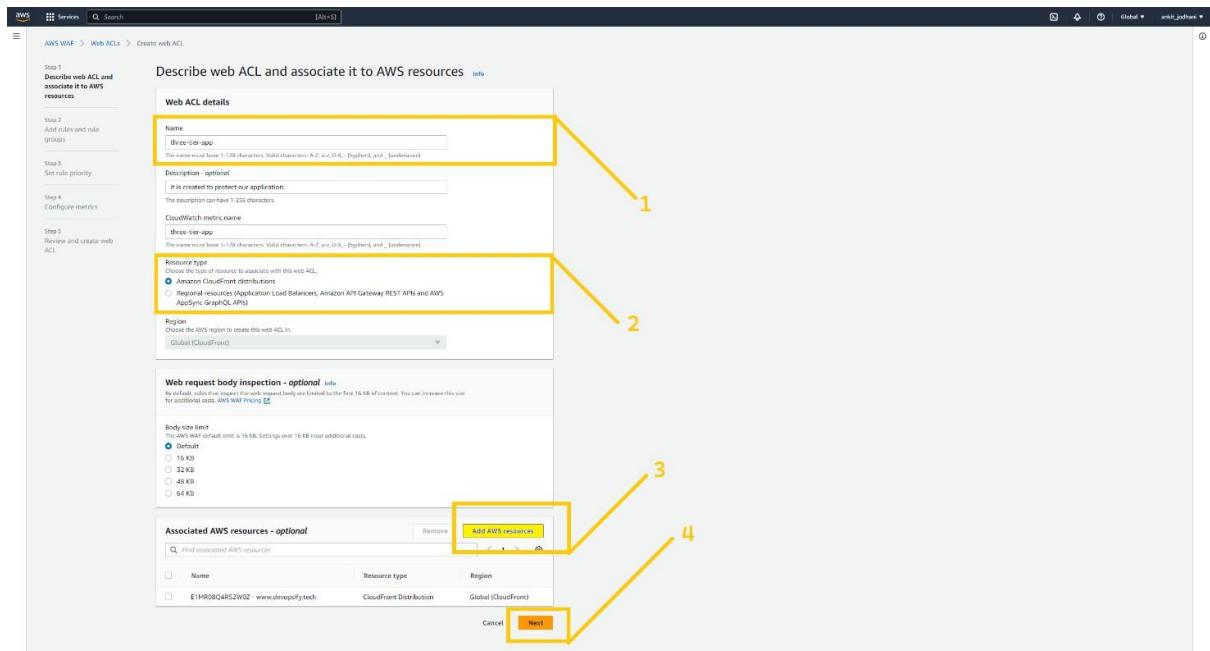
Search WAF in the AWS console, and click on the service.

A screenshot of the AWS Management Console search bar. The search input field contains the text "WAF". Below the search bar, a dropdown menu titled "Search results for 'WAF'" shows several service options: Services (11), WAF (1), AWS Lambda (1,000), Documentation (12,743), Knowledge Articles (98), Tutorials (9), AWS Firewall Manager (1), Control management of Network traffic, and AWS CloudWatch Metrics (1). The "WAF" option is highlighted with a yellow box.

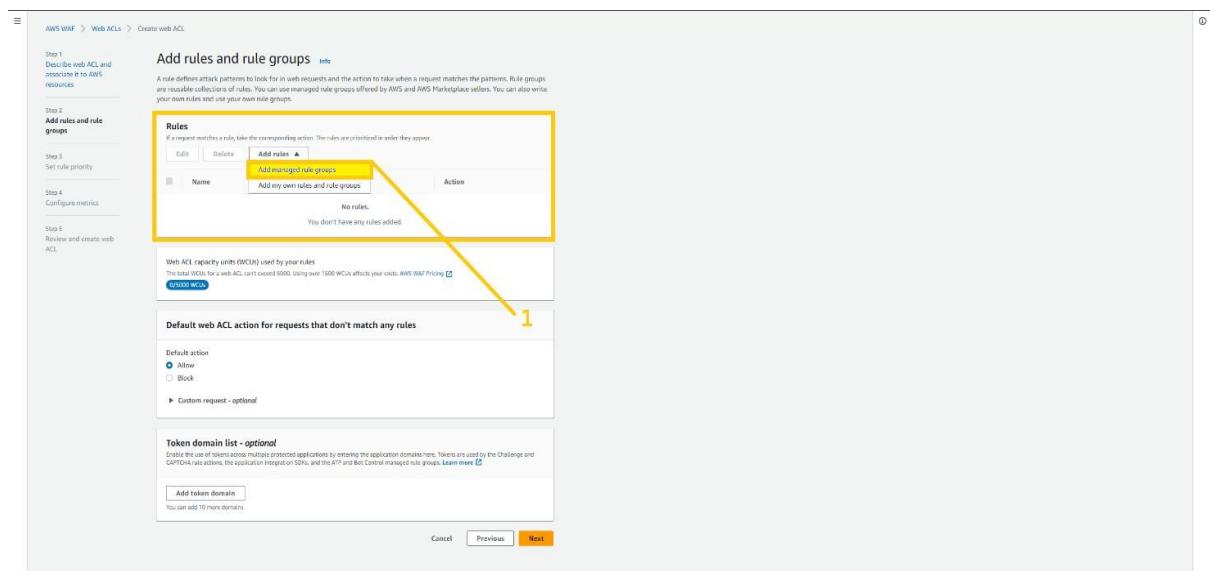
Click on the Web ACLs on the left panel and then click the button which is in the middle Create Web ACL.



Give some meaningful name to the ACL list, in resource type select the AWS CloudFront distribution. And then click on the Add AWS resource button and add the CloudFront distribution that we have just created.



Here, hit the add rule button on top and click Add manage rule group.



Over here we can add the rules to protect our web app front attackers. You can read the description and add the rules that suit your application security. Scroll down and save it.

The screenshot shows the AWS WAF console under the 'Web ACLs' section. It displays two main sections: 'AWS managed rule groups' and 'Free rule groups'.

- AWS managed rule groups:**
 - Paid rule groups:** Contains a single entry: 'Account creation fraud prevention - new'. It provides protection against the creation of fraudulent accounts on your site. It costs \$10 per month (prorated hourly) and uses a tiered fee model for requests analyzed by AWS WAF Pricing. An 'Add to web ACL' button is available.
- Free rule groups:** Contains several pre-defined rule groups:
 - Admin protection:** Contains rules that allow your site to block external access to specific admin pages. Capacity: 100. Action: Add to web ACL.
 - Amazon IP reputation list:** Contains rules that are based on Amazon threat intelligence. Capacity: 25. Action: Add to web ACL.
 - Anonymous IP list:** Contains rules that allow you to block requests from services that allow obfuscation of user identity. Capacity: 50. Action: Add to web ACL.
 - Cross rule set:** Contains rules that are generally applicable to web applications. Capacity: 700. Action: Add to web ACL.
 - Known bad inputs:** Contains rules that allow your site to detect and remove malicious inputs. Capacity: 200. Action: Add to web ACL.
 - Linux operating system:** Contains rules that block request patterns associated with exploiting vulnerabilities specific to Linux, including TTY attacks. Capacity: 200. Action: Add to web ACL.
 - PHP application:** Contains rules that block request patterns associated with exploiting vulnerabilities specific to PHP. Capacity: 100. Action: Add to web ACL.
 - POSIX operating system:** Contains rules that block request patterns associated with exploiting vulnerabilities specific to POSIX/UNIX-like OS, excluding Linux. Capacity: 100. Action: Add to web ACL.
 - SQL database:** Contains rules that allow you to block request patterns associated with exploitation of SQL databases, such as injection attacks. Capacity: 200. Action: Add to web ACL.
 - Windows operating system:** Contains rules that block request patterns associated with exploiting vulnerabilities specific to Windows, e.g., PowerShell cmdlets. Capacity: 200. Action: Add to web ACL.
 - WordPress application:** The WordPress application group contains rules that block request patterns associated with the exploitation of vulnerabilities specific to WordPress sites. Capacity: 100. Action: Add to web ACL.

Large yellow arrows point downwards from the 'Paid rule groups' and 'Free rule groups' sections towards the bottom of the page, indicating where to click to proceed.

Select default action Allow and hit the next button and that's it. we secured web application. You can see Web ACLs in the list.

The screenshot shows the AWS WAF console under the 'Web ACLs' section. A single Web ACL named 'three-tier' is listed.

Name	Description	ID
three-tier	It is created to protect our application	1db38360-e9f1-42e3-a705-4abfbfb06000

Testing

It's time to test our architecture. Let's see if it works as we expected. **Can we call it Resilient architecture? Did we implement the Warm Standby strategy properly?**

We are going to do manual failover by changing the rules of the security group of the ALB-frontend and ALB-backend. To make our frontend server and backend server inaccessible from the internet in **US-EAST-1 region**. So we can create a situation like a disaster.

First let's do a modification in ALB-frontend-sg.

1. Inbound rules section of SG-ALB-frontend-sg details page.

2. Inbound rules section of sg-0efaffb6acfe04 - ALB-frontend-sg details page.

3. remove rules

4. Edit inbound rules

Select ALB-frontend-sg. Click on the edit inbound rule. And remove all the HTTP and HTTPS rules from it. after doing this our CloudFront distribution won't be able to access this **ALB-frontend** and it have to route traffic to another region (**us-west-2**) ALB-frontend.

1. Inbound rules section of SG-ALB-backend-sg details page.

2. Inbound rules section of sg-0efaffb6acfe04 - ALB-backend-sg details page.

3. Removed

4. Edit inbound rules

let's do a modification in ALB-backend-sg.

1. Inbound rules section of SG-ALB-frontend-sg details page.

2. Inbound rules section of sg-0bb11c180dd785c7 - ALB-frontend-sg details page.

3. remove rules

4. Edit inbound rules

Select ALB-frontend-sg. Click on the edit inbound rule. And remove all the HTTP and HTTPS rules from it. after doing this route 53 will find it unhealthy and it have to route traffic to another region (**us-west-2**) ALB-frontend.

1. Inbound rules section of SG-ALB-backend-sg details page.

2. Inbound rules section of sg-0bb11c180dd785c9 - ALB-frontend-sg details page.

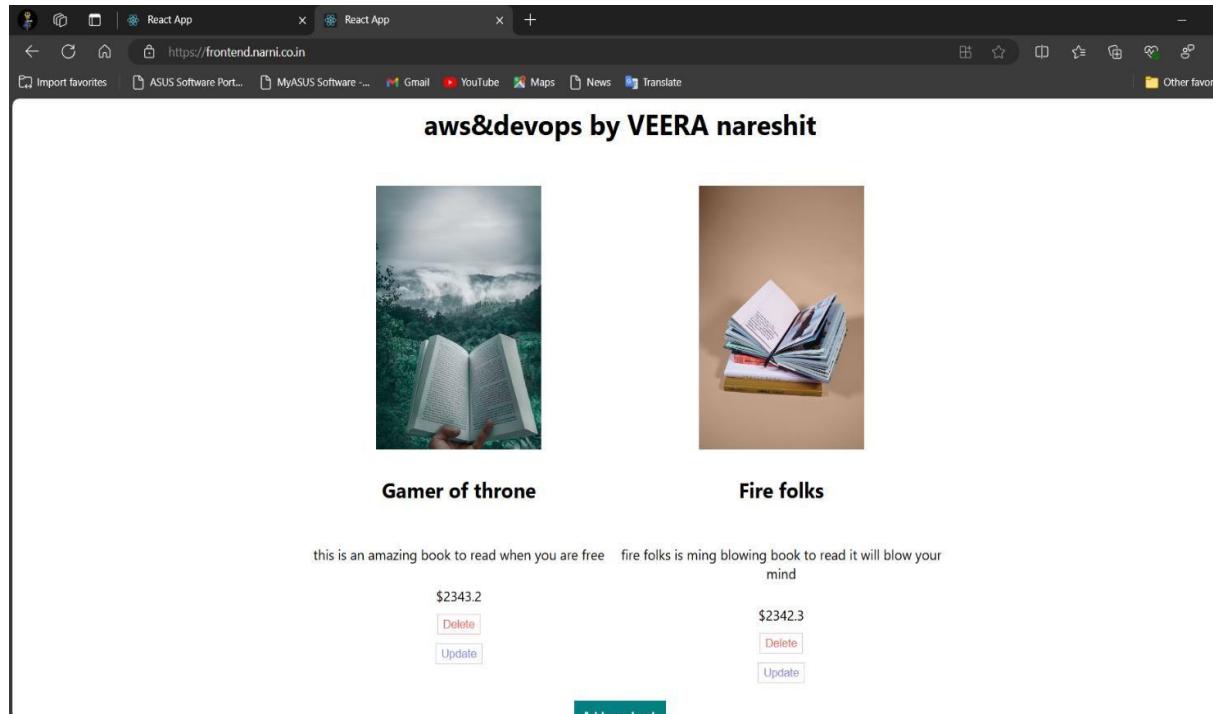
3. Removed

4. Edit inbound rules

So let's see if CloudFront and Route 53 are routing traffic to a secondary region's server. Take the domain name and try pasting it into the browser.

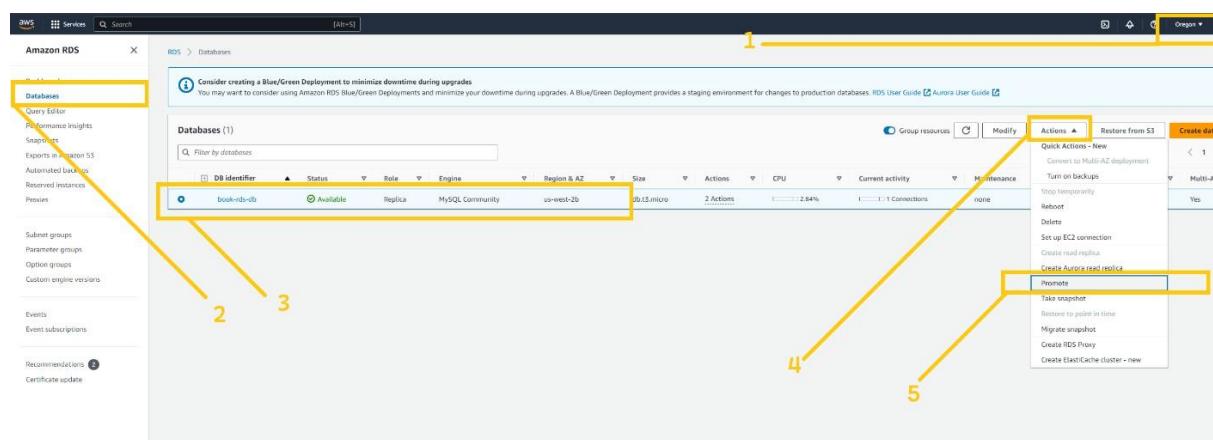
If we can see the website in fully functional mode, it means we have properly set up and configured our architecture.

Note: Please wait for at least 10 minutes so that Route 53 can identify the unhealthy resource. CloudFront takes 60 to 90 seconds for every request to route traffic. So, please don't drop the website immediately.



Yeahhh!! ☺ We Did it.

Our architecture working as we designed. But you can see that you can't add a book here. And it is because of read-replica. Read replica allows only read-only operation. We need to promote read-replica which is in the DR region(**us-west-2**). so that it becomes a database instance. And that allow read and write both operation.



I know it is taking a long time to open the website. But we can improve that by changing some configurations in CloudFront. Click on the origin tab. Select the first origin and click on the edit button.

The screenshot shows the AWS CloudFront Origins configuration page. It displays two origins: 'ALB-frontend-919122761.us-east-1.elb.amazonaws.com' and 'ALB-frontend-406461251.us-west-2.elb.amazonaws.com'. Below the origins is an 'Origin groups' section containing 'frontent_fallover_handler'. At the bottom, there are 'Edit', 'Delete', and 'Create origin group' buttons. Three yellow arrows point to the 'Edit' buttons in the first origin row (labeled 1), the 'Edit' button in the 'Origin groups' section (labeled 2), and the 'Edit' button in the 'Additional settings' section (labeled 3).

Click on the additional settings tab. And decrease the number. So that CloudFront won't wait too long for a response.

The screenshot shows the AWS CloudFront Additional Settings configuration page. It includes fields for 'HTTP port' (80), 'HTTP2 port' (443), 'Minimum origin SSL protocol' (TLS 1.3), 'Origin path - optional' (empty), 'Add custom header' (disabled), 'Enable Origin Shield' (disabled), and 'Additional settings' (Connection attempts: 3, Connection timeout: 3, Response timeout: 30, Keep-alive timeout: 5). Three yellow arrows point to the 'Connection attempts' field (labeled 1.write), the 'Keep-alive timeout' field (labeled 2.click), and the 'Save changes' button at the bottom right.

Now, if you try again you will feel very less latency compared to the previous one.

The screenshot shows a React application displaying two book covers. The left book, titled 'Gamer of throne', is described as 'an amazing book to read when you are free' and has a price of '\$2343.2'. The right book, titled 'Fire folks', is described as 'ming blowing book to read it will blow your mind' and has a price of '\$2342.3'. Both books have 'Delete' and 'Update' buttons below them.

Resource cleanup

◆ CloudFront

- disabled (takes a lot of time)
- delete

◆ RDS

- RDS instance or read replica delete in both regions (takes a lot of time)
- snapshot

◆ Route 53

- Delete both private hosted zone (rds.com)
- Delete all three records in the public hosted zone

◆ EC2

- Delete ASG in both regions
- Terminate Bastion host from us-east-1
- Delete ALB in both regions
- Delete TG from both regions
- Delete the Launch template from both regions
- Deregister AMIs which are created manually

◆ ACM

- Delete the certificate if you don't need it

◆ Backup services

- Delete resources from the backup vault in both regions
- Delete backup vault from N.virginia
- Delete Assign resources in the backup plan
- Delete backup plan

◆ VPC

- Delete NAT gateways from both regions (takes around 5 minutes)
- Release the Elastic IP
- Delete VPC in both regions (17 resources will be deleted on one click)

◆ WAF

- Remove Web ACLs from WAF

➤ Conclusion

In conclusion, the three-tier architecture implemented in our system ensures robustness and resiliency in the face of disasters or regional failures. With Route 53 and CloudFront handling DNS routing and failover mechanisms, our system can seamlessly switch to a secondary region, guaranteeing uninterrupted website availability. The use of Application Load Balancers and the RDS database further enhances scalability and reliability in processing user requests and storing data.

Ohh God 😅 And here it ends... 🎉🎉