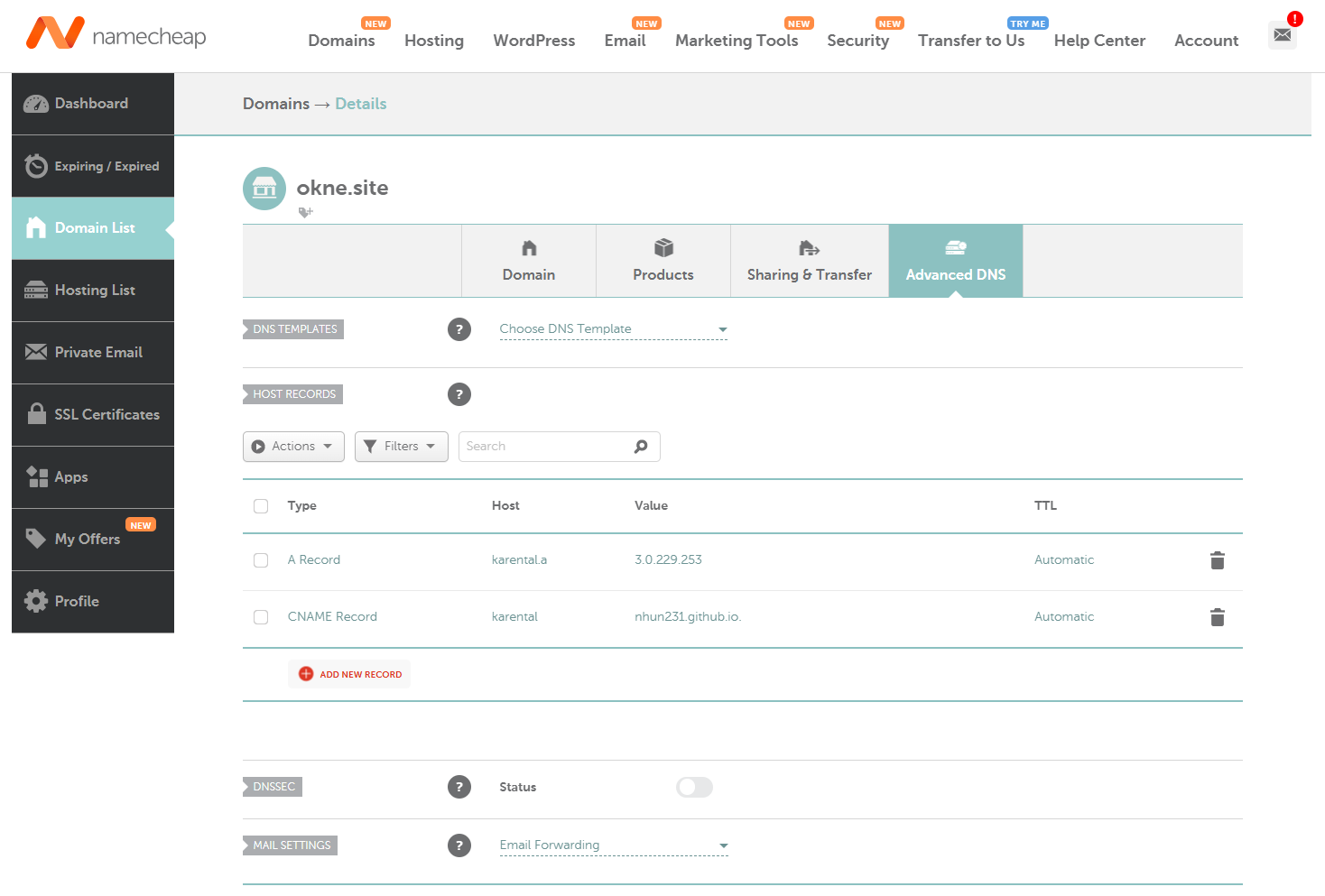
This document is simple guide which would guide you how to deploy the source code

# Domain Name

You will need a domain name, in here I would use namecheap to get a domain name and config it, you can choose whatever platform, the configuration is similar across them.

After you get a domain, go to the management dashboard of the domain, access Advance DNS session. Because we are going to deploy frontend and backend separately, we should config subdomain. You can come here to config it.

Remember to verify email, only after that, your domain could run.

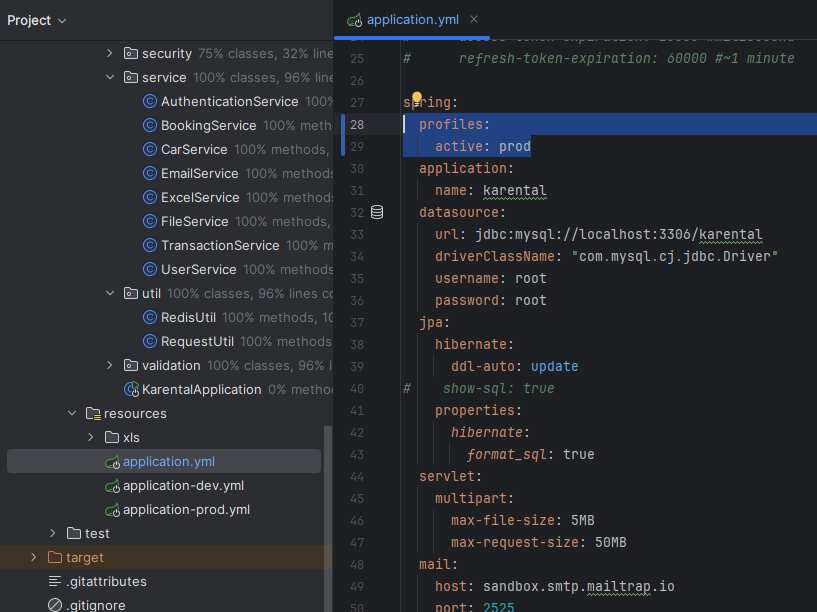
# Backend

## Techstack

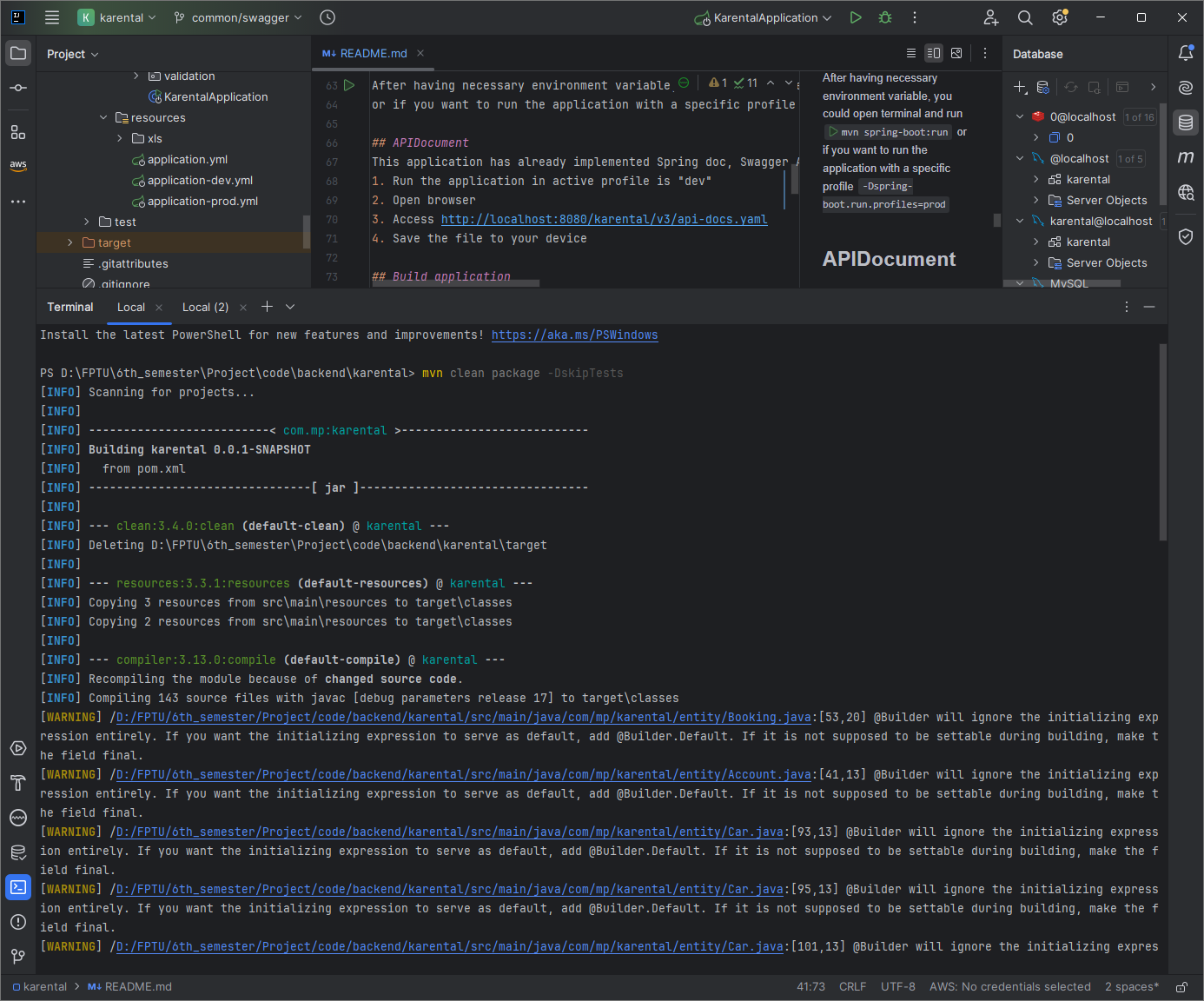
* Maven >= 3.9.0
* JDK17
* Spring boot 3.4.2
* MySQL 8.0.41
* Redis 7.4.2
* AWS S3
* Ngnix 1.27.4
* Docer engine, docker compose, docker cli (or just download Docker Desktop

## Build package

In this project, Maven is used to build package, by default, the active profile of this project is “prod”-production, if you don’t want it to run in prod profile, go to application.yml and change this

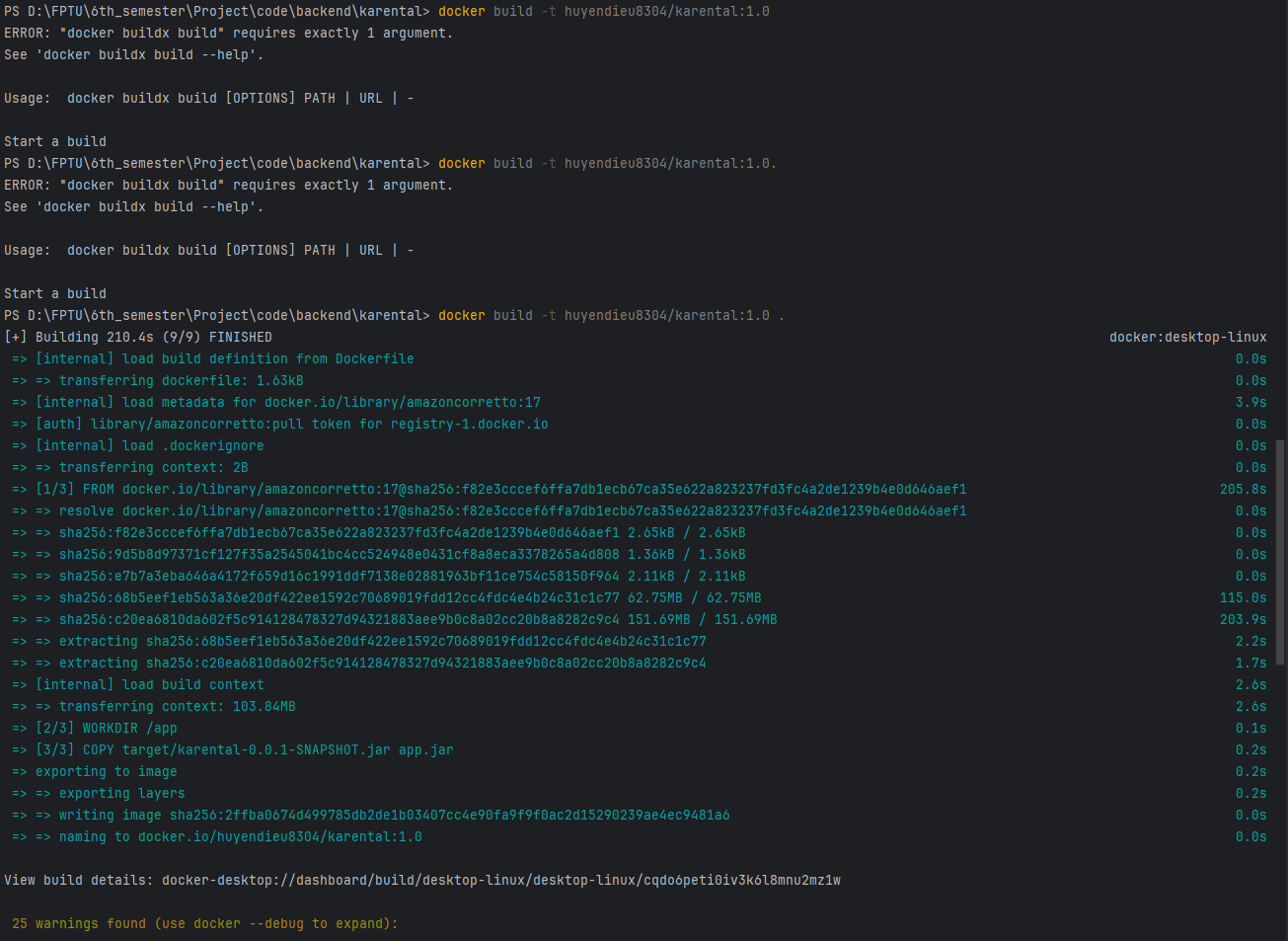


Open terminal in root directory of the project and run *`mvn clean package* -Dspring.profiles.active=prod *`*, maven would buil project in profile prod if the config above not be changed. In case you don’t want to run tests while building the project, run `*mvn clean package –DskipTests`.*

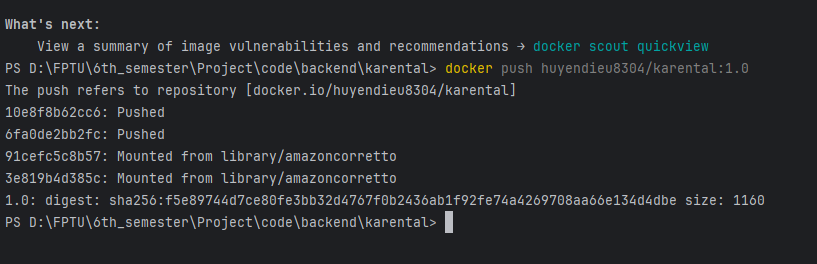


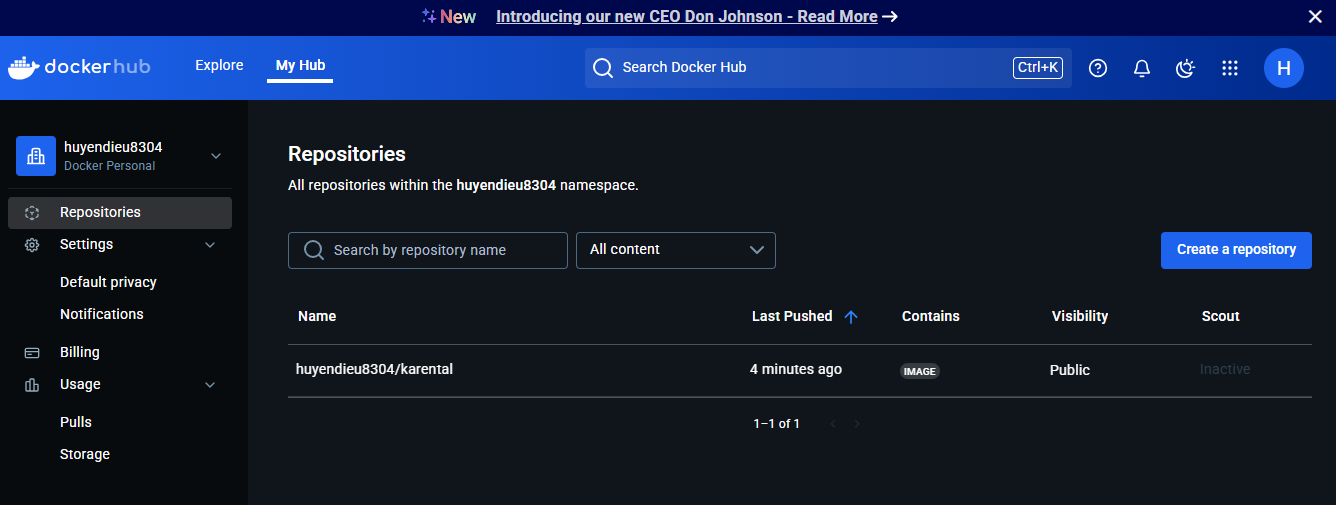
## Build docker image

Still in root directoy of the project, there is a Dockerfile, containing neccessary configuration to build an image, you just need to run *`docker build -t <your-dockerhub-username>/karental:<tagname> .`* to build the image of the application.



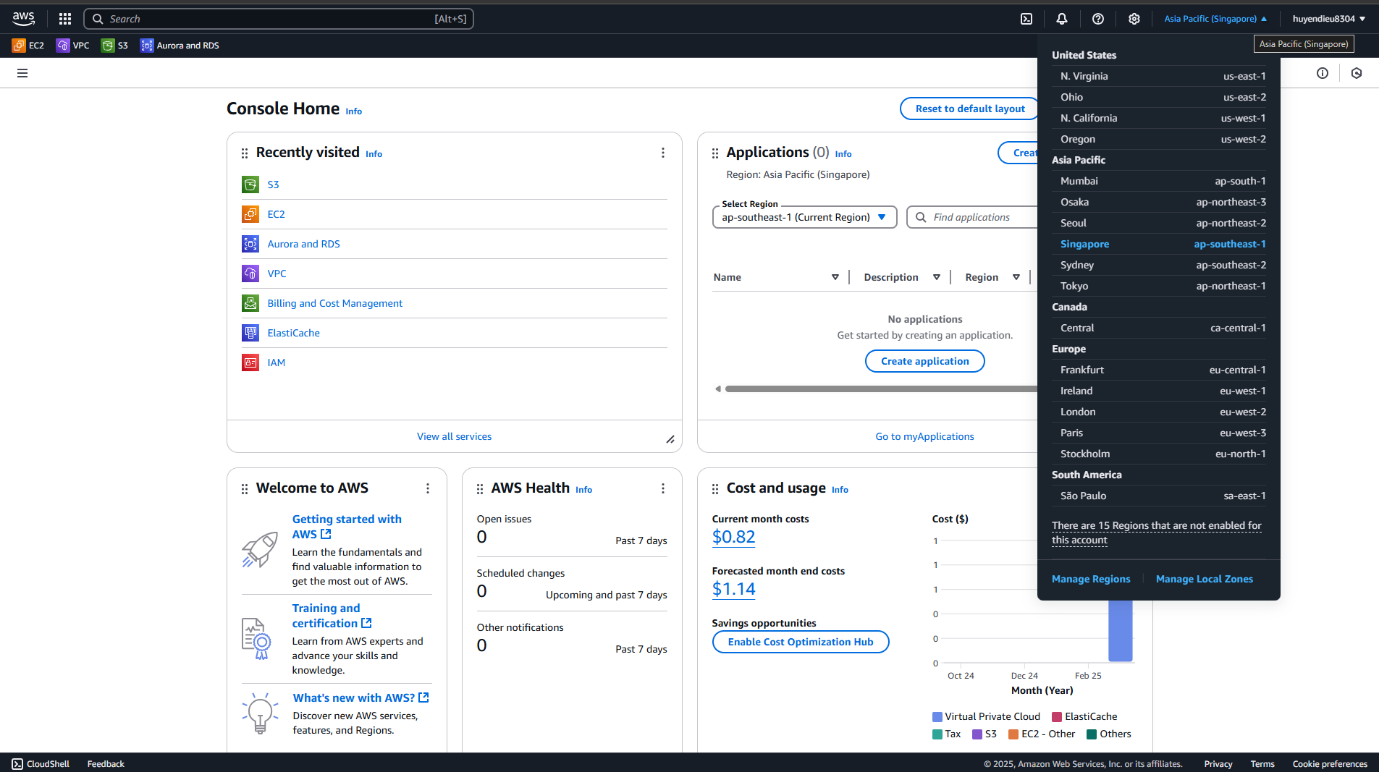
Next, you should push the built image to your image registry (in here we are using Docker Hub to store the image). Just run ` *docker push <your-dockerhub-username>/karental:<tagname>* `.

Open your docker hub repository and you would see a repo like this



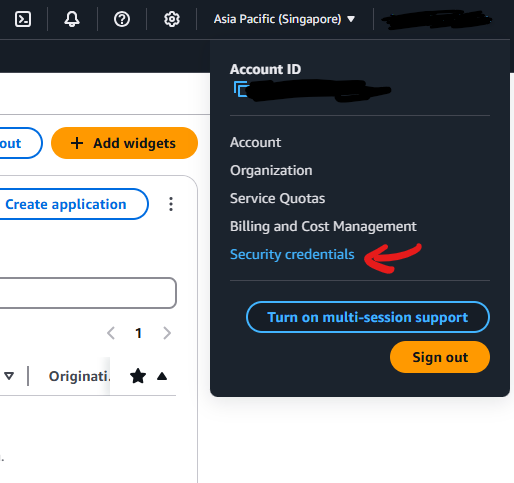
## Set up necessary environment to deploy

In this, we will use AWS to deploy our website. First, we need to have an AWS account and log in to it. In the Console Home, look to the top right, you would see next to your username is the location of the Amazon Region enabled to your account. Please choose a region suitable for you.

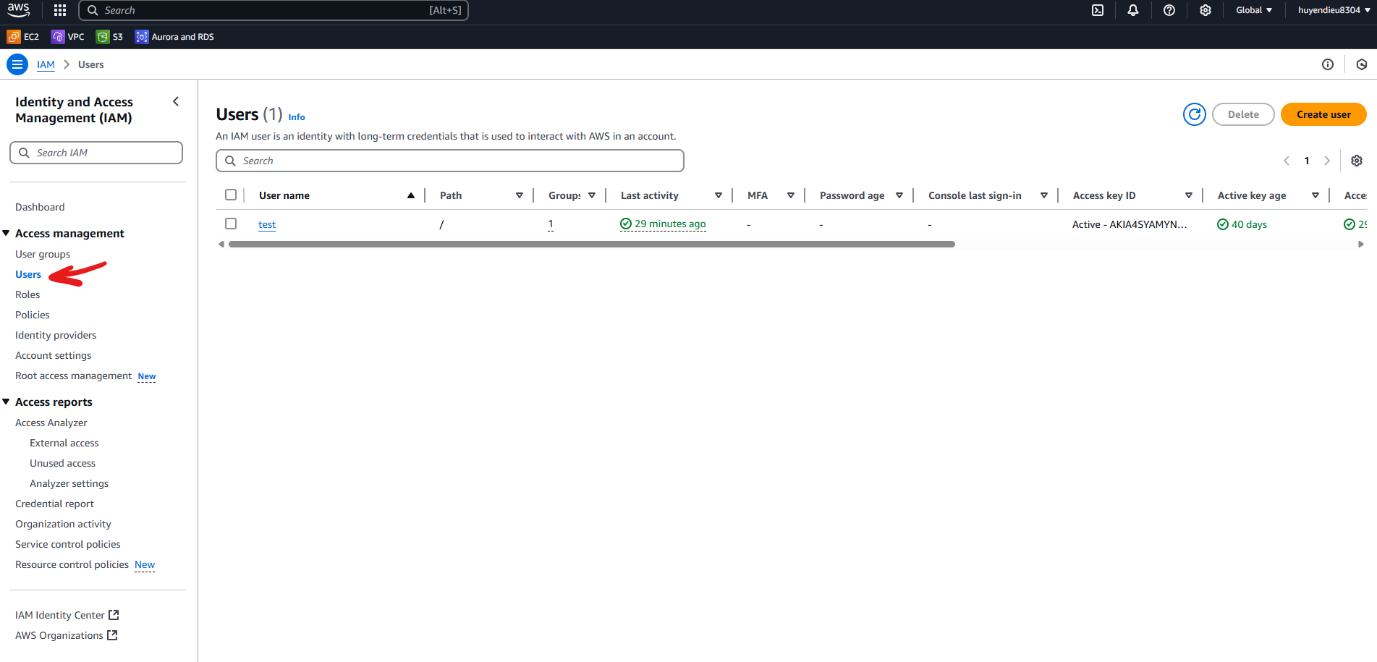


Before we really create “somthing” in here, we should need to know that to manipulate services in AWS using Java code, we have [AWS SDK for Java,](https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/home.html) and to work with our AWS services, we need to let our app to be authenticated. In here I’ll use IAM user, you can acess this [link](https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/credentials.html) to find out more about how to authenticate your access.

Click to your username in the top right corner, a menu appears, choose “Security credentials”.

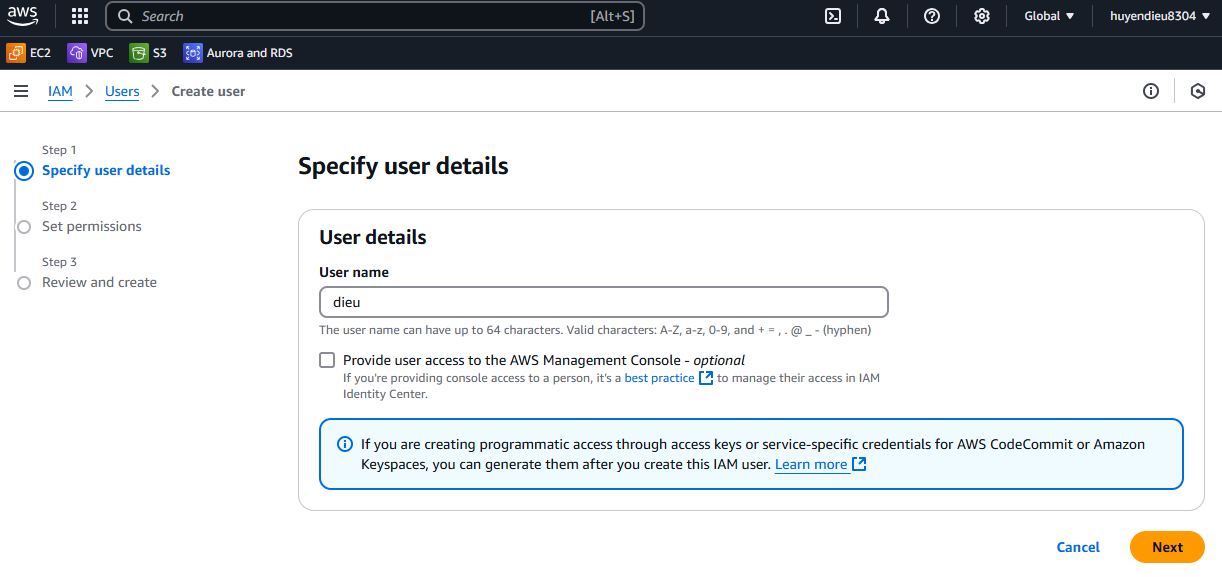


In the next page, choose Access Management/Users and you get a page like this one below.

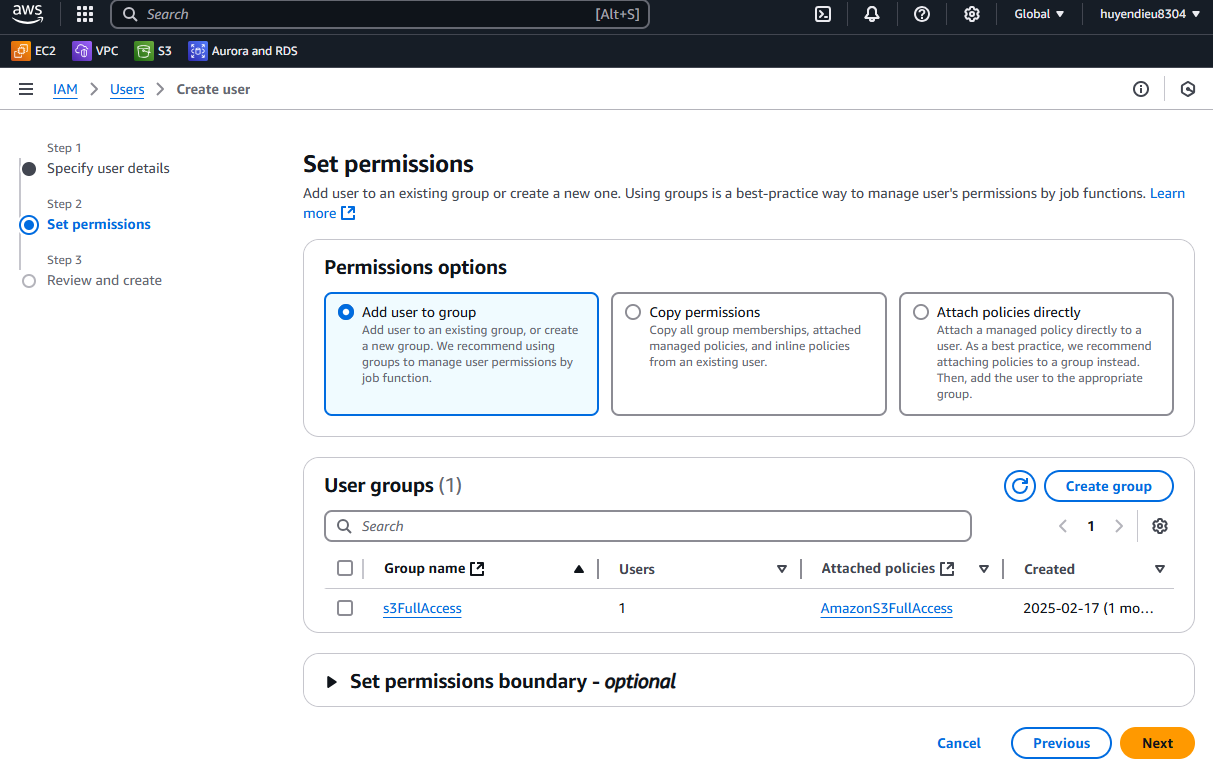


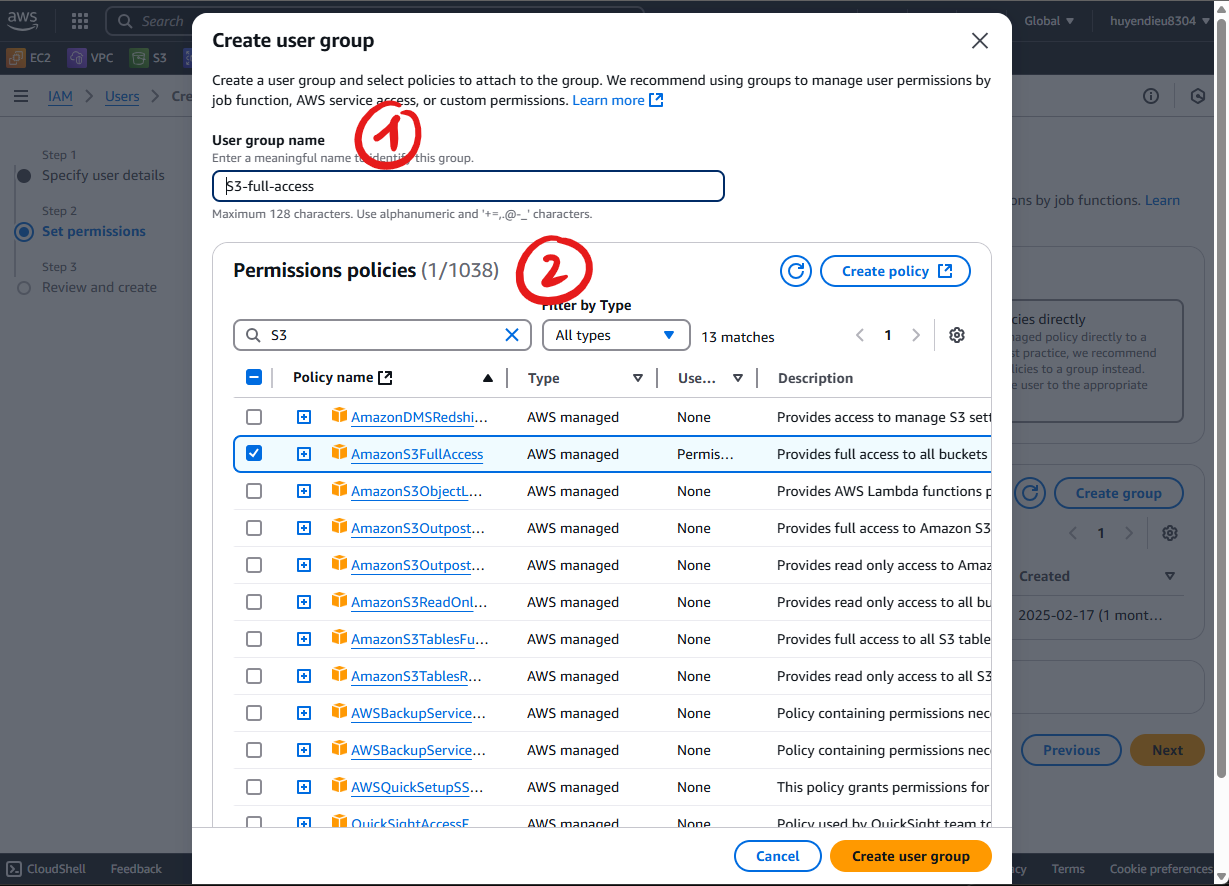
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account. We’re going to create a new user by clicking on the orange button “Create user”.

Step 1: choose your user name, then Next

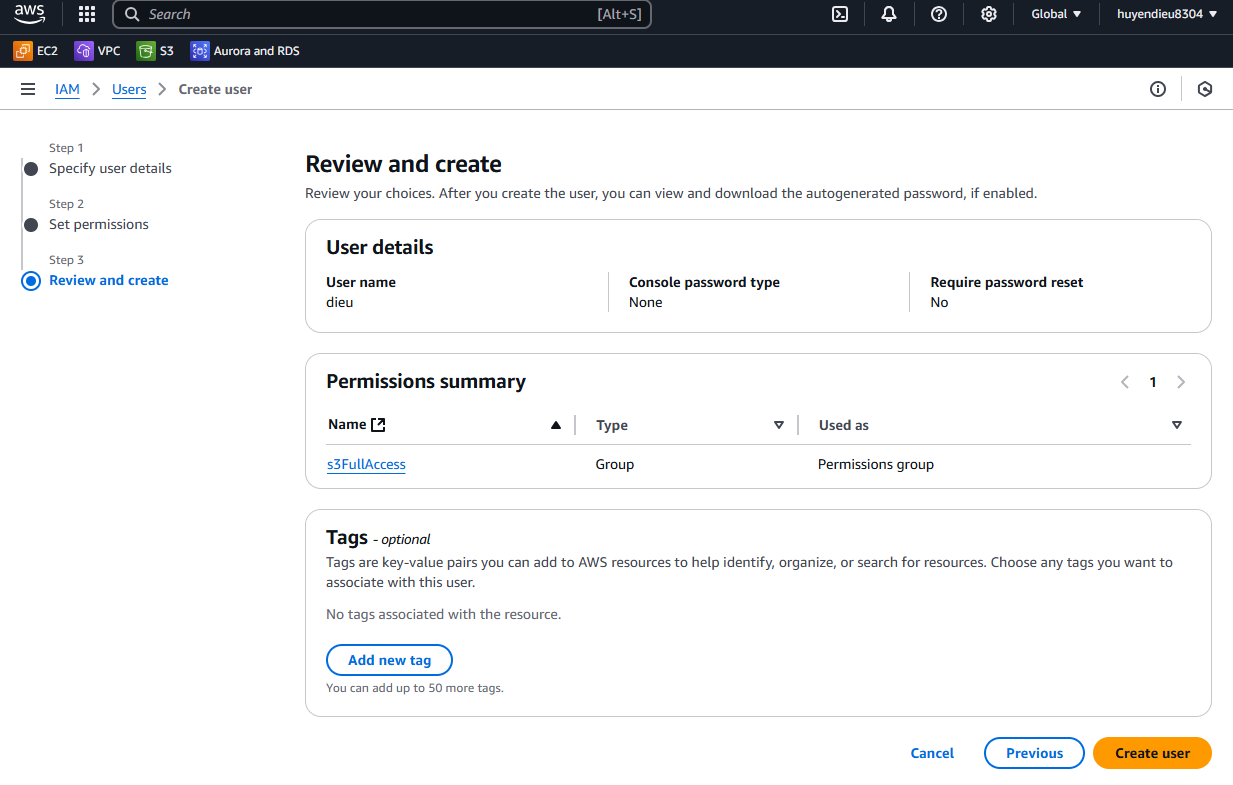


Step 2: Set permission for your user. I recommend you put the creating user to a specific user group. If you don’t have any group, click on “Create group” and choose appropriate policies or create new policy. To keep it simple, I would choose from existing 1038 policies.

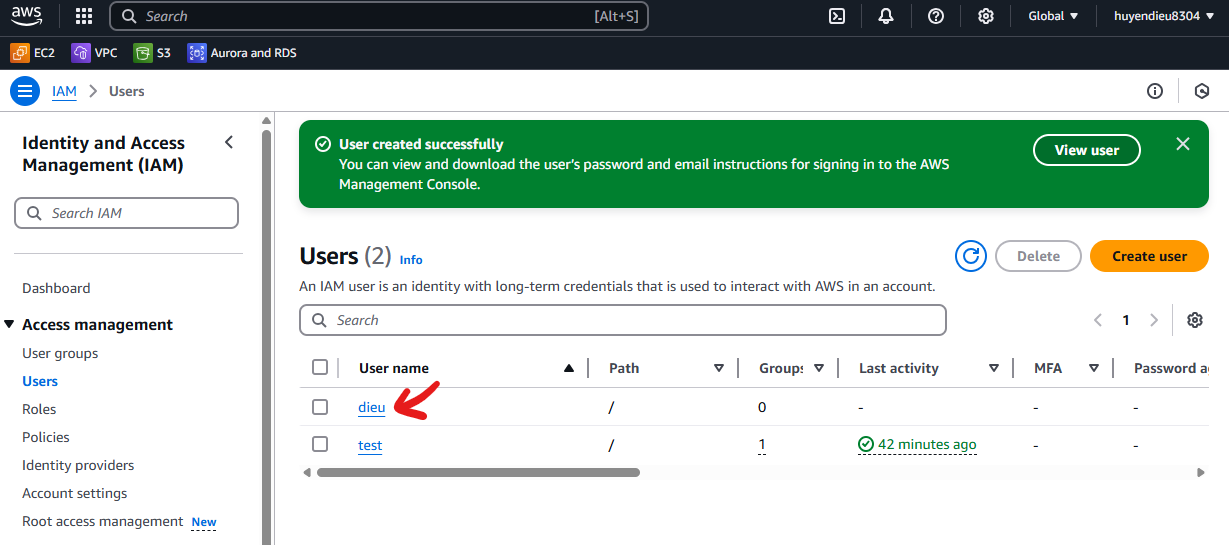




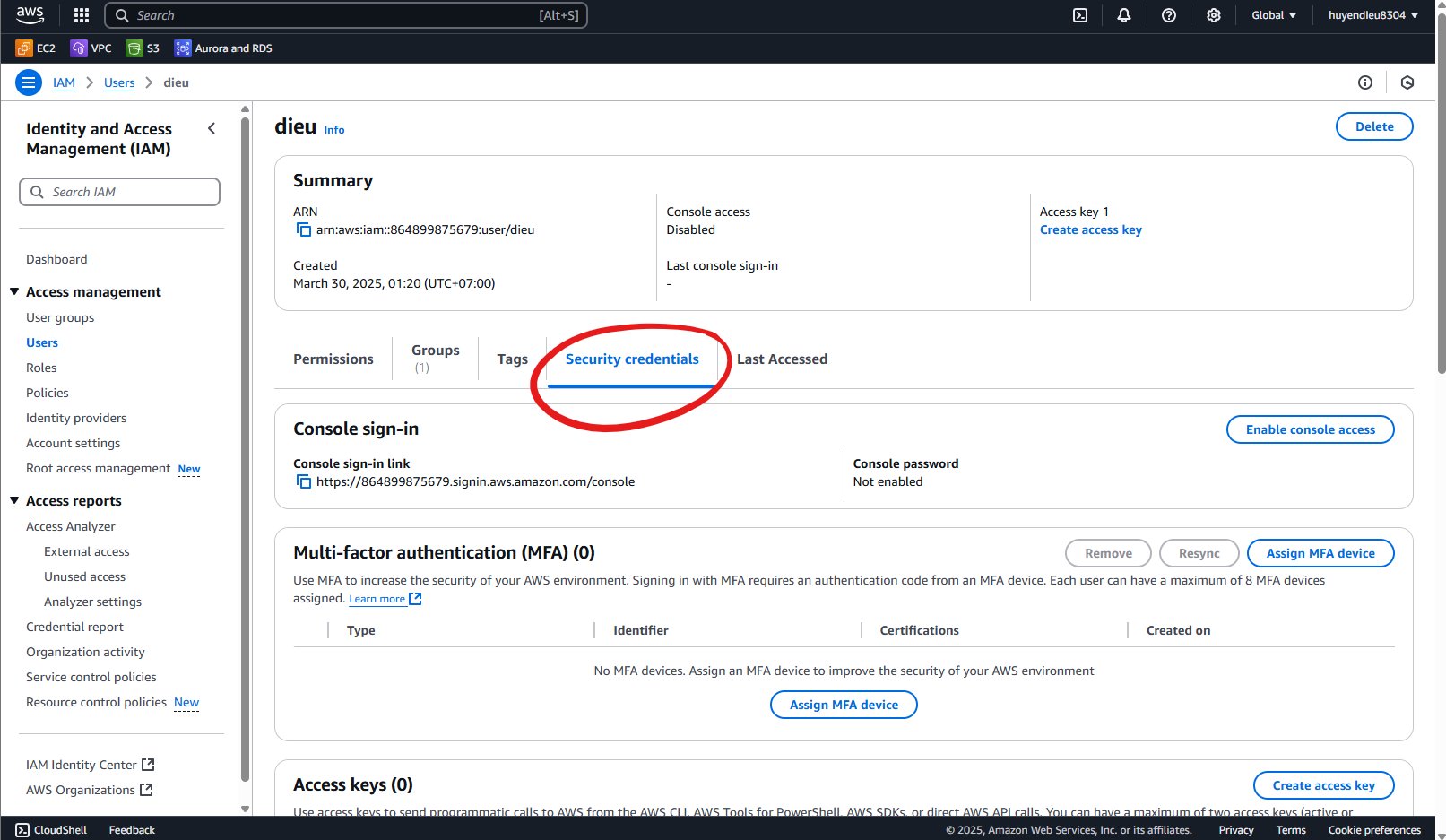
Step 3 just is review, click “Create user” to continue.



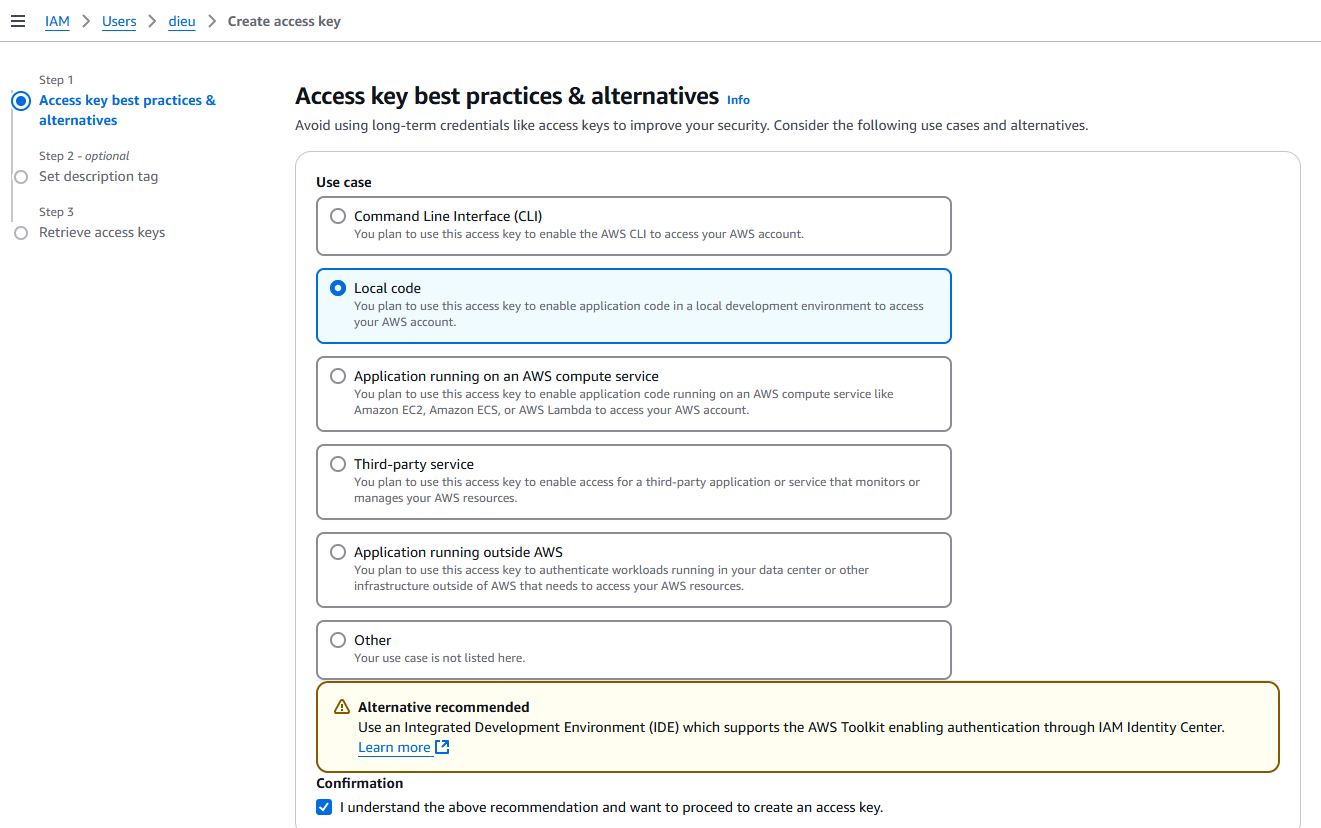
You would get a page like this, now let’s create credential for this user. Click to the user you just created.



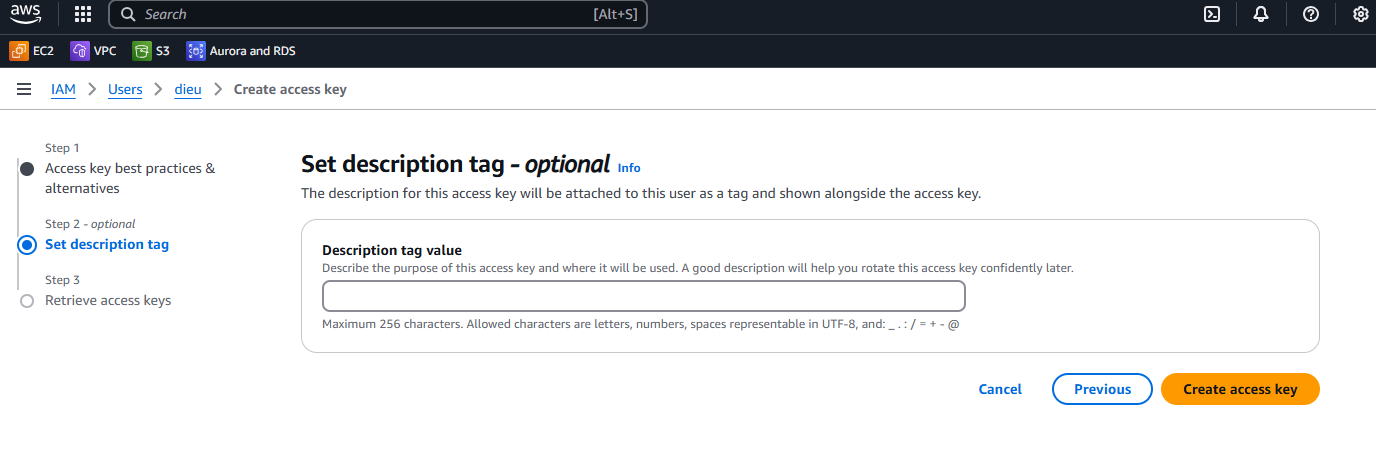
Go to “Security credentials”, scroll down and create access key



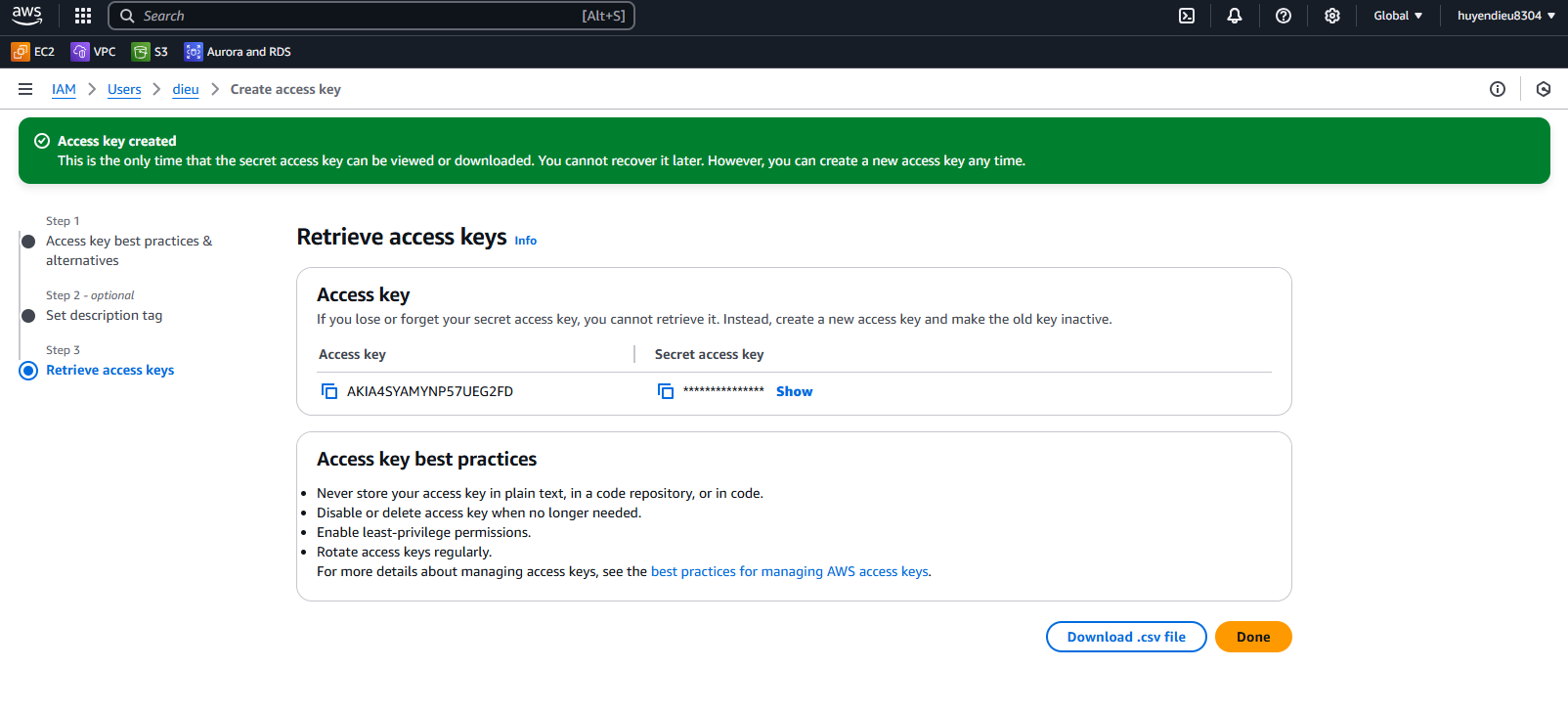
Step 1: For simple, I choose Local code



Step 2: Optional

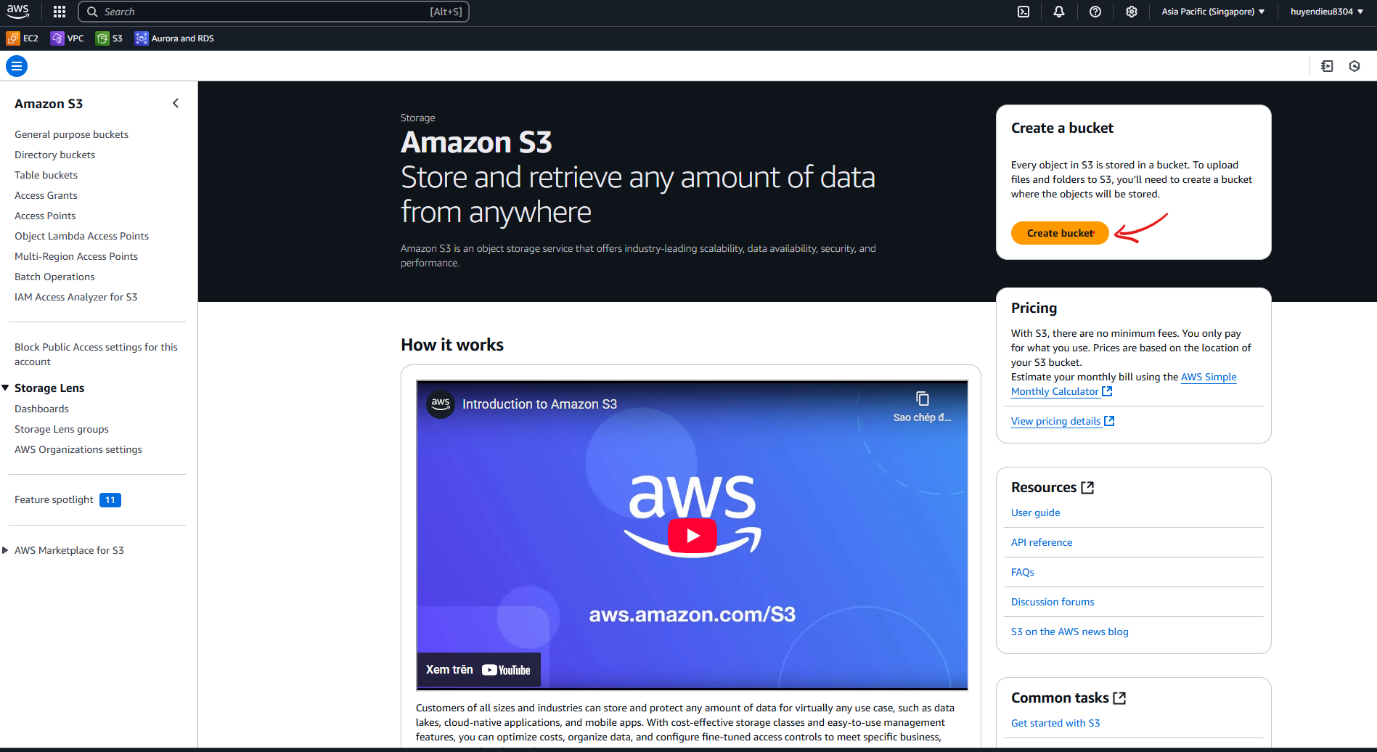


Step 3: Retrieve access key. This step is very crucial, the key just appears once, so you should save it somewhere safe for long term use. These keys will be set up to the environment variables of the app, which I would mention in section 5.

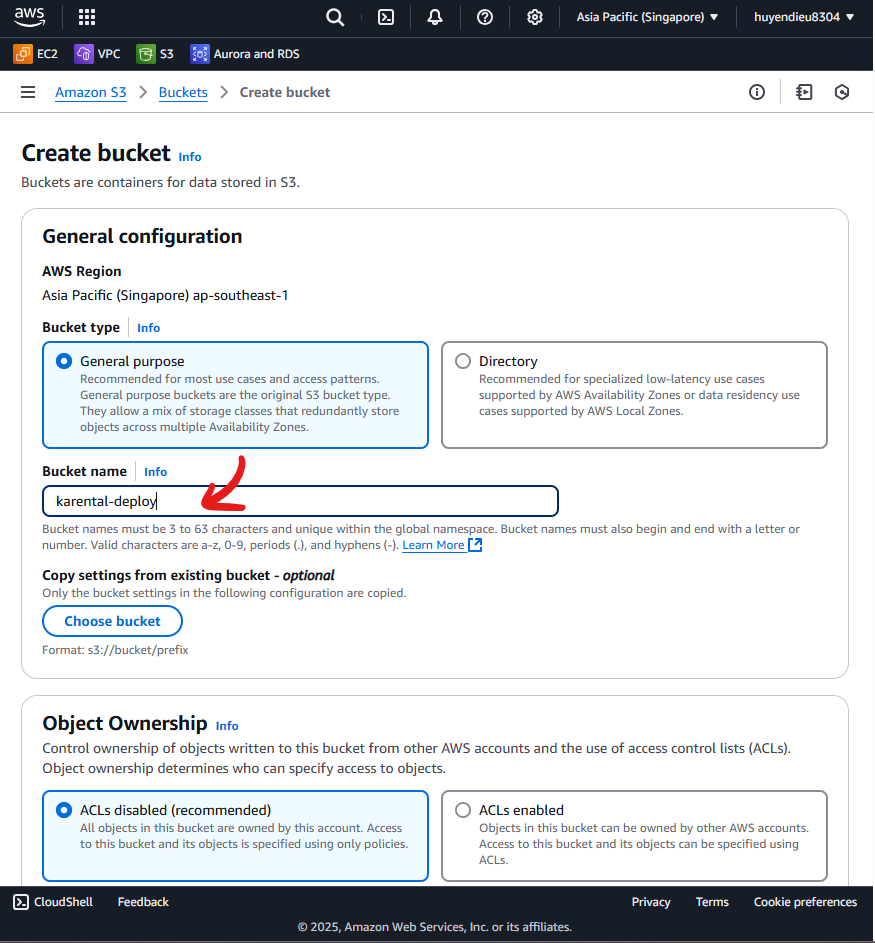
Now, let’s create a bucket!

### S3 – The file storage

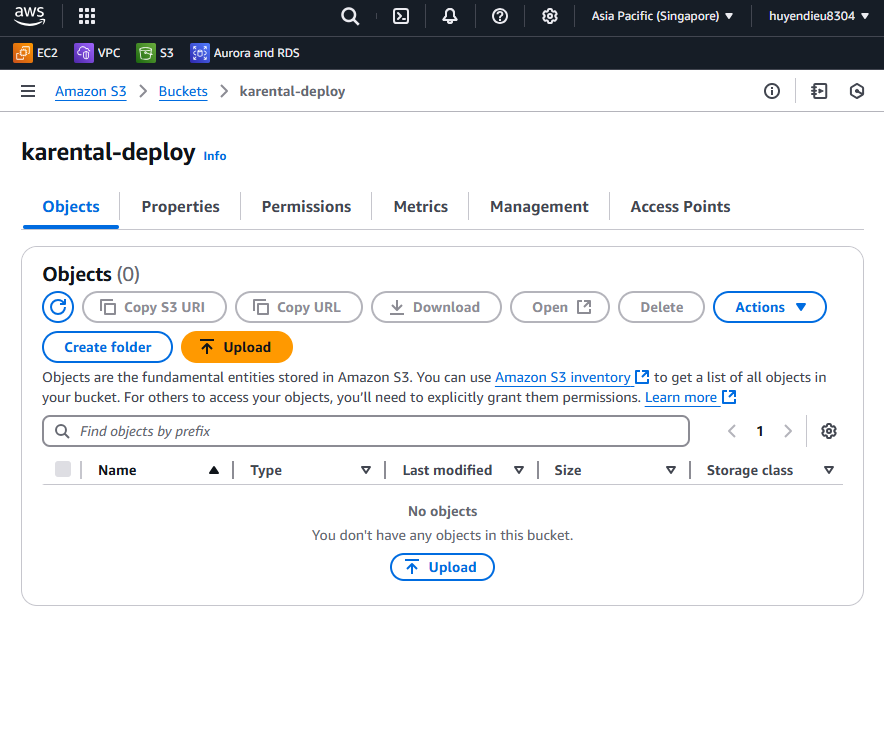
In the search bar on top of the Console Home, type S3 and access it. At the first time you use it, it might look like this.



Click to “Create bucket” button to create a bucket, a bucket is something like your directory on your computer. I would keep everything as default setting.



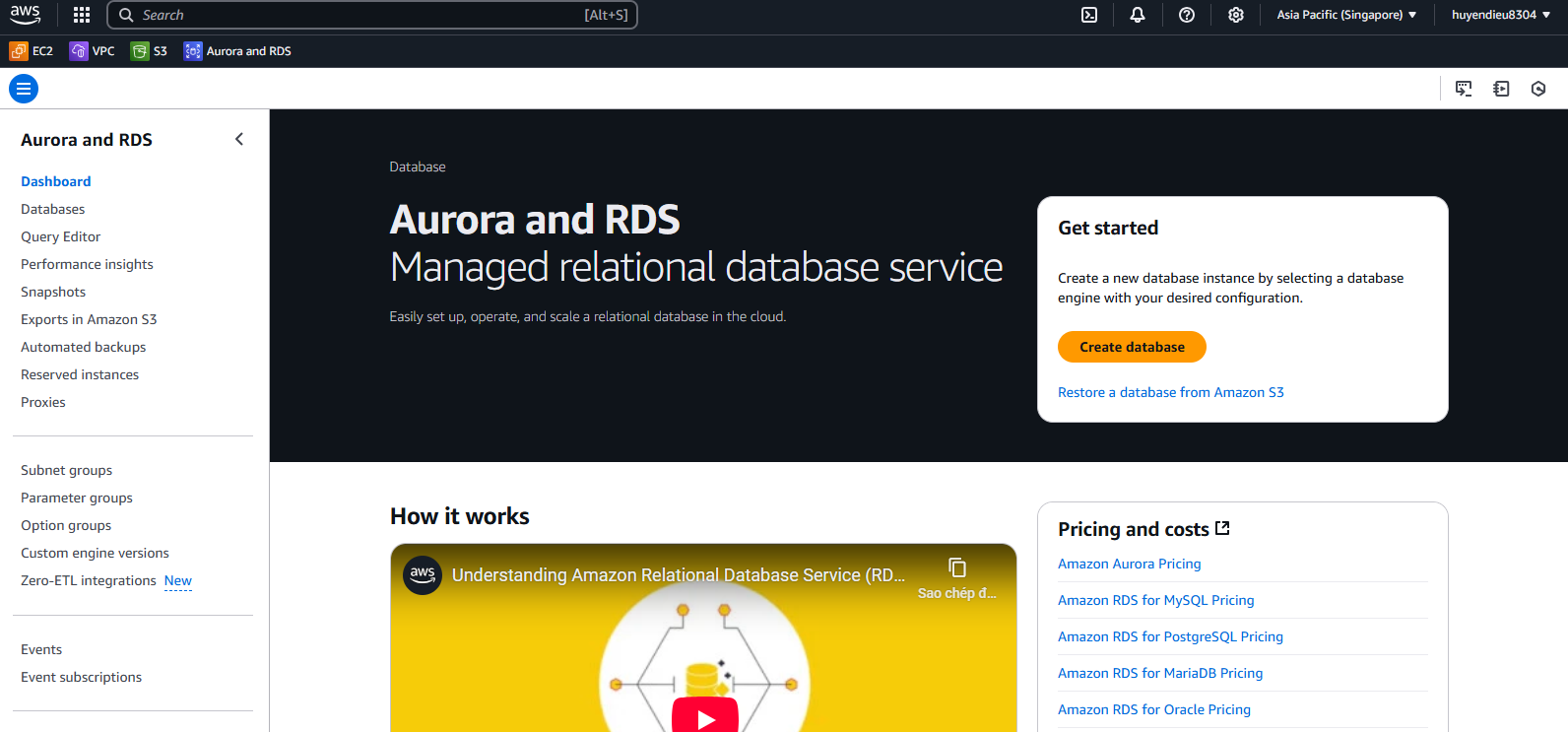
After creating, you would get this



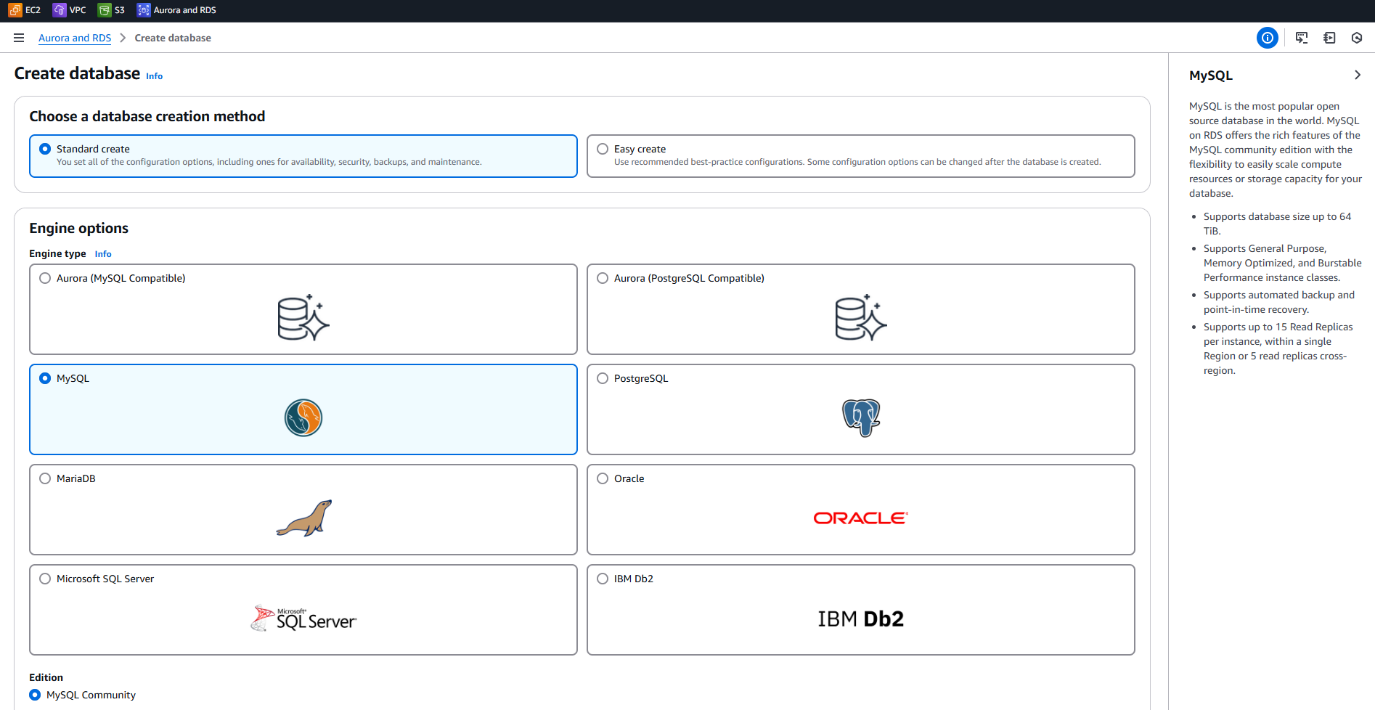
You would need the name of the bucket to put it in environment variable in section 5.

### RDS – The database

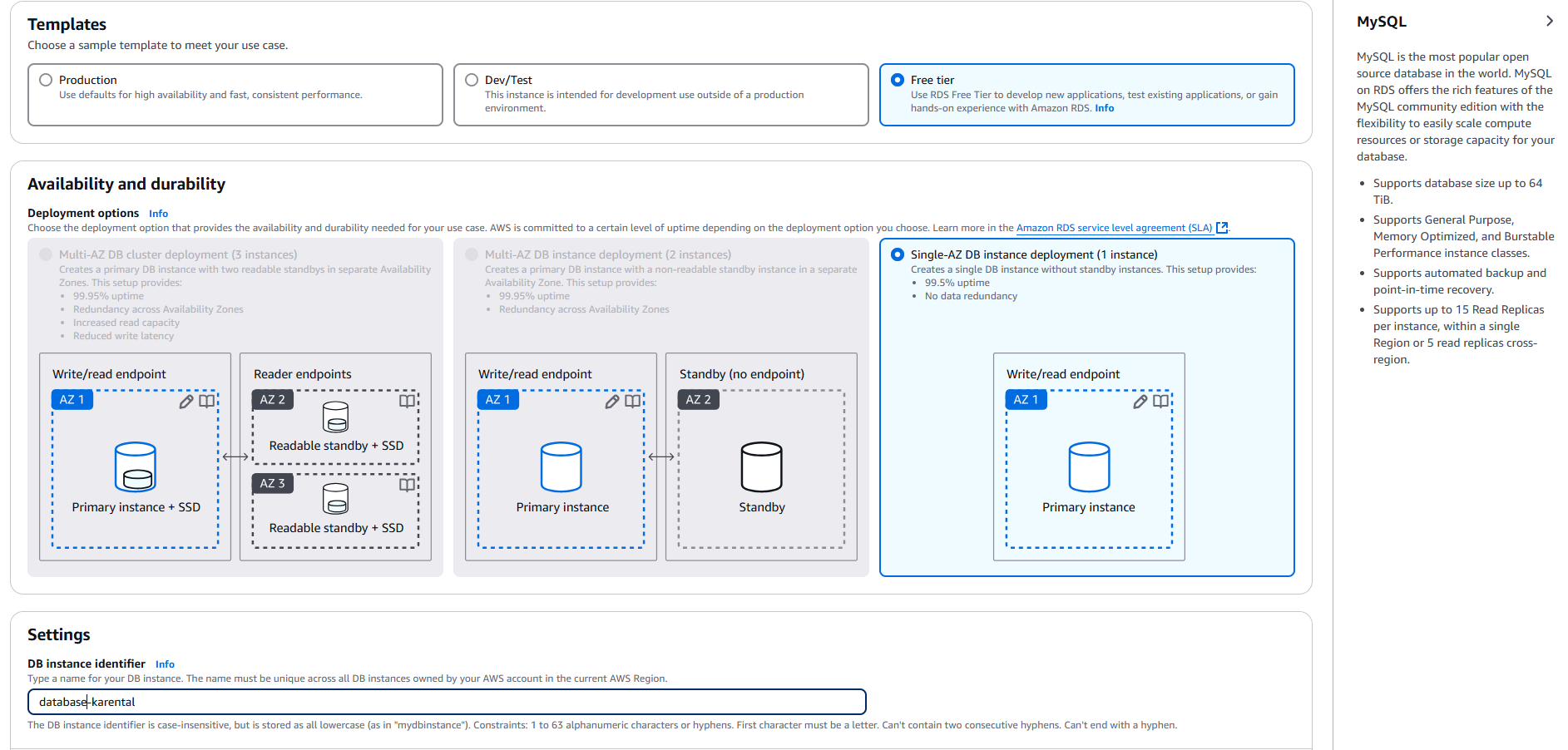
Using the search bar to access RDS, a place where you can create relational database. Click on “Create databse” to continue



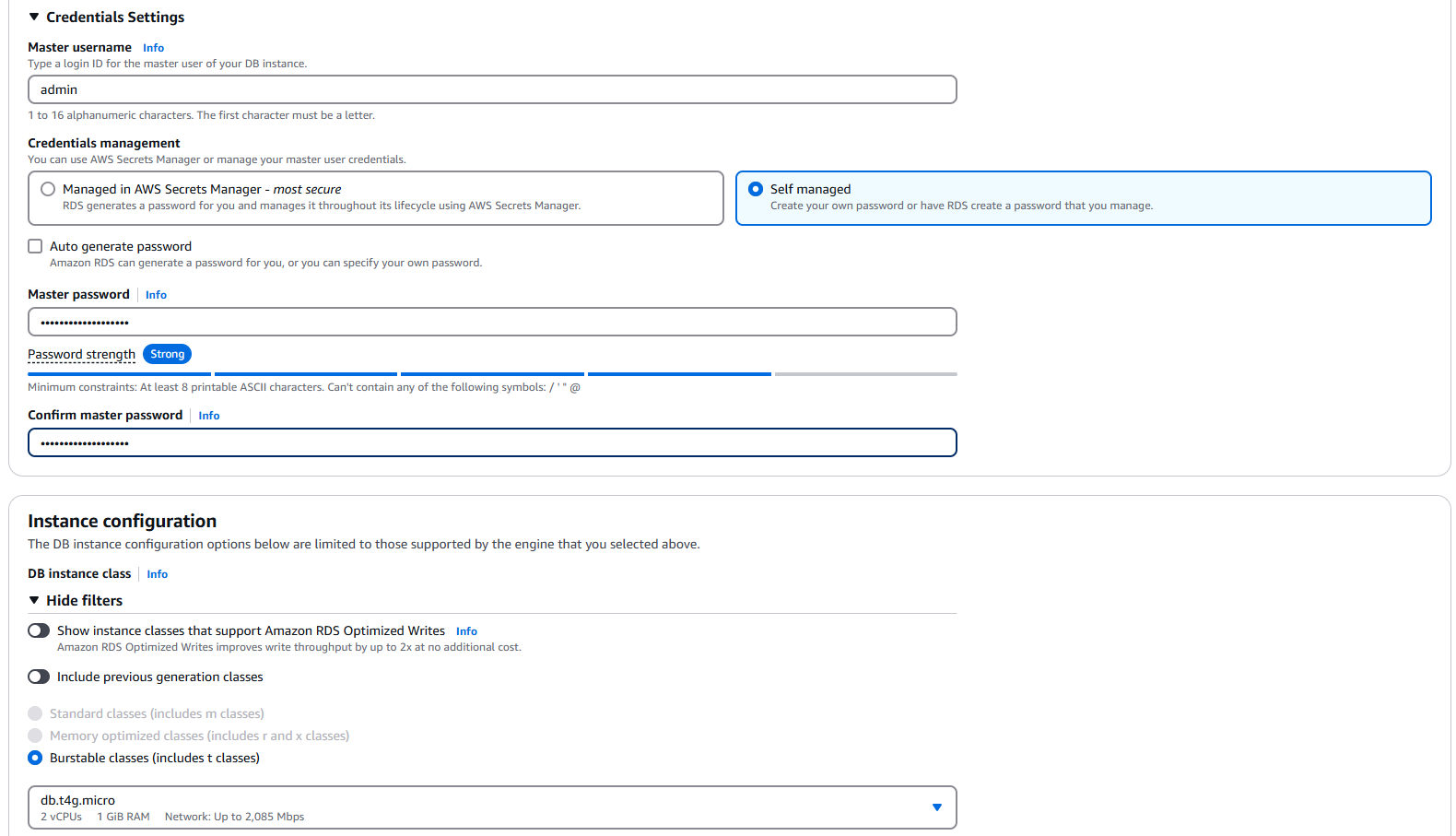
I’ll choose standard creation of MySQL database, Community edition, version 8.0.40



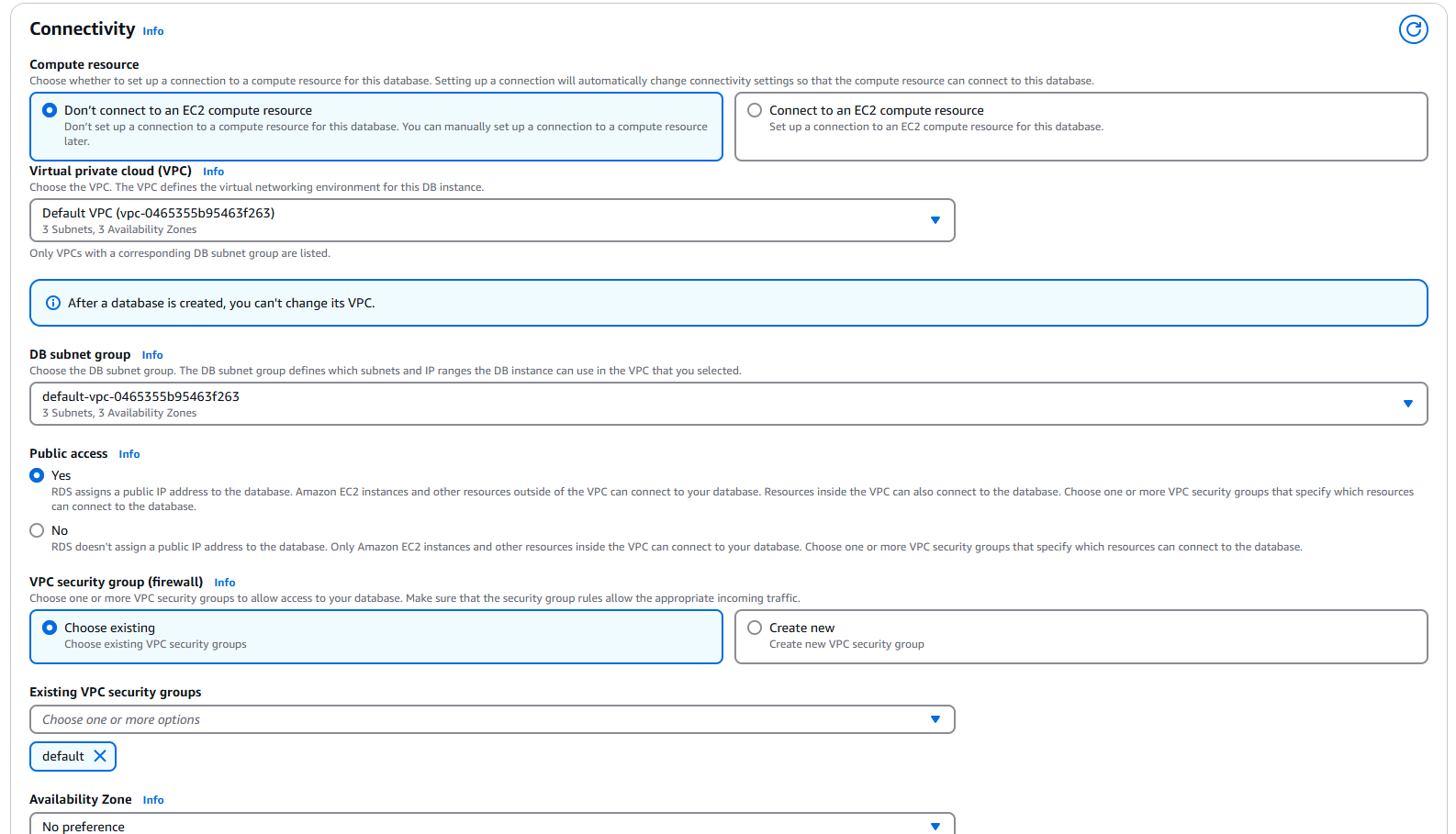
And then, free tier and input the name of the database server.



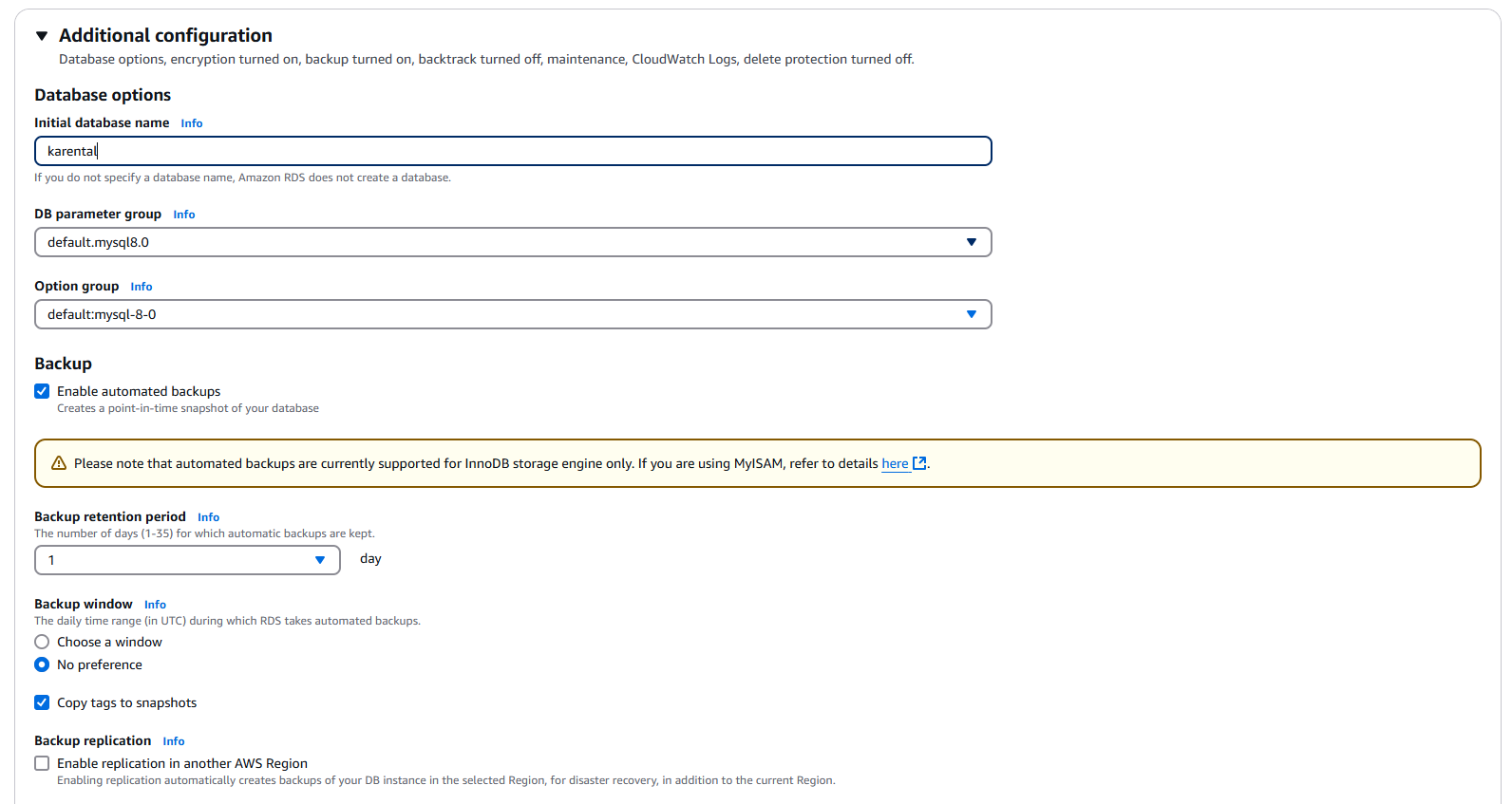
Set up the username and password to connect to the database, please remember this information because it will be used as environment variables in our application.



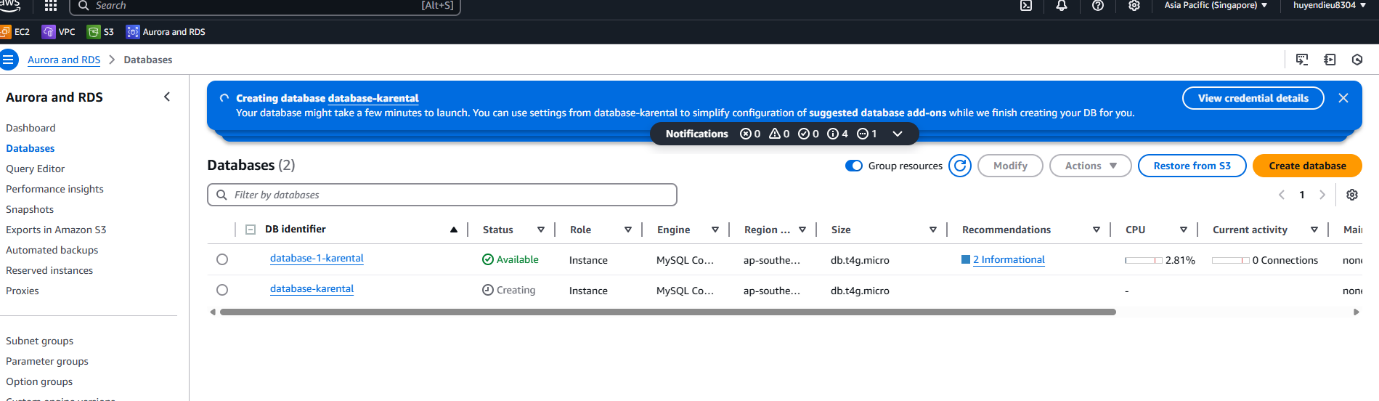
You can keep most of the rest as default, but if you want to access the database from your computer, set the “Public Access” to yes.



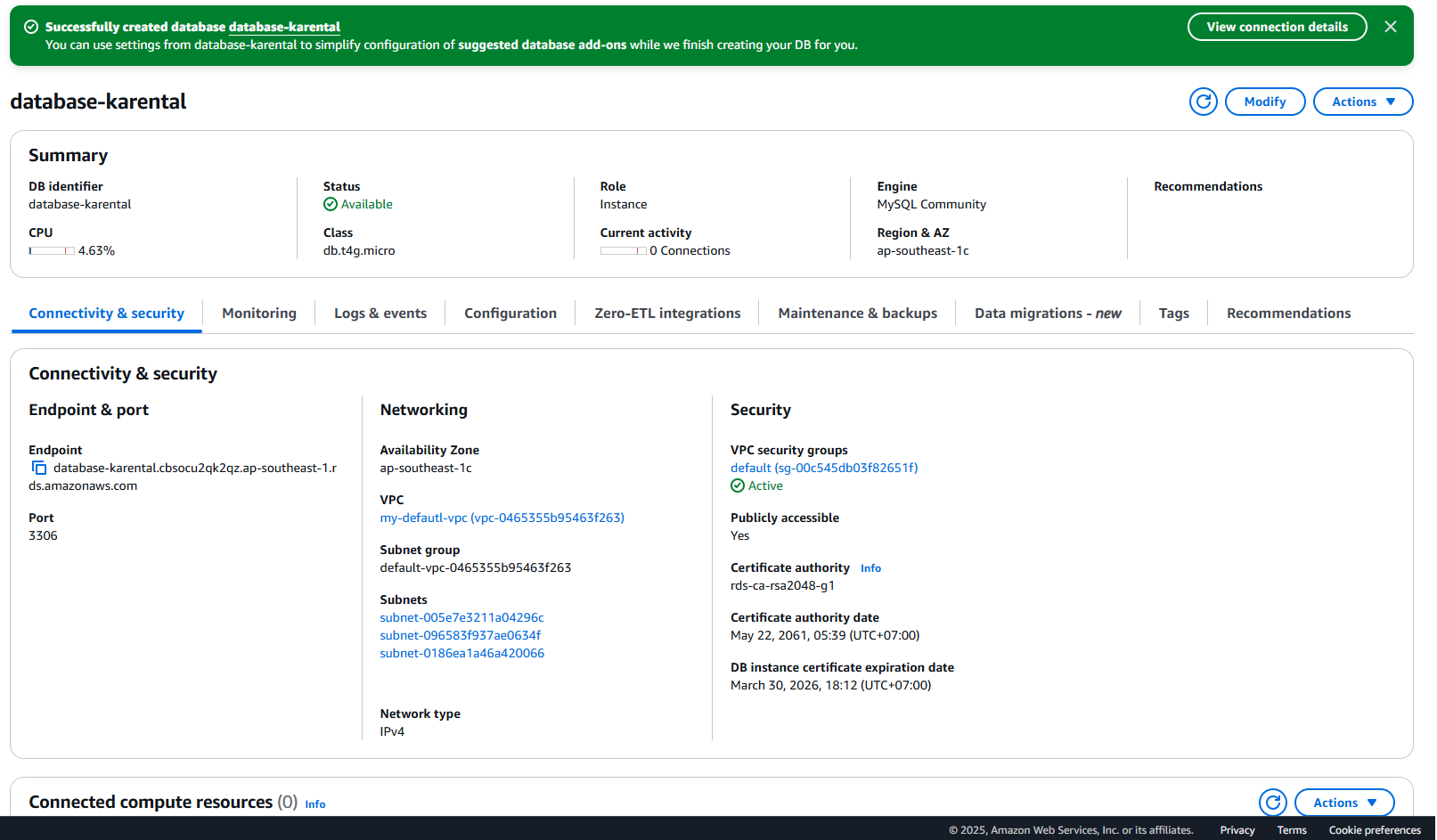
Open “Additional configuration” to create an initial database, which is the database that would exist in the database server right after creation.



Click on “Create database” at the end of the page, wait for a while and you get the 1 MySQL server running.



Don’t worry if you did something wrong, you can modify it in “Modify” section. In the screen below, look at “Connectivity & security” section, using the endpoint as the host in to connect to this database in any database management tool (MySQL workbench, Data Grip or data source management in IntelIj)

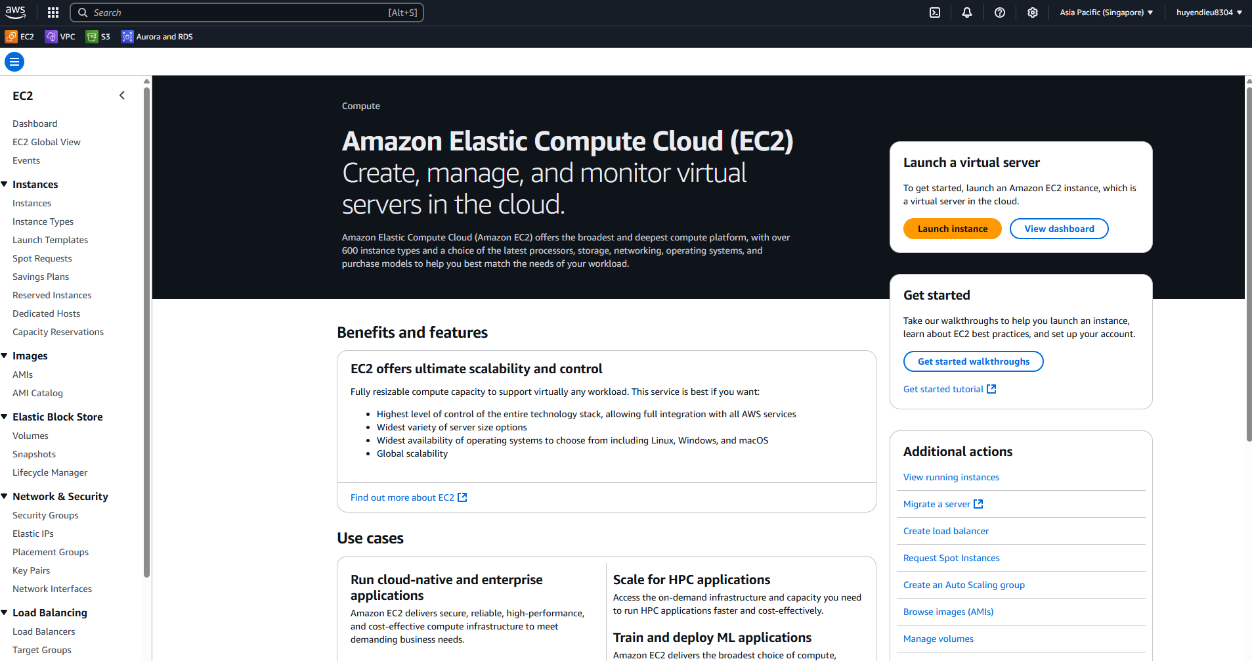
Alright, now you’re having a remote database running on AWS, let’s move on to the next step!

### EC2 – The server

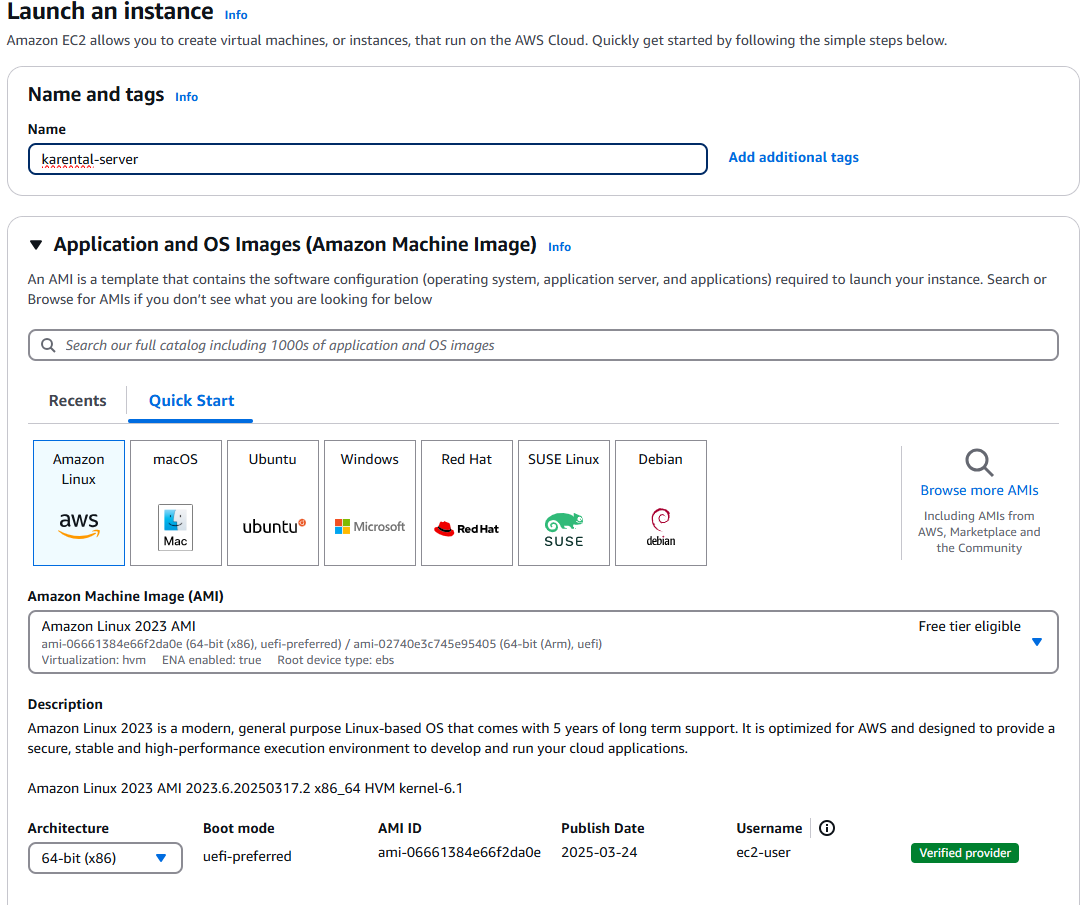
Now, let’s create an EC2 instance, it’s like a virtual private server, where you can run a computer in it, and your application will run on it.

#### c.1 Launch Instance

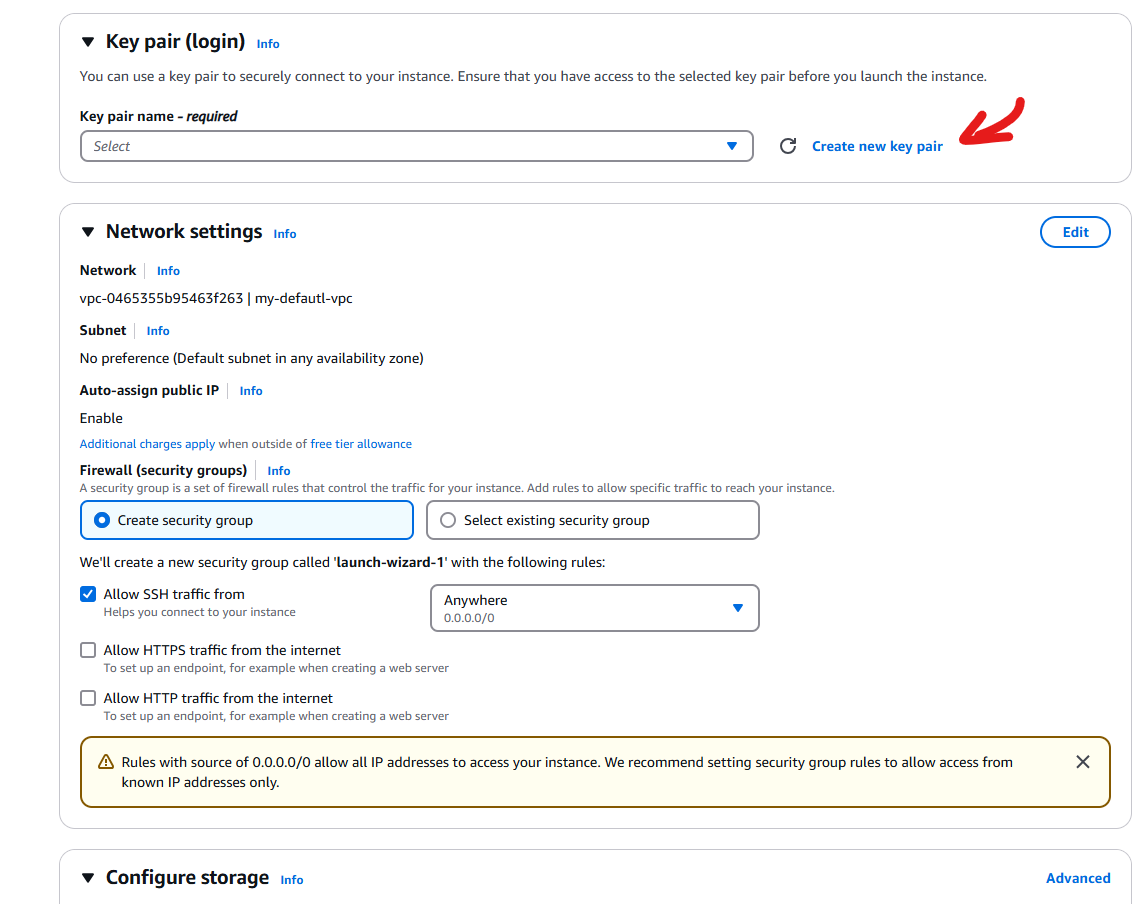
Using the search bar again, search for EC2 and you will get the page below. Click on “Launch instance”

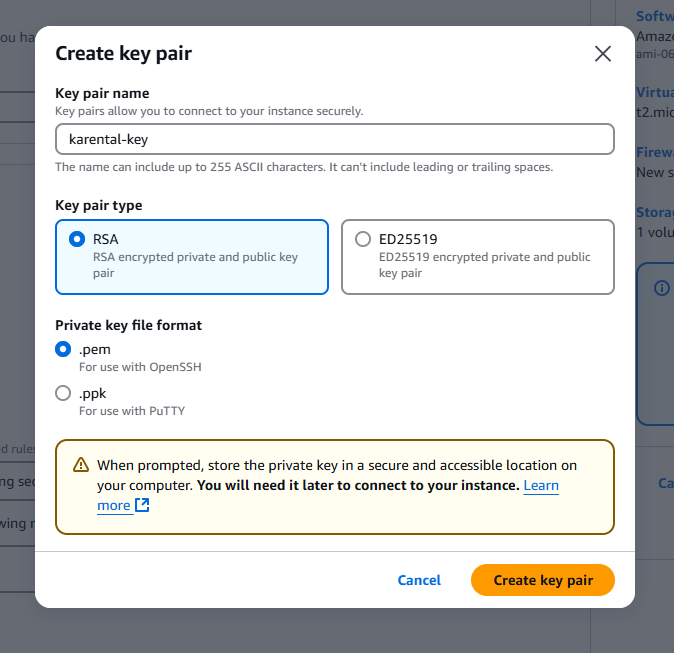


Name your instance, and choose operation, in here I’ll keep as default for simple.

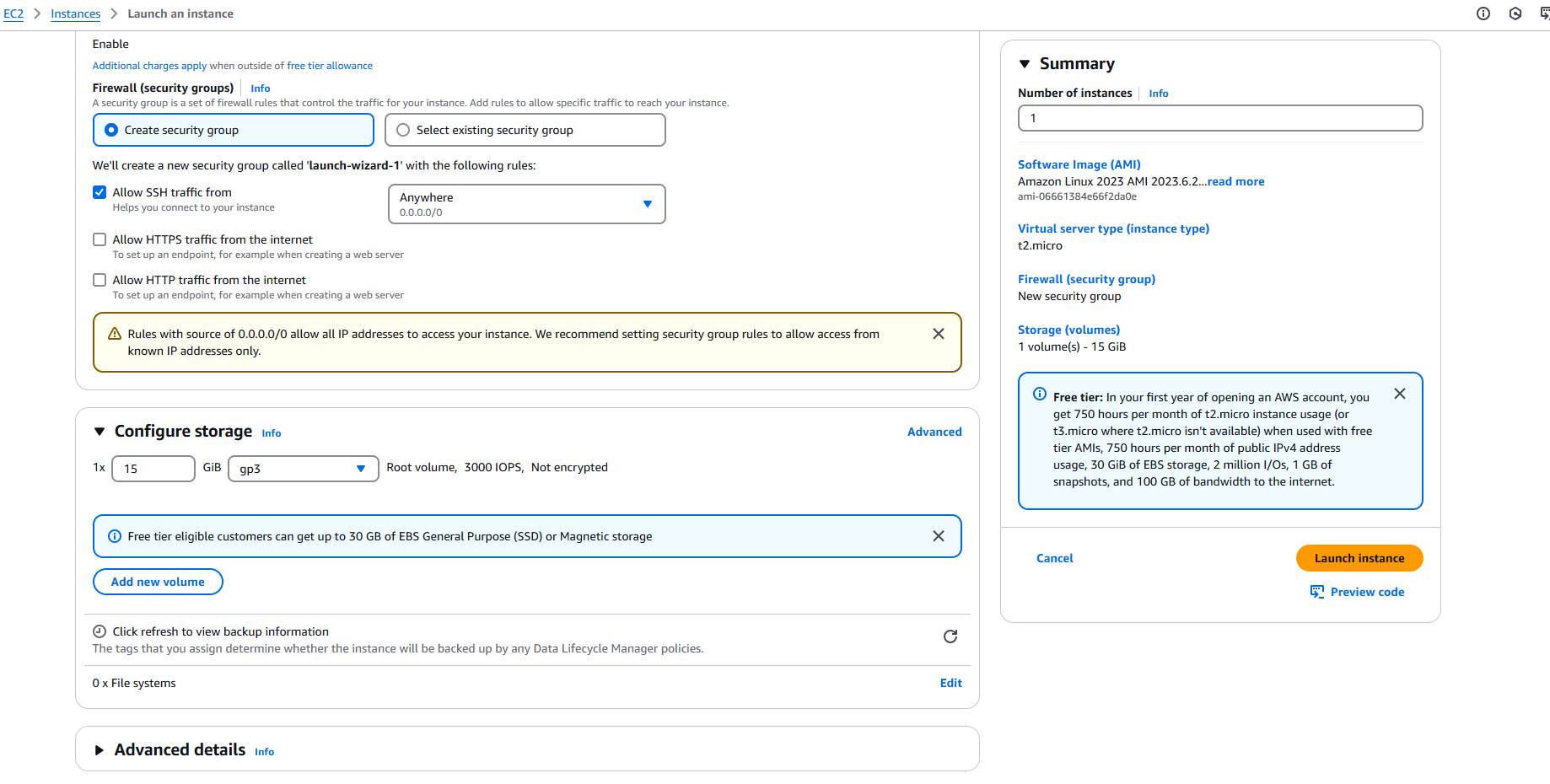


Scroll down a little bit, you would see “Key pair” section. Key pair is used to authenticate access to your instance, click to create a new one.



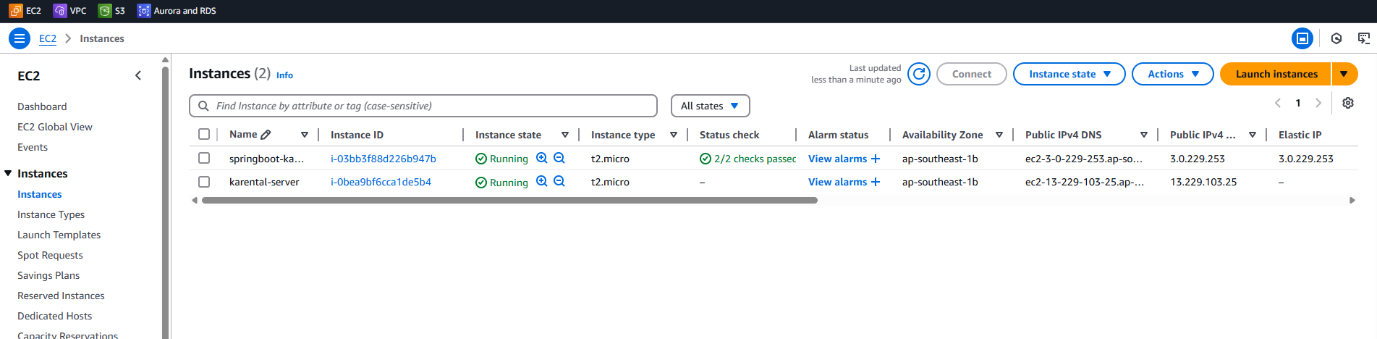


Click “Create key pair” and save the key to a safe place. Now you can keep the rest as default, but I recommend increase the storage to about 15-20GiB. Then in the right side bar you would see the summary of the instance, click on “Launch instance” and continue.

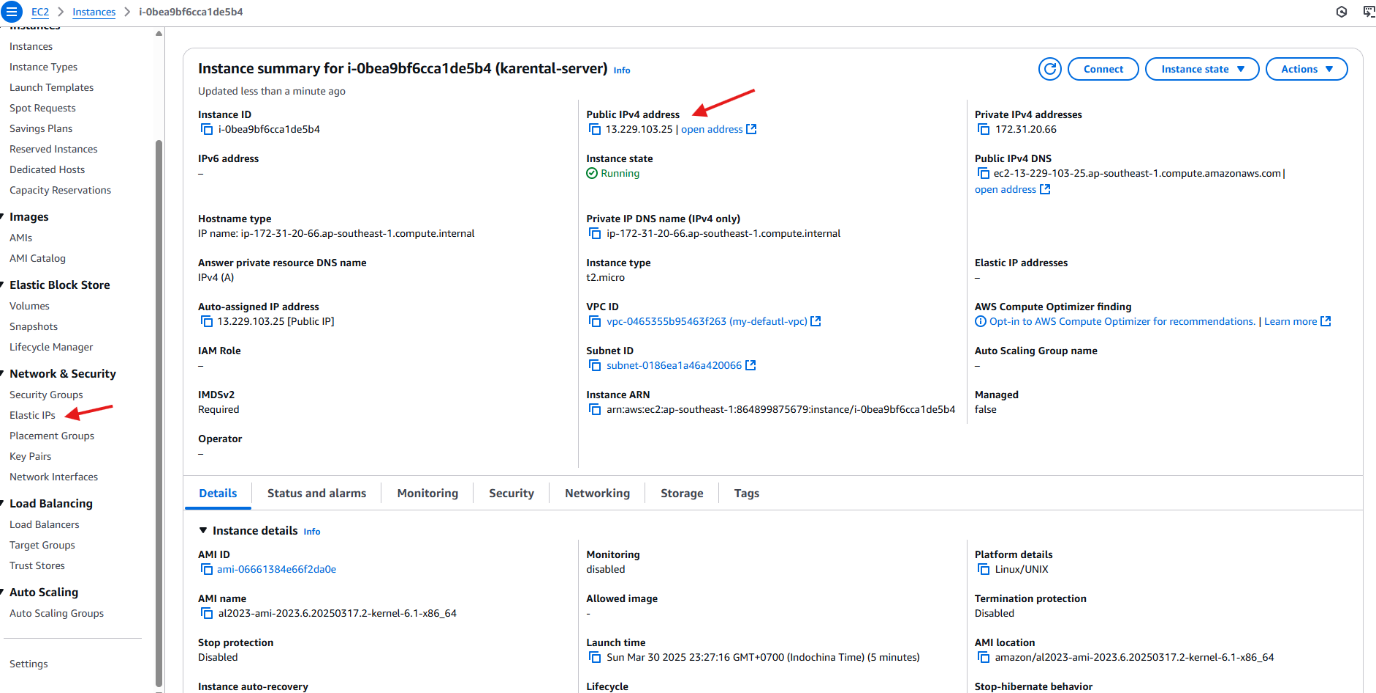


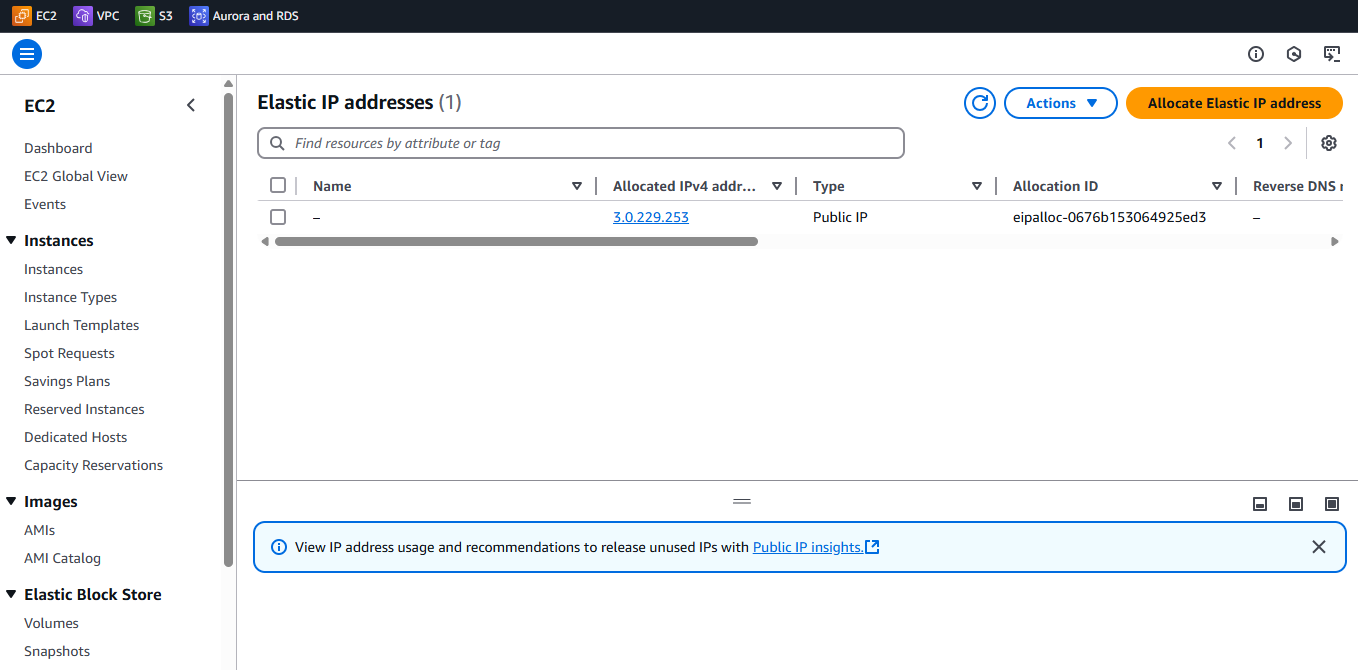
#### c.2 Allocate IP

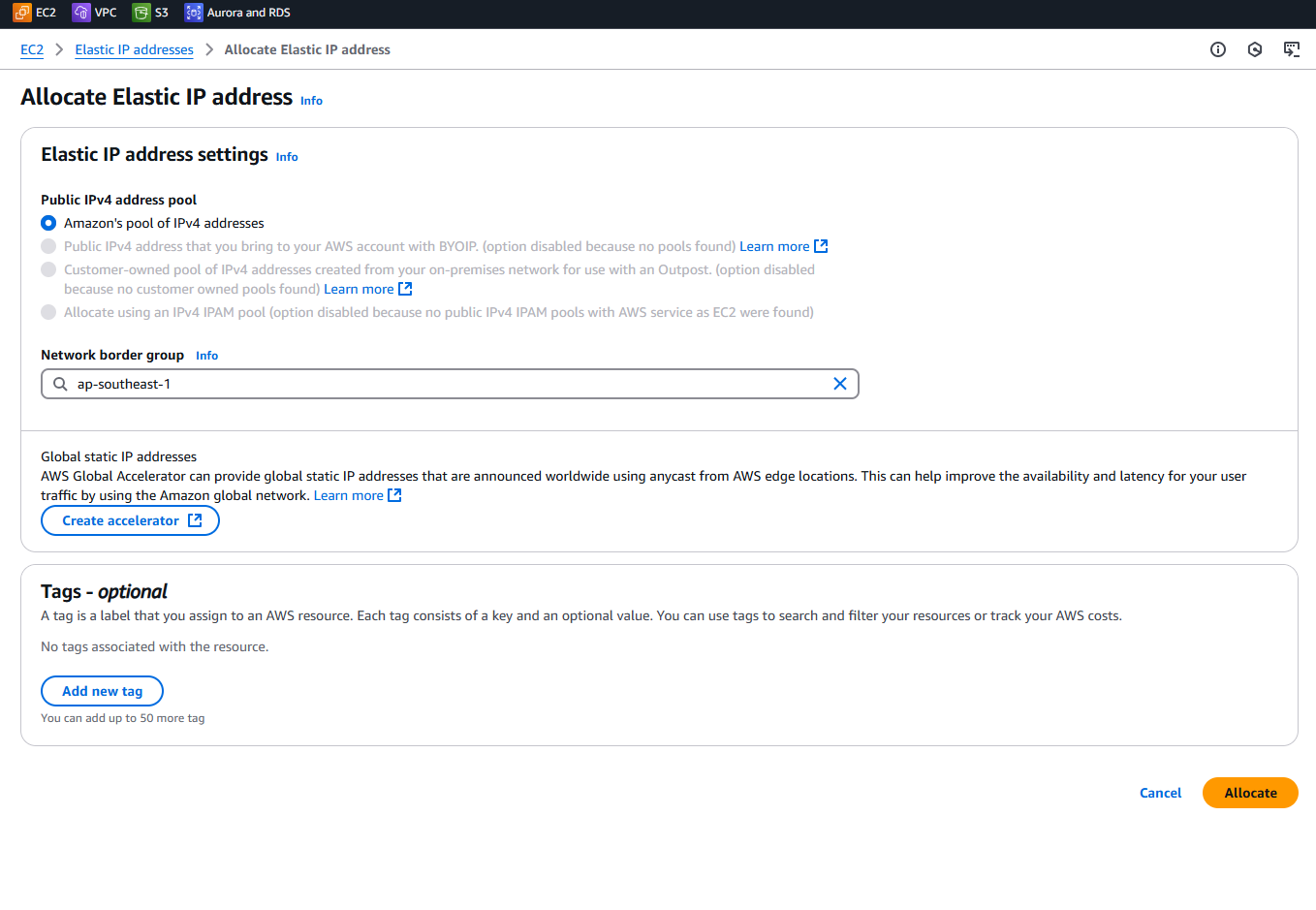
Go to the dashboard, now you see the instance running, click to it.

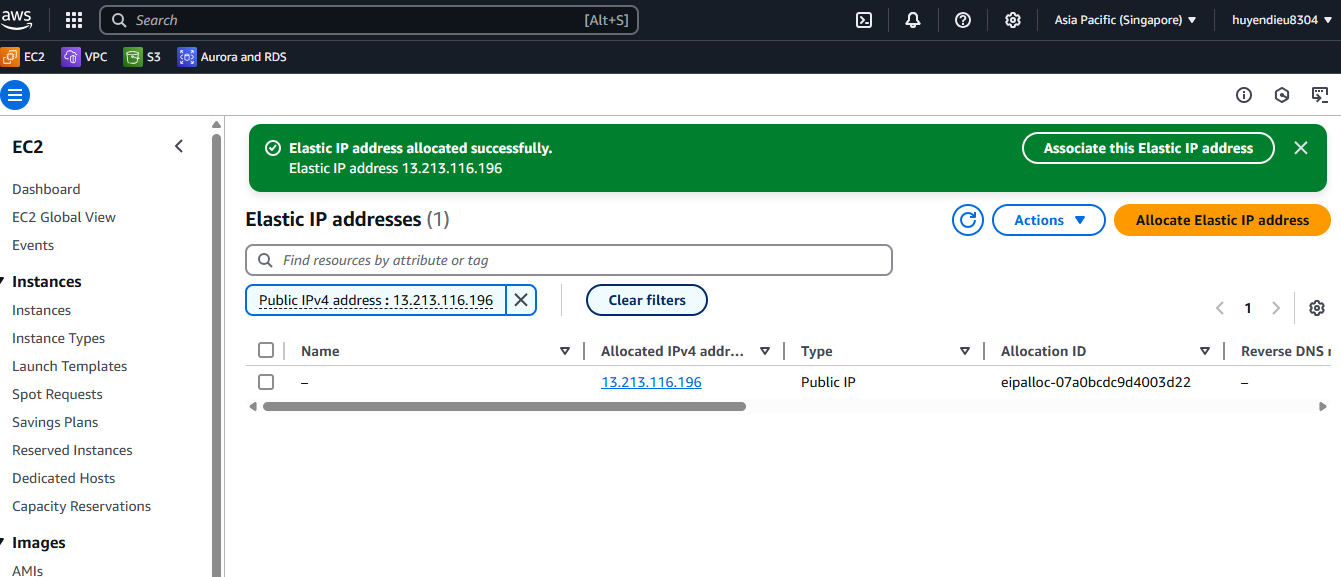


You will see the IP address of your instance, but if you stop the instance and start it again, this address will be change, so that to keep it unchange, open “Elastic IPs” in the left side bar.

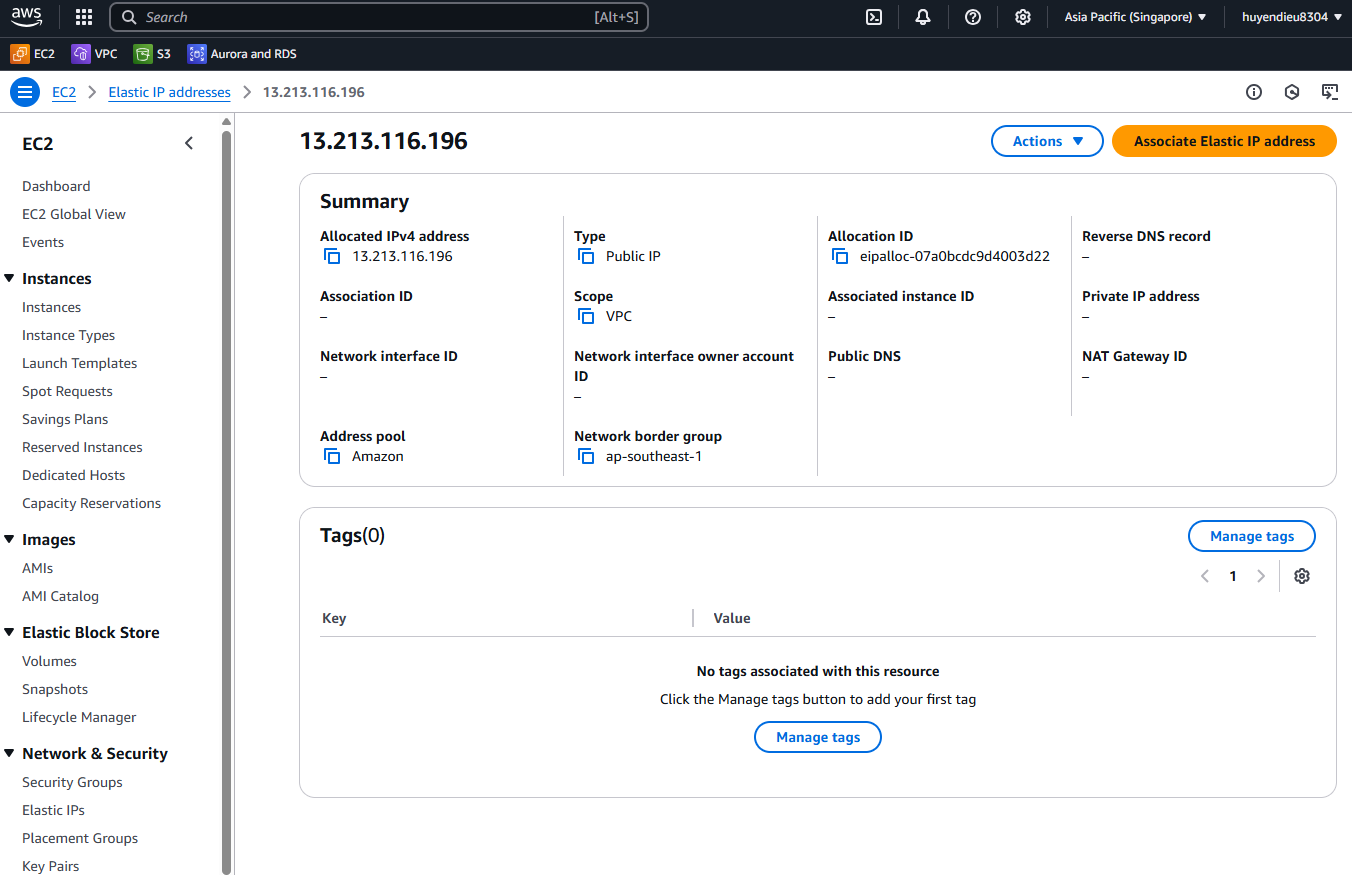


Click on ”Allocate Elastic IP address” and then "Allocate”

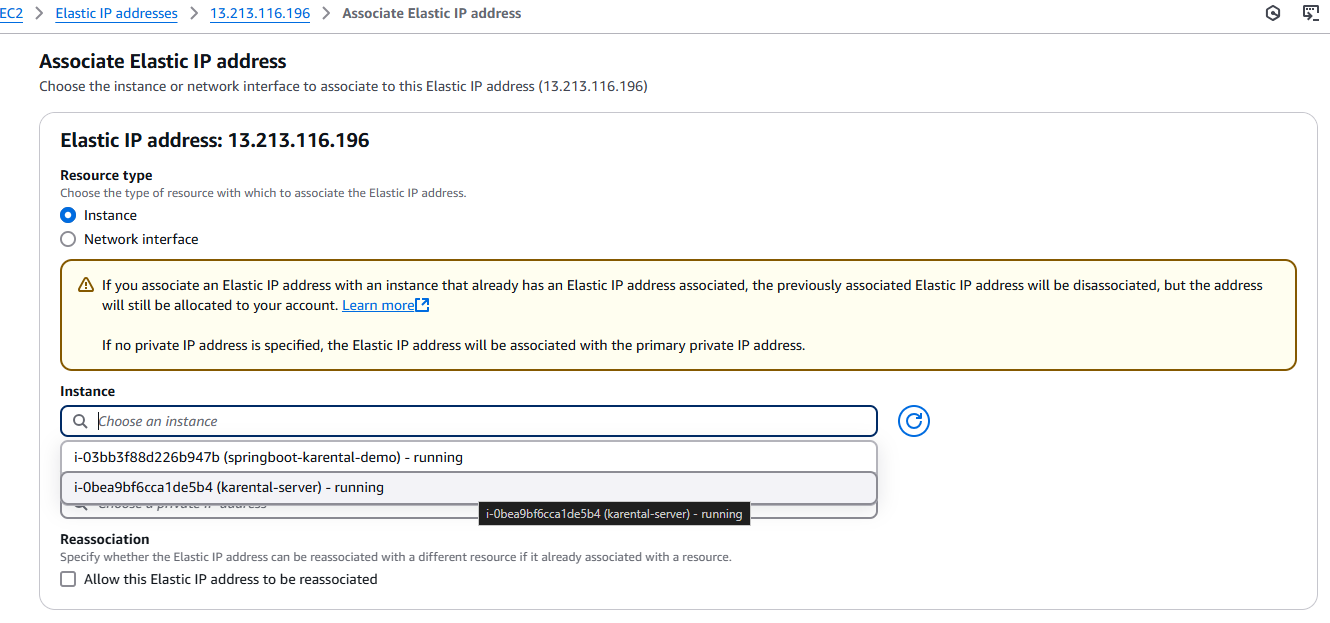


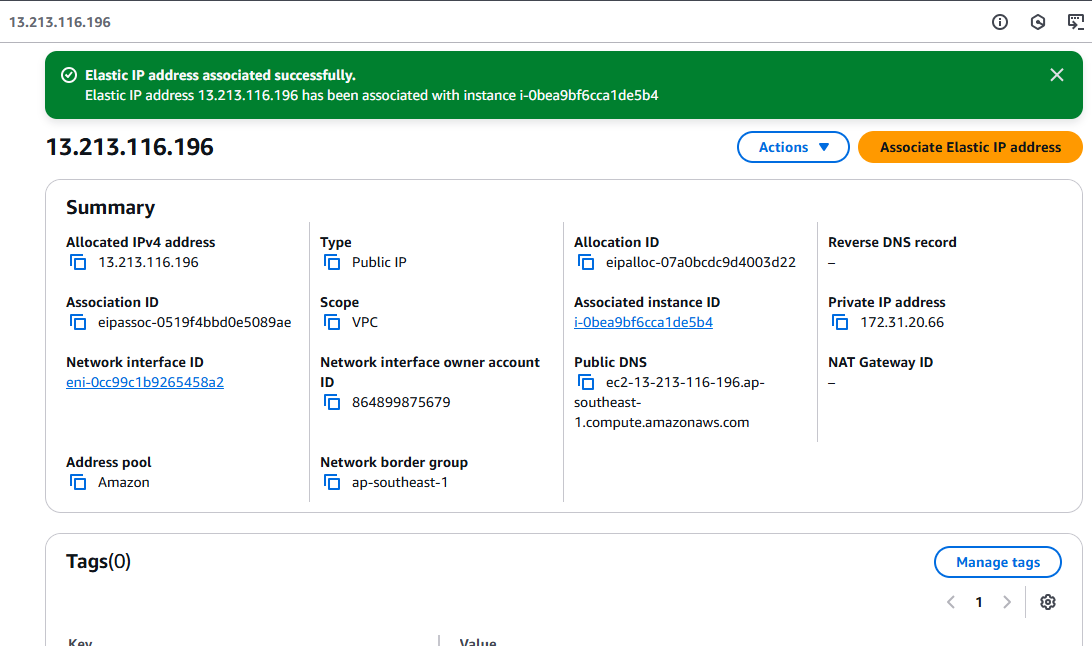


Click to the “Associate this Elastic IP address” or on the IP, then “Associate Elastic IP address”



Choose your instance and click “Associate”



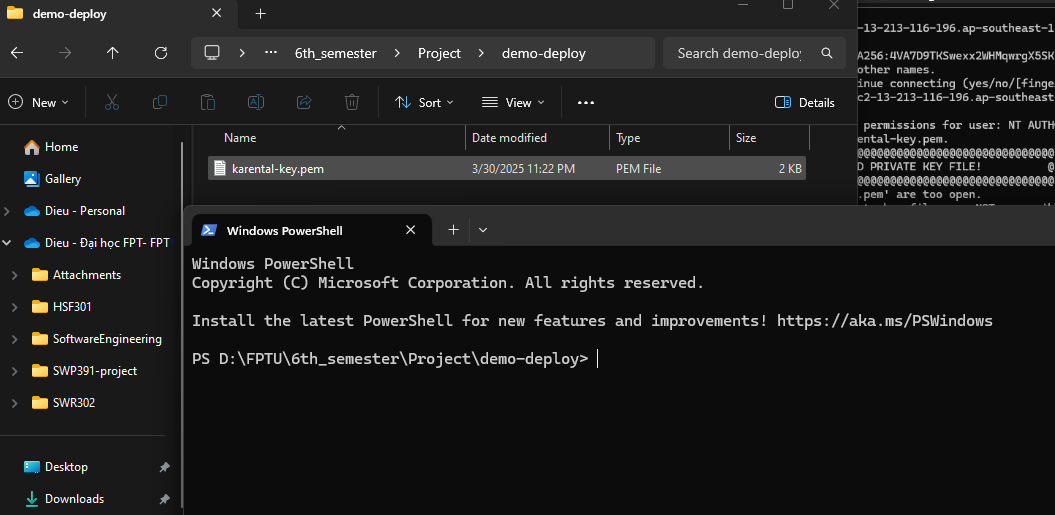


Get back to the instance dashboard, you could see the IP address now is same as your associate IP address.

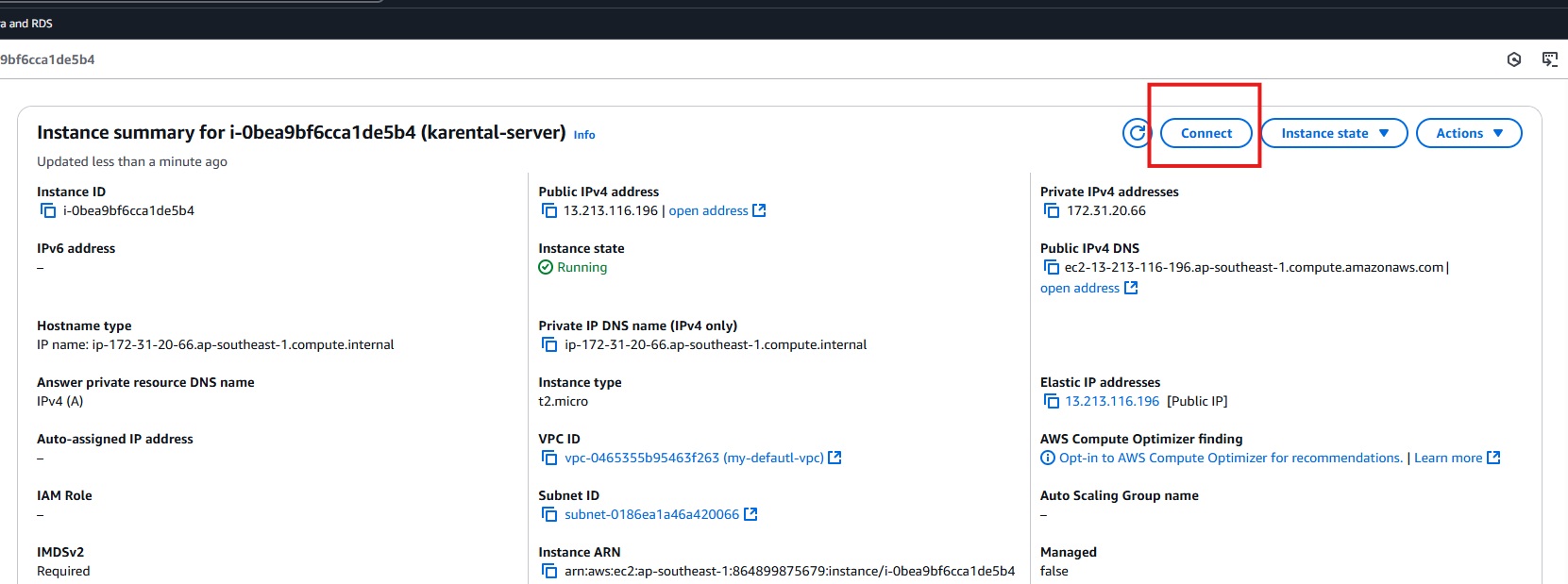


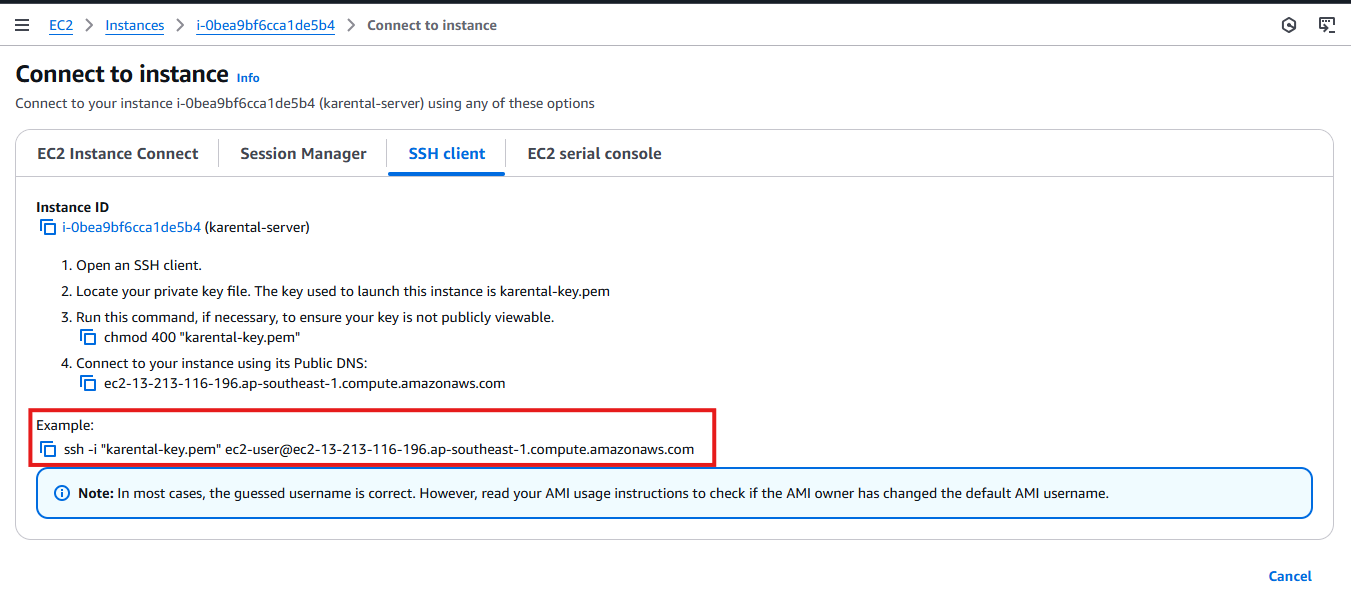
#### c.3 Connect to the instance using SSH

Now to connect to the instance using SSH, open Terminal in the directory where you store the private key.



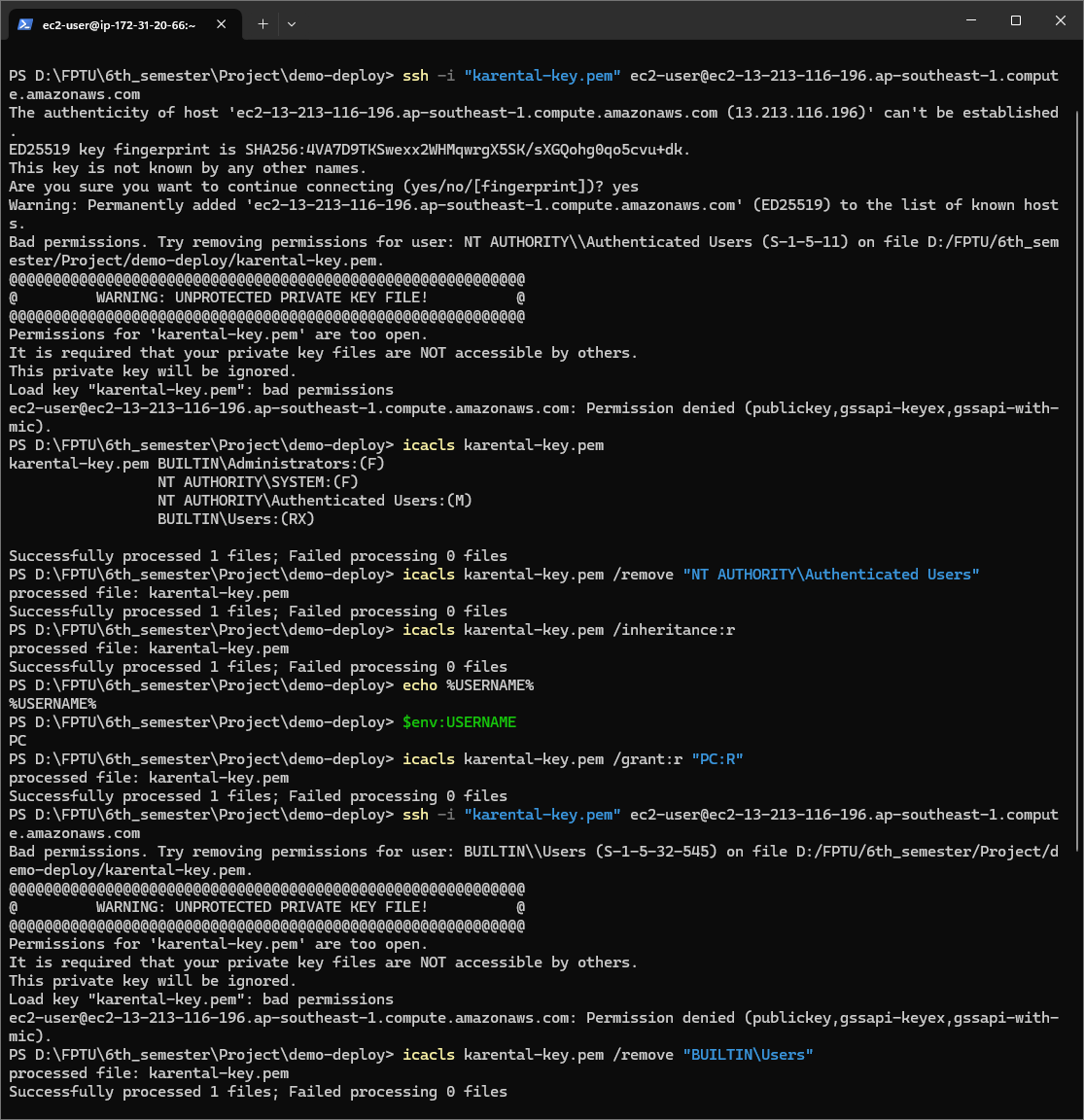
Move back to the dashboard of the instance, click on “Connect”



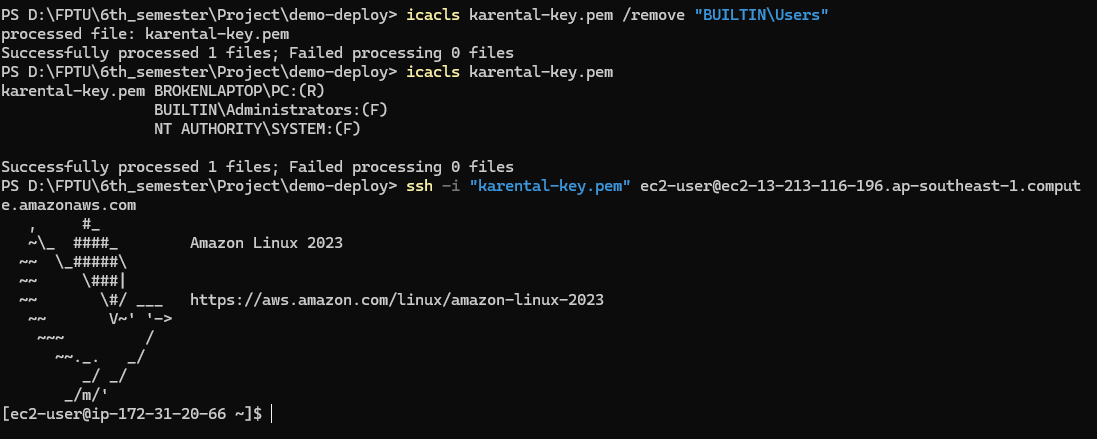
Open section SSH, copy the Example on the page and paste it in the opened termial.

You might not be able to connect to the instance, due to your private key is open to many users on your computer, you need to remove the authority from other users, and just keep read authority for your current user. In the picture below, is how I do that.

icacls springboot-karental-demo.pem



After that, run the coppy command again and you will get this



#### c.4 Set up instance

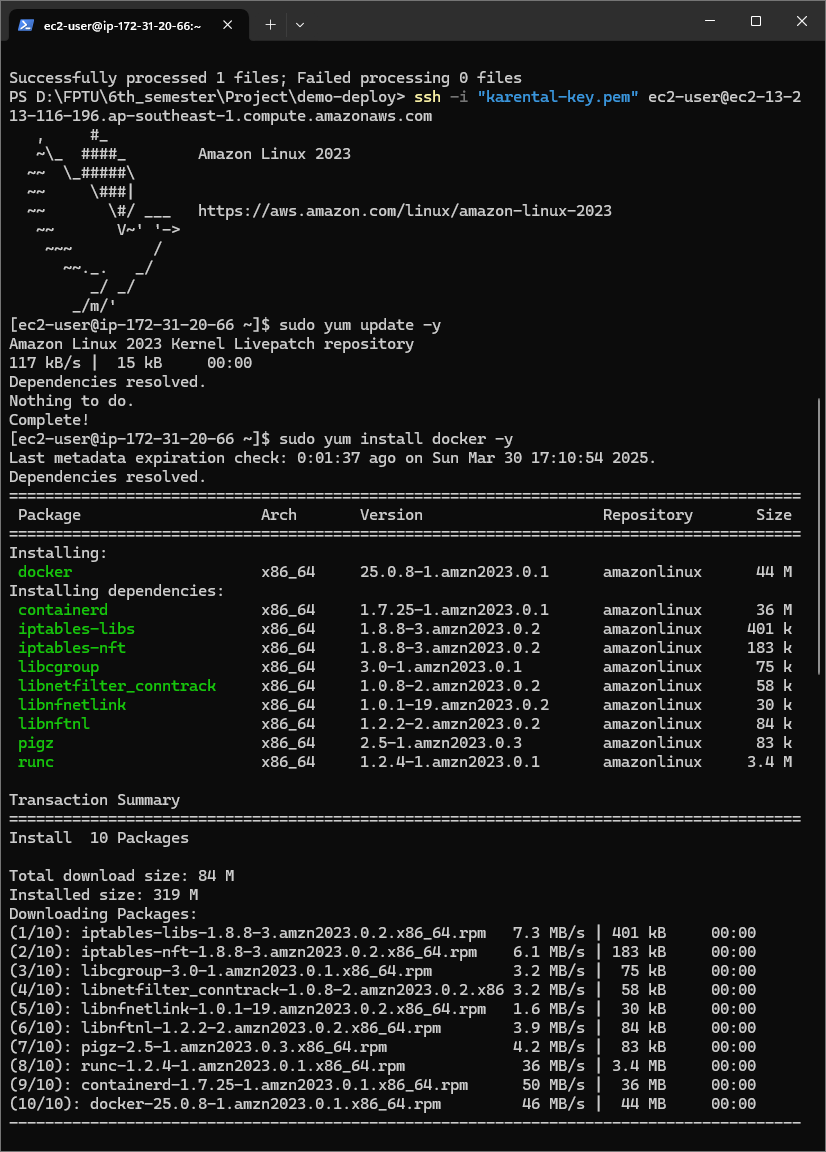
Ok, now you should update all your package in your server, run following command to set up your “computer”

$ sudo yum update –y

* sudo: supperuser do, it means that you’re running this command as a administrator with the root authority.
* yum: package manager, it’s kinda like npm but for operating system.
* update: update all the package to the latest version
* -y: auto yes when asked “do you want to update”

$ sudo yum install docker –y

* Install docker



$ sudo systemctl start docker

* systemctl: the command-line tool to interact with system services
* start: tells sytemctl to start a service
* docker: the name of the service being started

$ sudo systemctl enable docker

* enable : Enanle Docker to start automatically at boot



$ mkdir -p ~/.docker/cli-plugins

* mkdir: create directory
* -p: if directory exist, error not occur
* ~/.docker/cli-plugins: the directory, we will put docker compose in here

$ curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o ~/.docker/cli-plugins/docker-compose

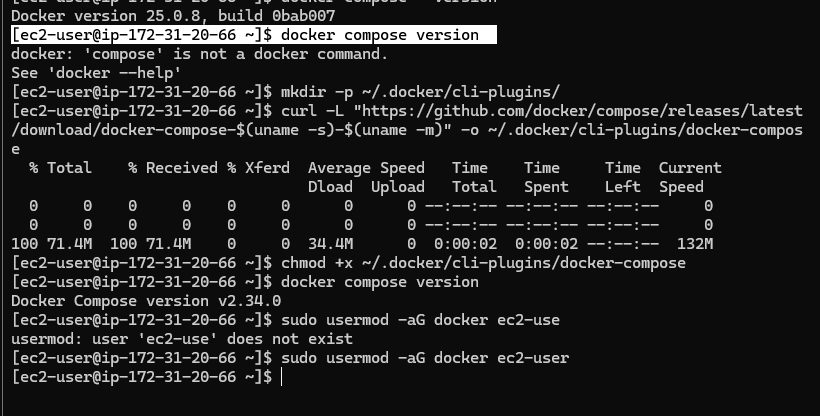
* curl -L: download file from URL, -L help to track redirect
* "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)"
  + URL to get latest docker compose
  + $(uname -s)-$(uname -m): determine the operating system and CPU architect to get the right file
* -o ~/.docker/cli-plugins/docker-compose: where to save the file

$ sudo chmod +x ~/.docker/cli-plugins/docker-compose

* Allow the file to be able to execute

$ sudo usermod -aG docker ec2-user

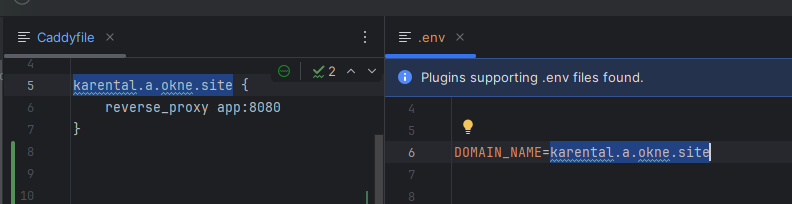
* Adding the current user to docker group so that here after, you no longer need to run docker command start with “sudo”
* usermod: modify user’s information
* -aG: add user to group, keep other groups that this user belongs to



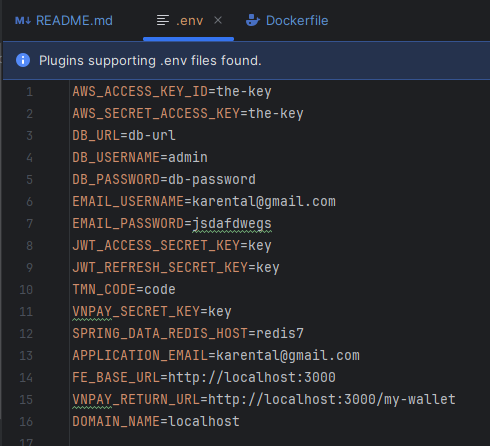
## Modify docker compose

Now, come back to the source code, in the root directory of the project, there is a directory named “/deploy”, most of what you need to run the application is in that directory.

Firstly, you need to change the domain name in the ./deploy/Caddyfile so that when your app’re running, it can get certificate from Let’s encrypt. You also need to update DOMAIN\_NAME in “.env” - a file contains all environment variables.

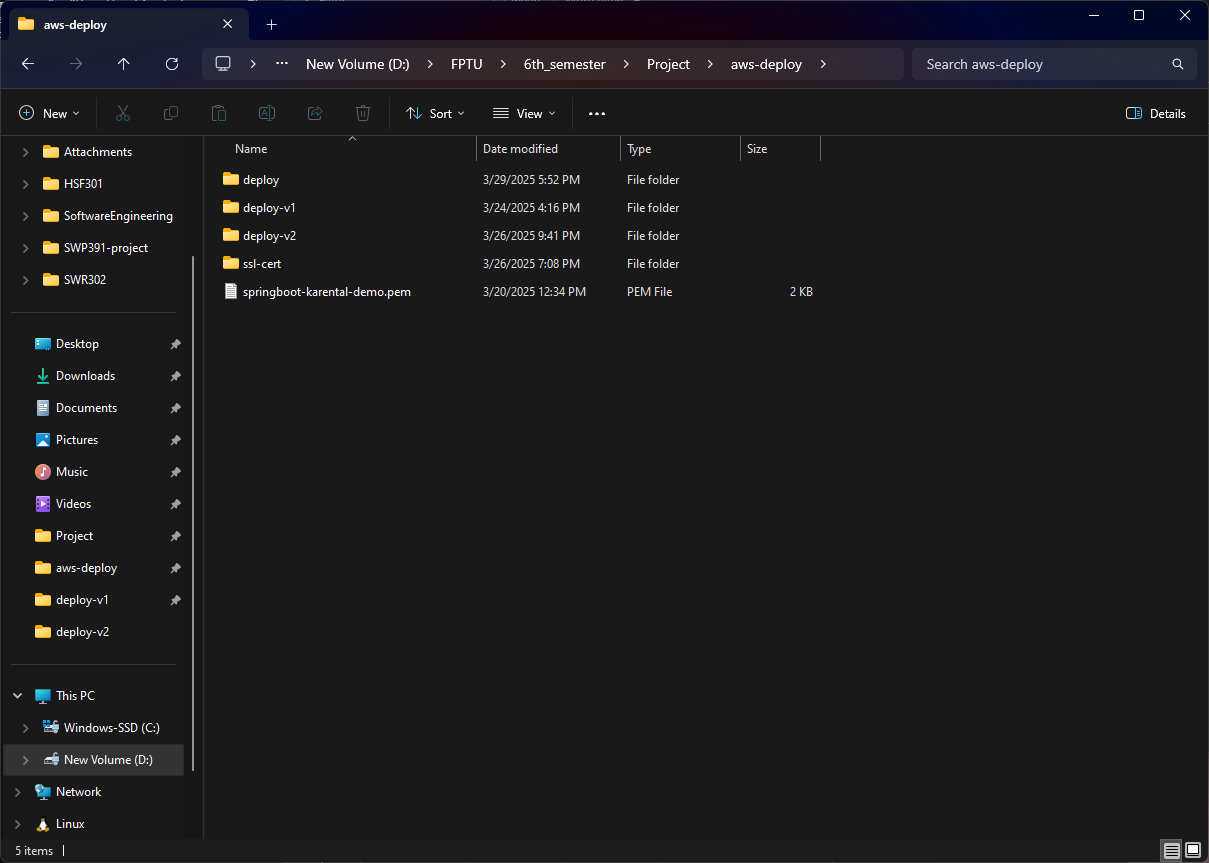


You just need to provide others variable in the “.env” file so that the docker compose could get environment variables to run the app image.

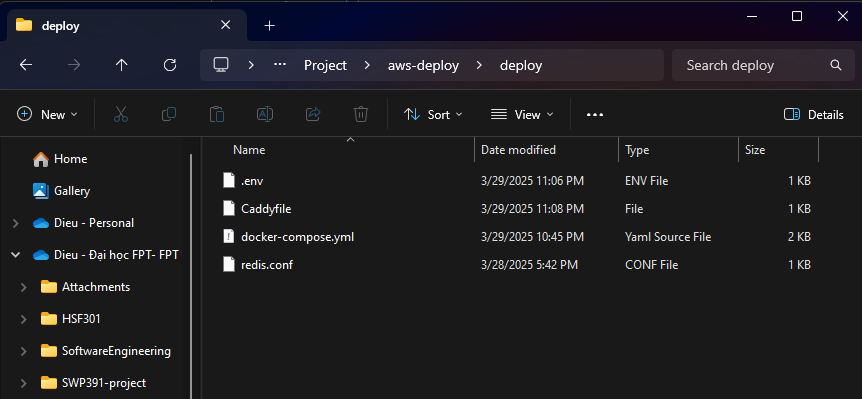


|  |  |
| --- | --- |
| **Environment Variable** | **What is it used for** |
| AWS\_REGION | The AWS region where you choose to run S3 on |
| AWS\_S3\_BUCKETNAME | The S3 bucket name that you use to save file |
| APPLICATION\_EMAIL | The email address that all the mail in the system is sent from |
| AWS\_ACCESS\_KEY\_ID | The key id to access AWS, this is used to access AWS S3 |
| AWS\_ACCESS\_KEY | The key to access AWS, this is used to access AWS S3 |
| DB\_URL | The url of the datasource |
| DB\_PASSWORD | The password used to access database |
| DB\_USERNAME | The username used to access database |
| DOMAIN\_NAME | The domain of the website where you access it, if you run this application in local, input "localhost" |
| EMAIL\_PASSWORD | The password of the application's email, this is used in Spring mail |
| EMAIL\_USERNAME | The username of the application's email, eg: [karental@gmail.com](mailto:karental@gmail.com) |
| FE\_BASE\_URL | The base url of the front end, eg: if you have a react app run local it would be [http://localhost:3000](http://localhost:3000/) |
| JWT\_ACCESS\_SECRET\_KEY | The key to sign JWT access token |
| JWT\_REFRESH\_SECRET\_KEY | The key to sign JWT refresh token |
| SPRING\_DATA\_REDIS\_HOST | The host where Redis run, eg: localhost for Redis running on local |
| TMN\_CODE=T3GTKJIG | The VNPay params to verify request |
| VNPAY\_SECRET\_KEY | The VNPay params to verify request |
| VNPAY\_RETURN\_URL | The address directed after successfully make transaction, eg: [http://localhost:3000/#/my-wallet](http://localhost:3000/" \l "/my-wallet) |
| MAILTRAP\_PASSWORD | If you not run the app in "prod" profile, please use Mailtrap, put Mailtrap password of your inbox in this variable |
| MAILTRAP\_USERNAME | If you not run the app in "prod" profile, please use Mailtrap, put Mailtrap username of your inbox in this variable |

Then copy the directory /deploy in the source code to there.



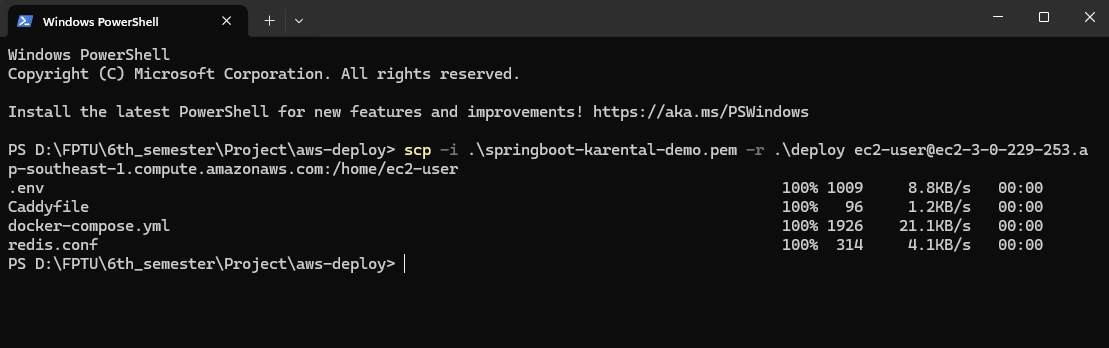
Put the “.env” file in here and we will start to put our project on EC2



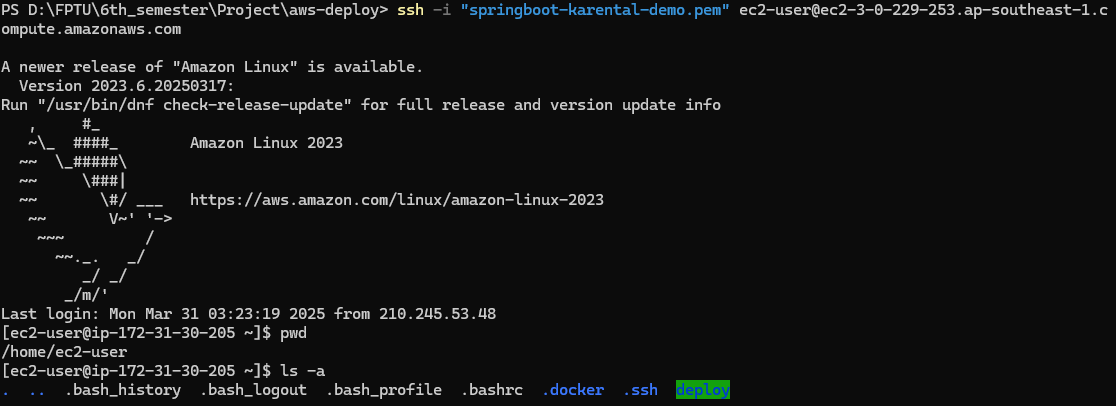
Open Terminal in the folder where your private key locates. Run following command.

scp -i .\springboot-karental-demo.pem -r .\deploy ec2-user@ec2-3-0-229-253.ap-southeast-1.compute.amazonaws.com:/home/ec2-user

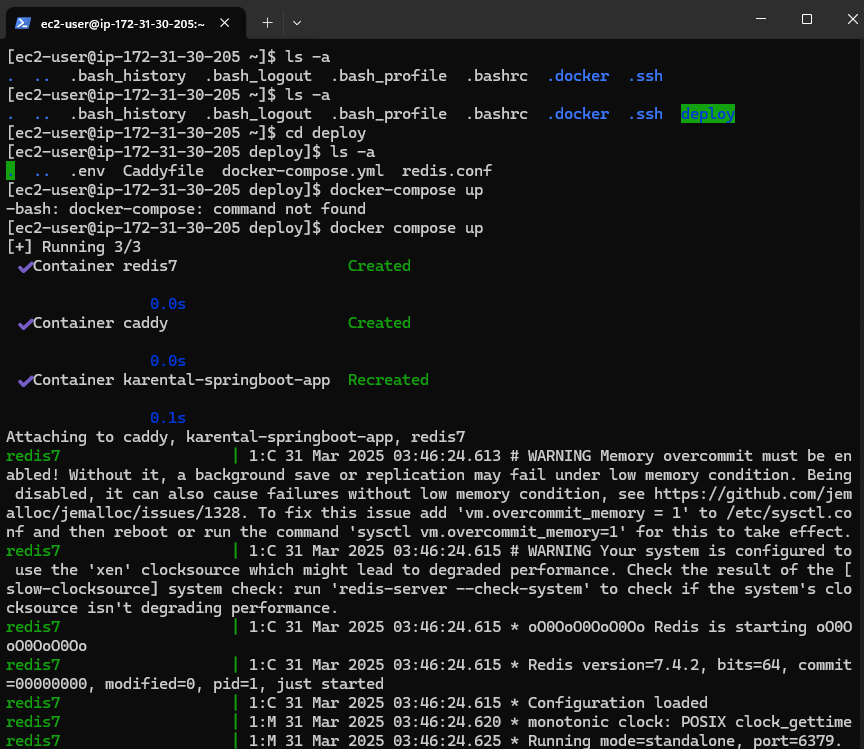
* -i <your-private-key>
* -r: copy recursive, all the file in the folder would be upload
* .\deploy : the folder
* The rest is the destination folder on your server



Connect to EC2 instance again, and you would see the folder in there



Cd to deploy and $ docker compose up, then you got your app running



Now, access https://<your-domain-name>/karental/ to find out more.Frontend