AWS project - the Jedi Council and Mission Log

This is the Technical documentation for hosting a static website and a FullStack application with AWS services. The documentation contains two parts:

- Hosting static website with AWS S3 and CloudFront
- Hosting FullStack application with AWS services

This documentation assumes the reader is familiar with the AWS management console and has a general understanding of the AWS services.

Product and Project Overview

This Project has two goals:

1. Migrating and Hosting a static website on AWS
2. Migrating and Hosting a FullStack application on AWS

The main purpose of migrating the products to AWS is to utilize AWS' well-manages services as well as reduce the cost of Capital expenditure

> Notes: This product is under the scope of the free tier of AWS. However, some steps might fall out of the scope of the free tier. In such cases, there will be an extra note for the step.

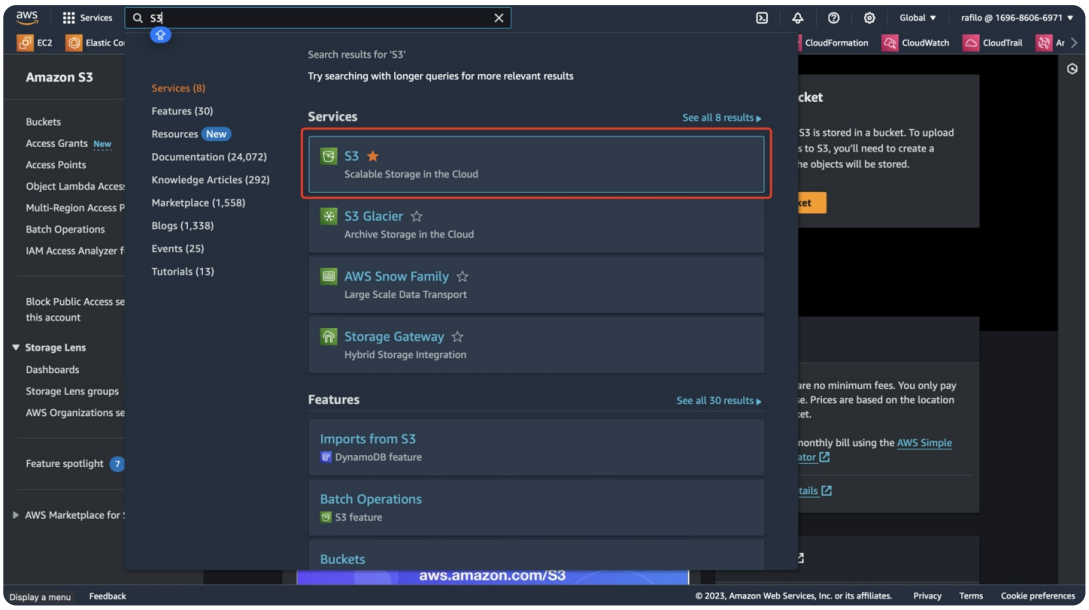
Hosting static website with AWS S3 and CloudFront

Hosting a Static Website in AWS consists of the following steps:

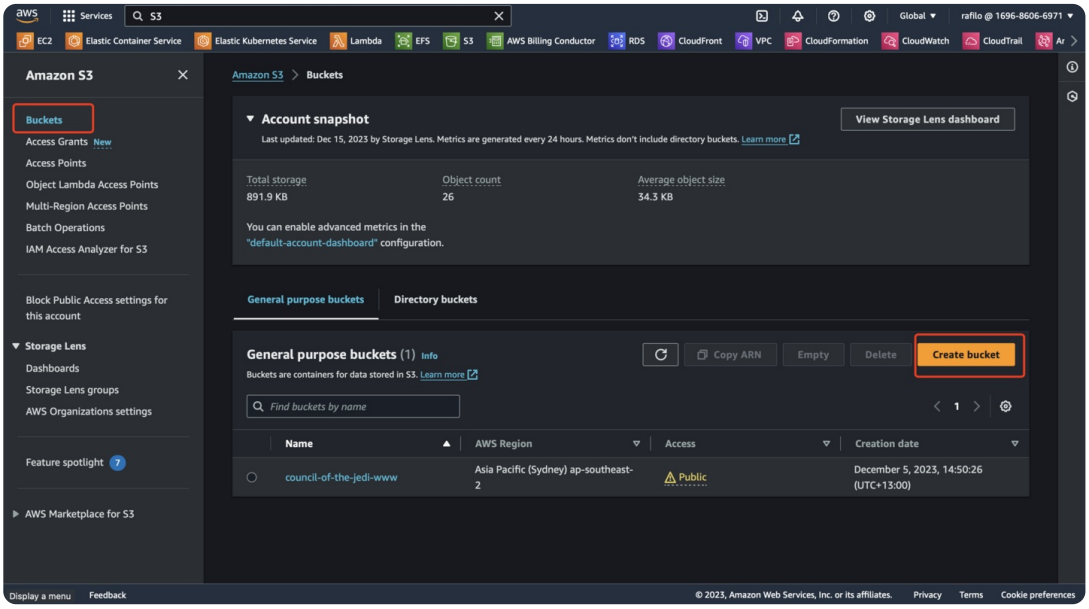
1. Create an S3 Bucket and Upload your static website to S3 Bucket
2. Hosting your static website with S3 Static website hosting service
3. Configure CloudFront for more convenient and faster access to your static website for people from all over the world

1. Create an S3 Bucket

1. Search "S3" on the navigation bar and navigate to S3 panel

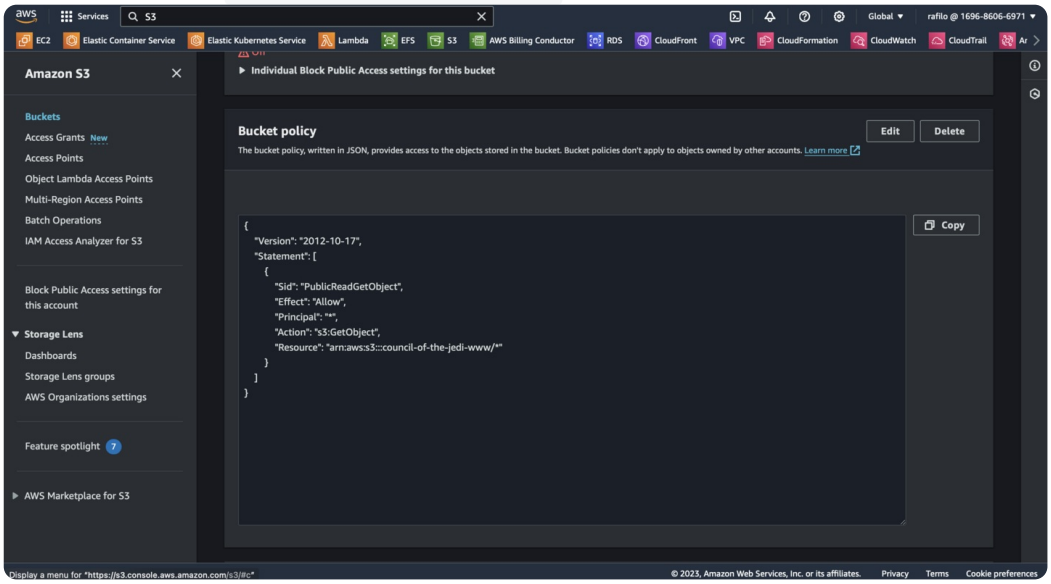


2. Navigate to Buckets, then click "Create Bucket"

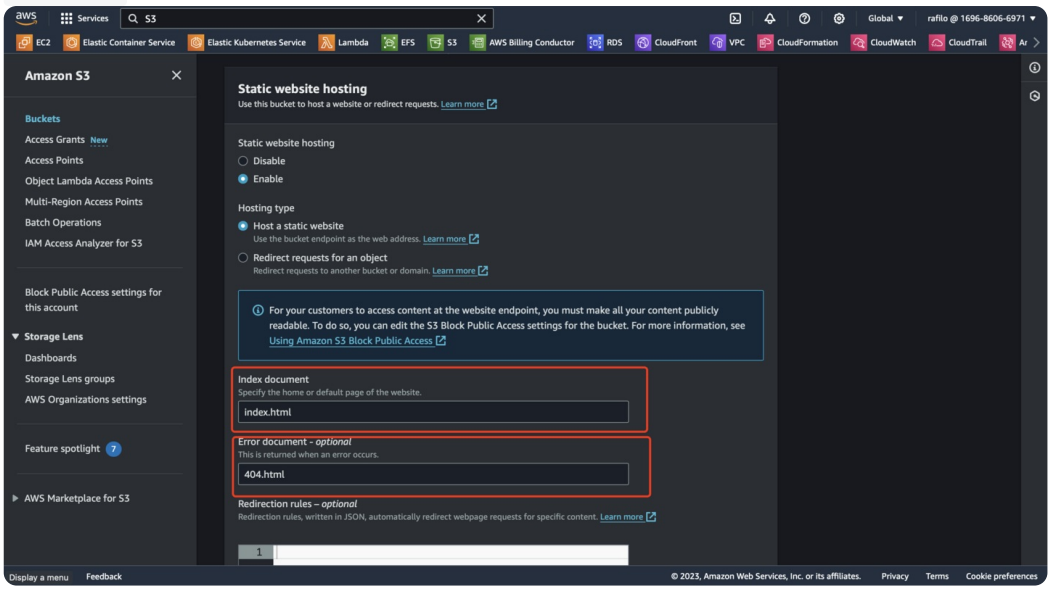


3. In the Create Bucket page
    1. Enter the bucket name
    2. In "**Object Ownership**" part, choose "**ACLs disabled (recommended)**"
    3. In "**Block Public Access settings for this bucket**" part, **uncheck** the box "Block all public access"
    4. keep other things default, then choose "Create bucket"
4. After the bucket is created, navigate to Buckets, and click the name of the bucket that is created
5. In the Object Panel, choose "Upload" to upload the static website files provided here
6. After uploading the files
    1. choose "Permissions" panel
    2. navigate to "Bucket Policy" part, click "Edit"

3. enter the following code in the text area, and replace <your bucket name> with your bucket name (e.g: `arn:aws:s3:::jedi-bucket`):

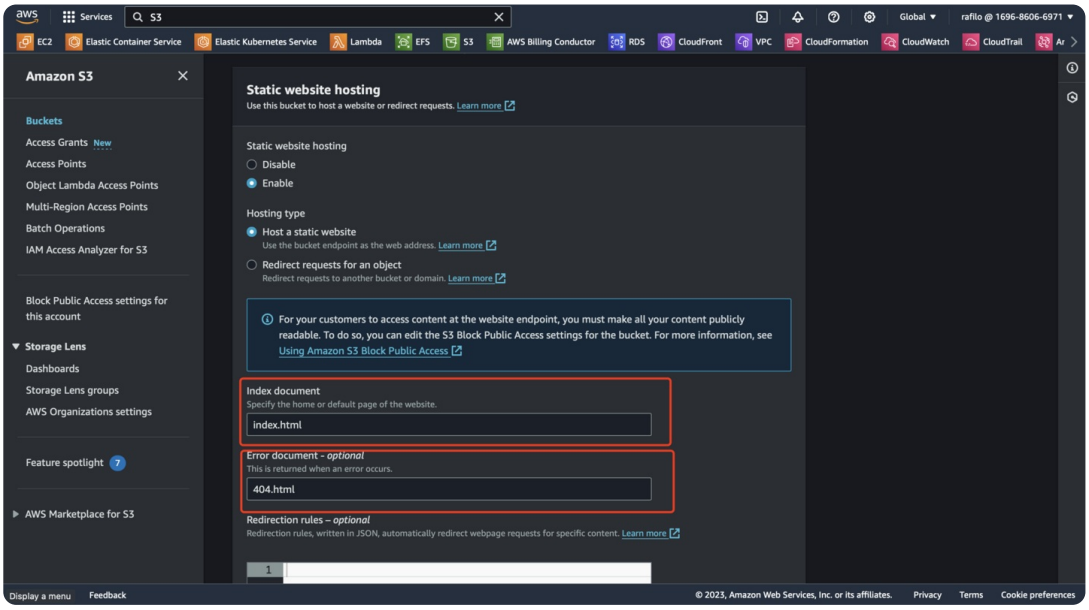



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<your bucket name>/*"
        }
    ]
}
```

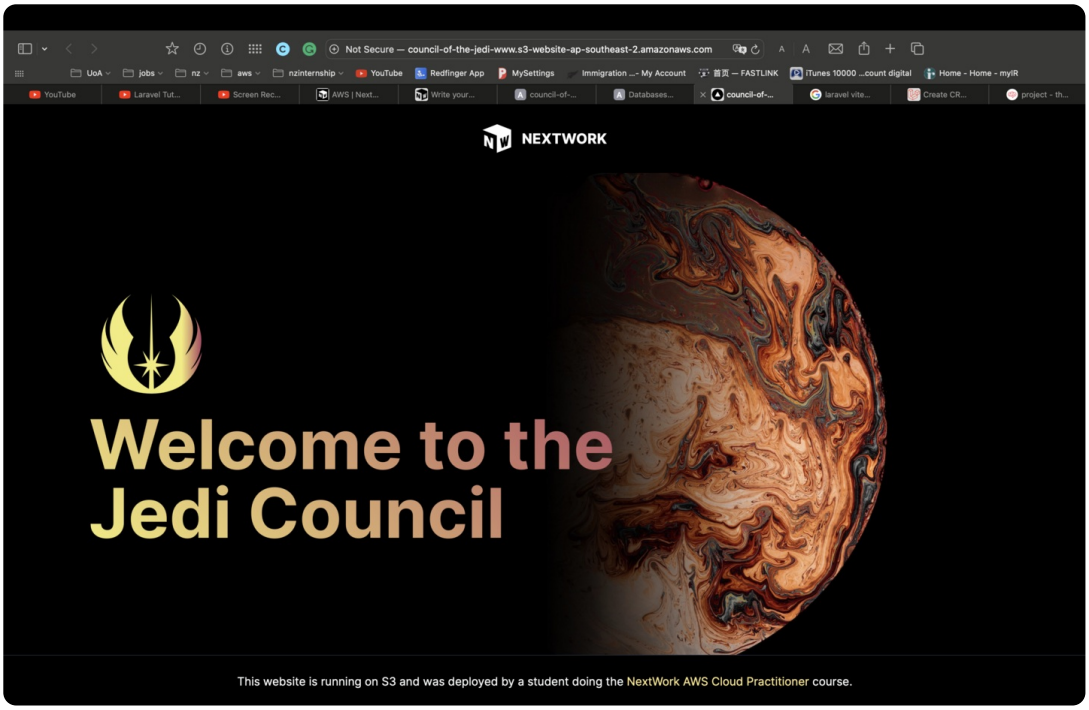
4. choose "Properties" panel
5. navigate to "Static Website Hosting" part, click "Edit"
6. On the page, specify "Index document" as `index.html` and "Error document" as `404.html`
7. 

Click "Save Changes"

7. Now you'll be able to access the website through the URL displayed in "Properties" → "Static Website Hosting" part



8. Enters the Jedi Council.....



Extra notes: Difference between ACLs and bucket policies

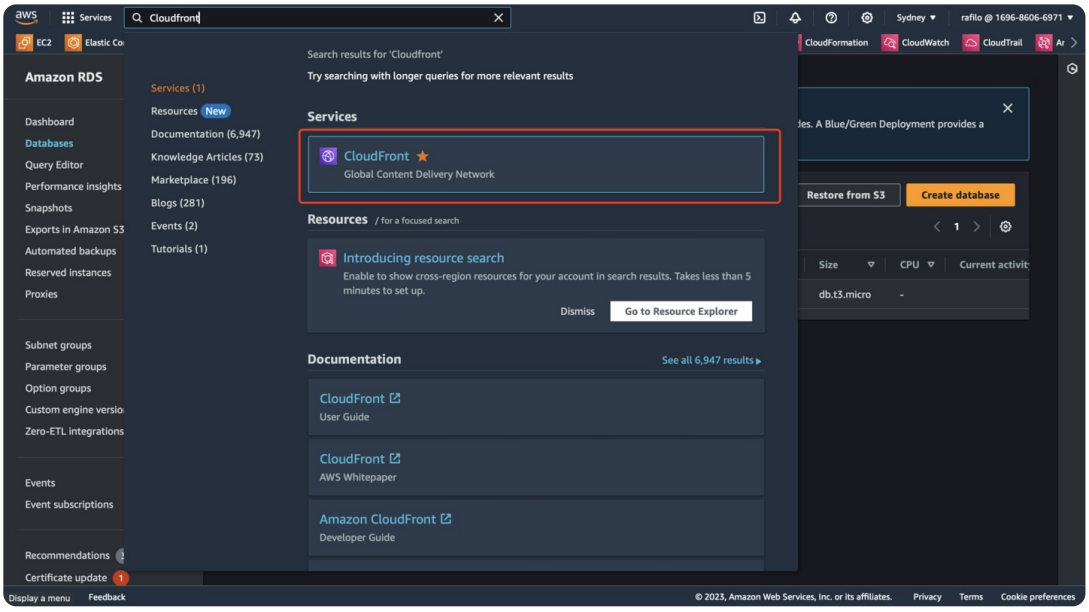
Access Control Lists (ACLs) are legacy (but not deprecated), and bucket/IAM policies are recommended by AWS.

ACLs give control over buckets AND objects, policies are only at the bucket level

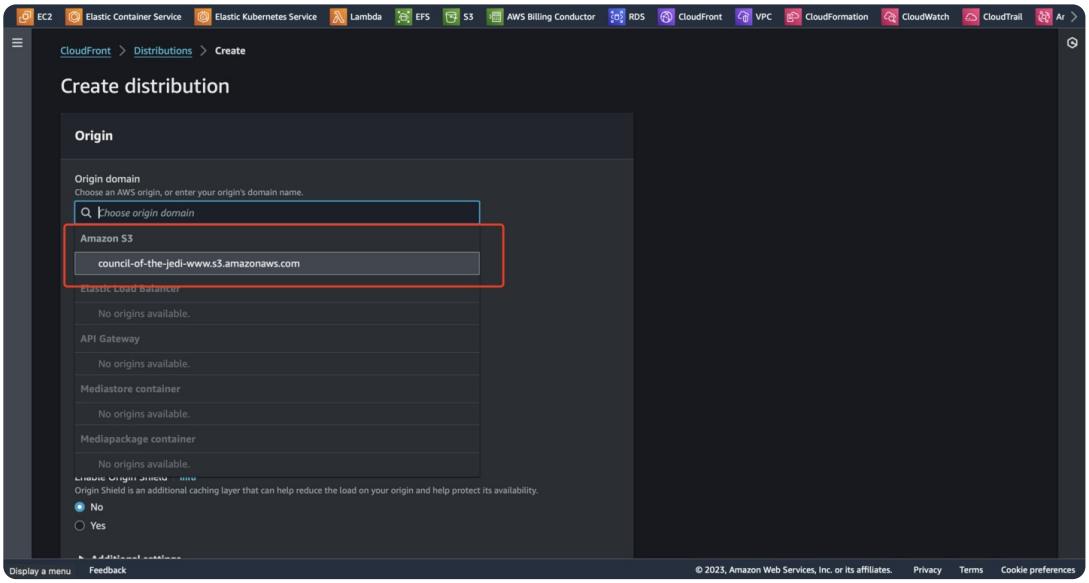
Use ACLs when objects are not owned by the bucket owner, need to manage permissions at the object level, and control only object-level permissions. In other cases use bucket policies.

2. Create a CloudFront Distribution

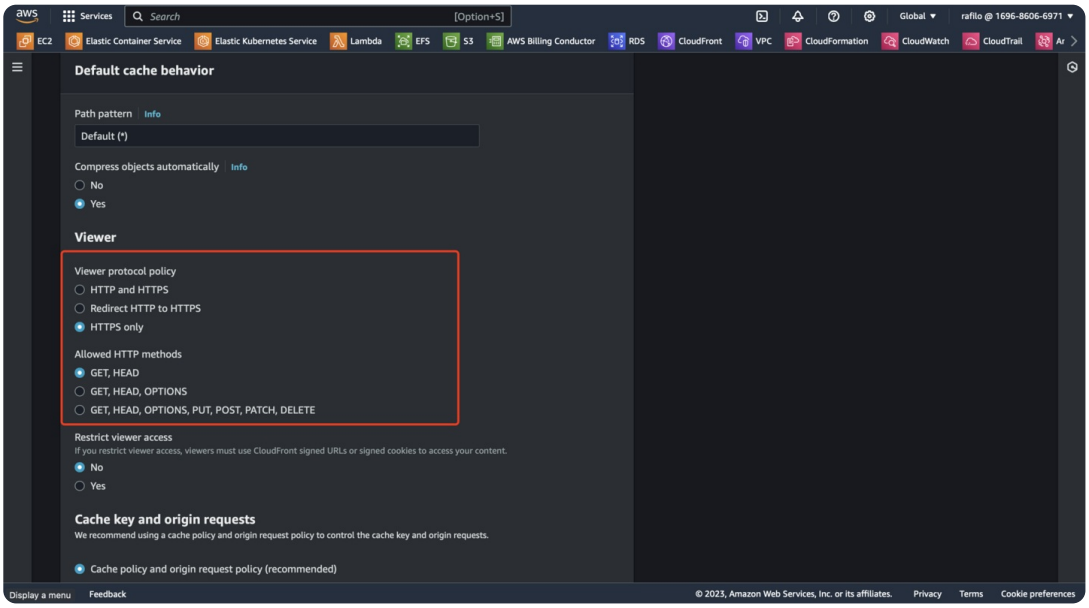
1. Search "CloudFront" on the navigation bar and navigate to CloudFront panel



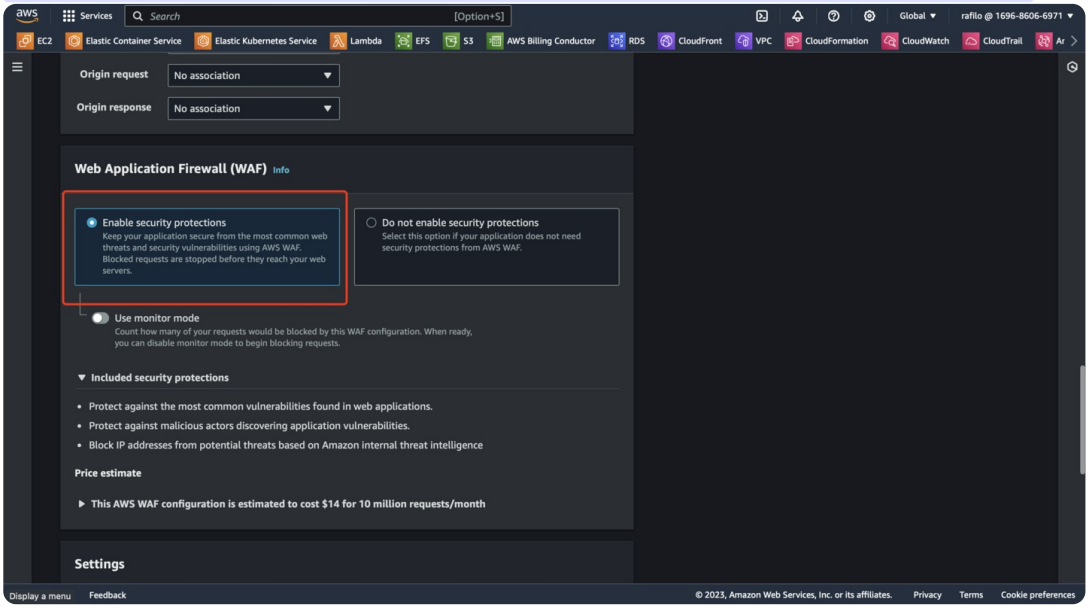
2. Navigate to "Distributions", then choose "Create Distribution"
3. In the Create Distribution page, navigate to "origin" part
4. Click the textbox under "Origin domain", select your S3 bucket



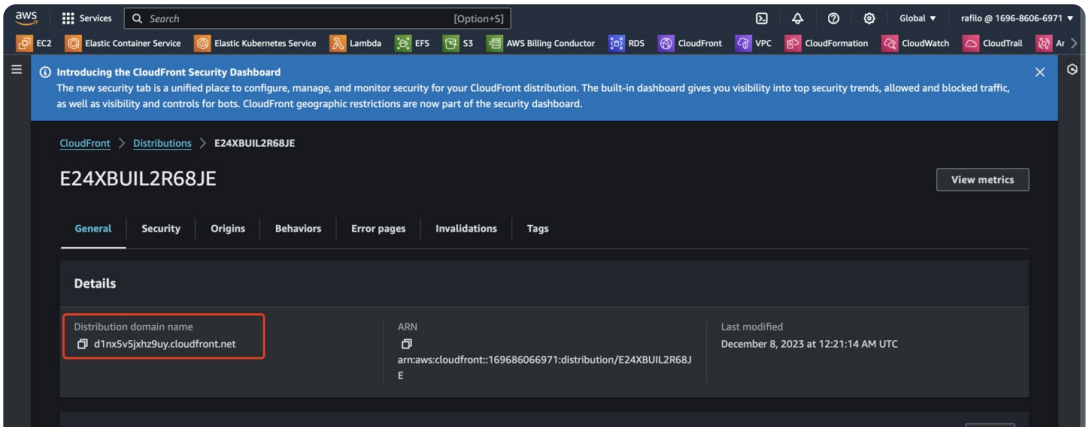
5. Navigate to "Default Cache Behavior" part. In "Viewer" section, select "HTTPS only" for Viewer protocol policy, and select "GET, HEAD" for Allowed HTTP methods

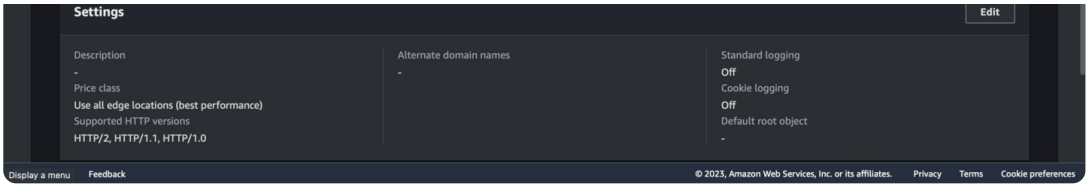


6. Navigate to "Web Application Firewall (WAF)" part, select "Enable Security protections



7. Keep other things default, choose "Create distribution" at the bottom of the page
8. After the distribution is created, click the name of your distribution. You can see the URL of your static website generated by CloudFront. Copy the URL to your browser and open it





9. Enters the Jedi council powered by CloudFront...

> ℹ️ Note: In step 4 above, you can enable S3 endpoint for a more secure connection

> ⚠️ **Attention: Enabling WAF for hosting static websites in Step 6 will cause extra costs, feel free to skip this step if you want to stay within free tier services.**

Hosting FullStack application with AWS

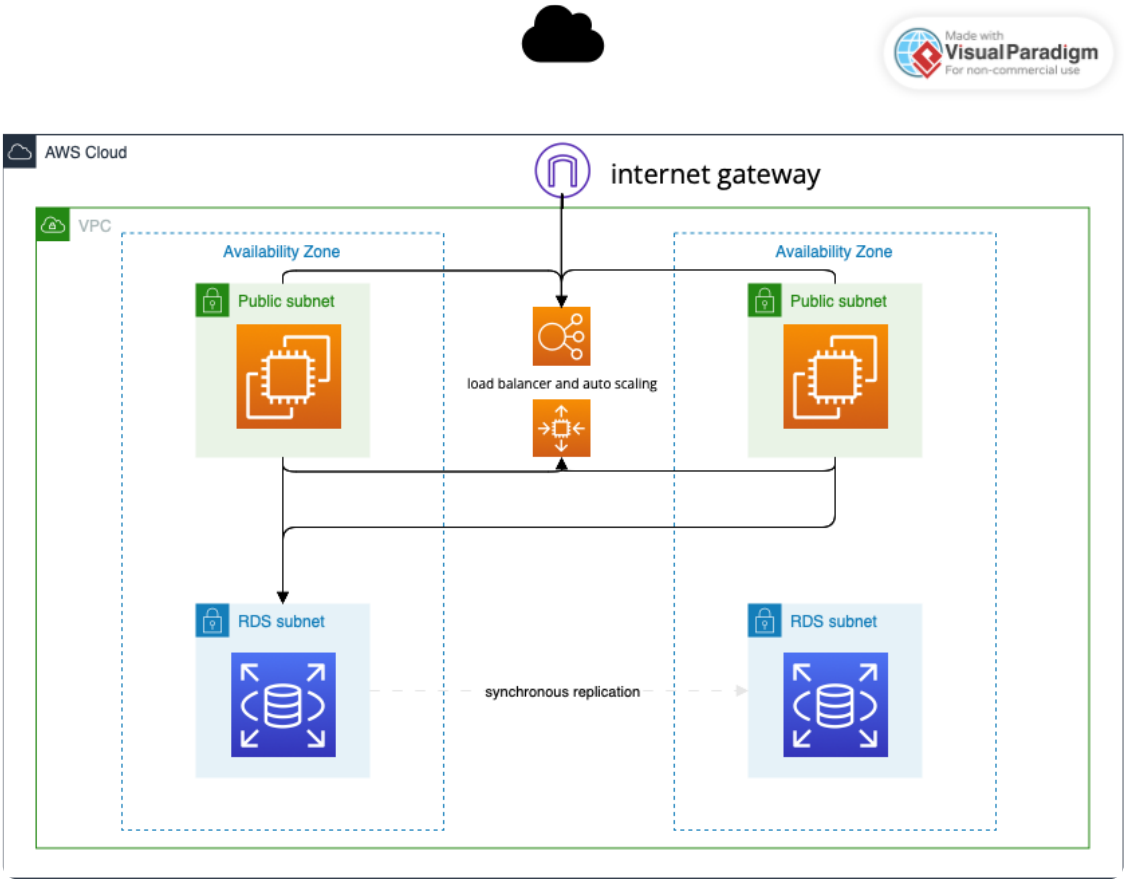
Hosting a FullStack application with AWS is more complicated compared to hosting a static website. It consists of the following steps:

1. Create a public VPC with subnets
2. Create an EC2 instance to run the application
3. Create a PostgreSQL Database with RDS to store the data
4. Create an Auto Scaling Group to ensure the availability of the application

ⓘ Before processing the following steps, choose the availability zone that you prefer in advance

1. Create a public VPC with subnets

In this project, we are going to generate a VPC with **2 public subnets (for hosting EC2 instances) and 2 private subnets**, as shown in the picture below

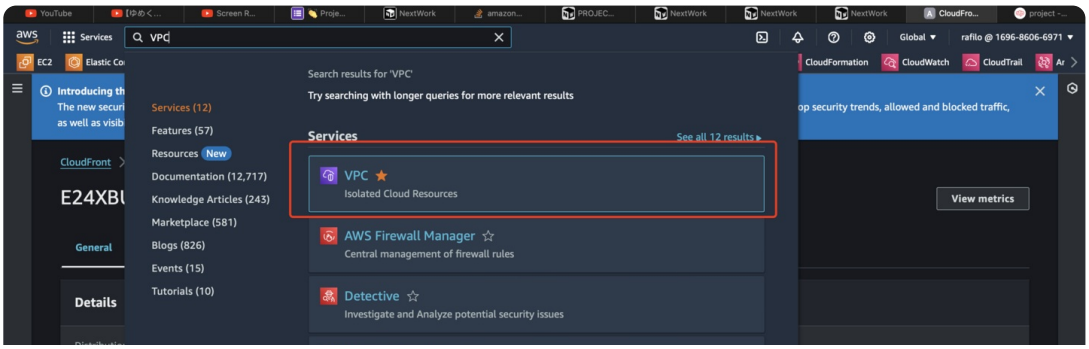


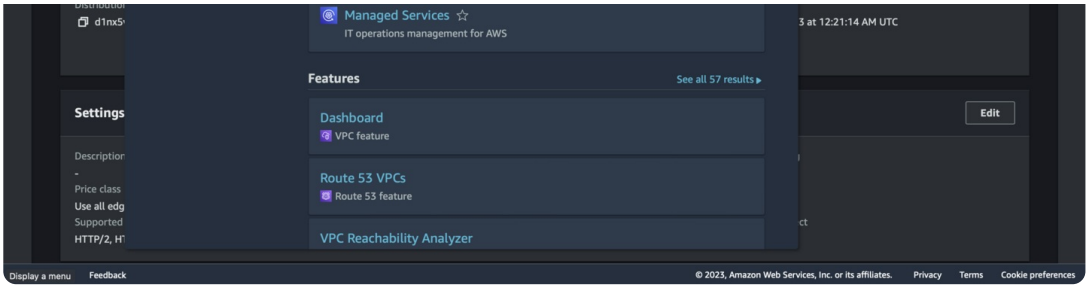
You may wonder... "Why do we need 2 public and private subnets and 2 availability zones?"

Amazon wants to ensure that your application has high availability, so they "suggest" that if you want to associate your VPC with other services, there have to be at least 2 public and private subnets as well as 2 availability zones (for both connecting EC2 instance to RDS and enabling auto-scaling group)

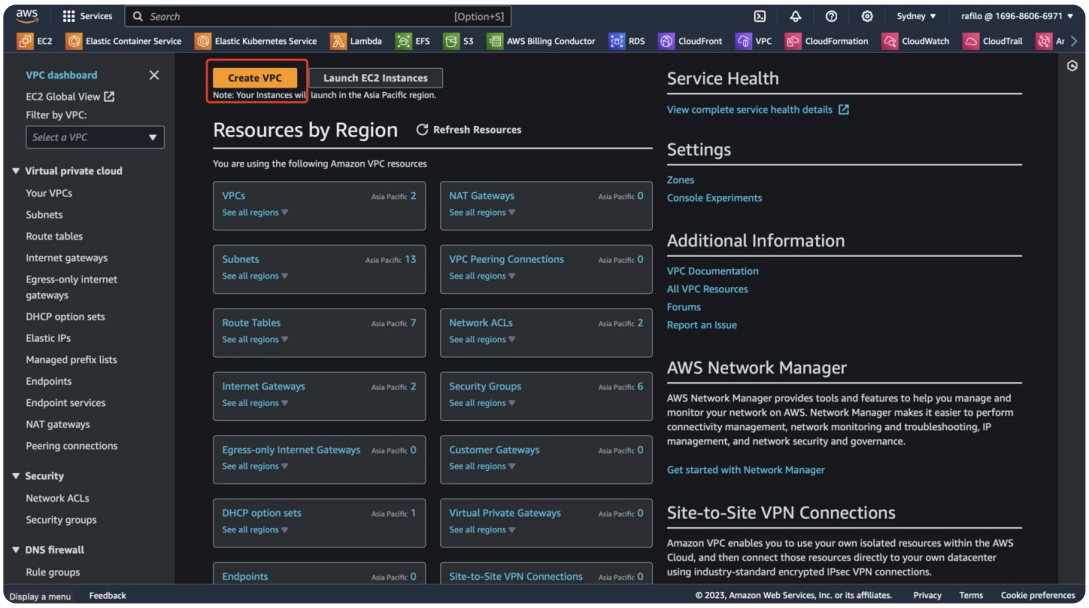
More details: https://stackoverflow.com/questions/52321470/ec2-load-balancer-at-least-two-subnets-must-be-specified

1. Search "VPC" on the navigation bar and navigate to the VPC panel

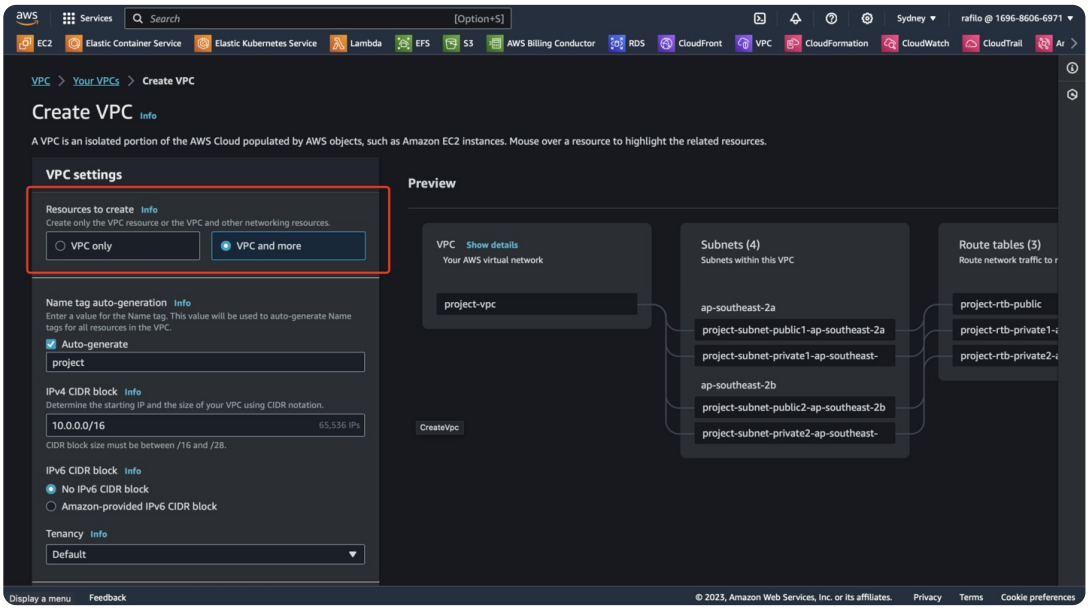




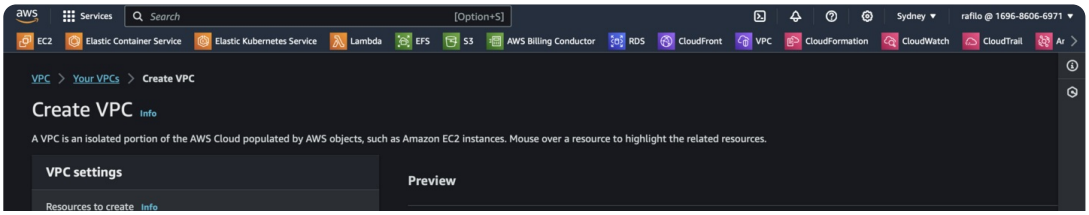
2. choose "Create VPC"

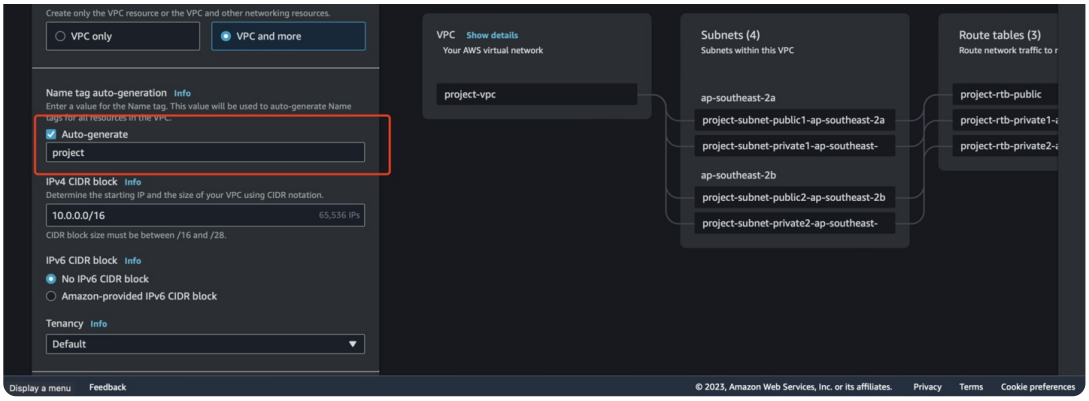


3. In Create VPC page, select "VPC and more" in "Resources to create" part



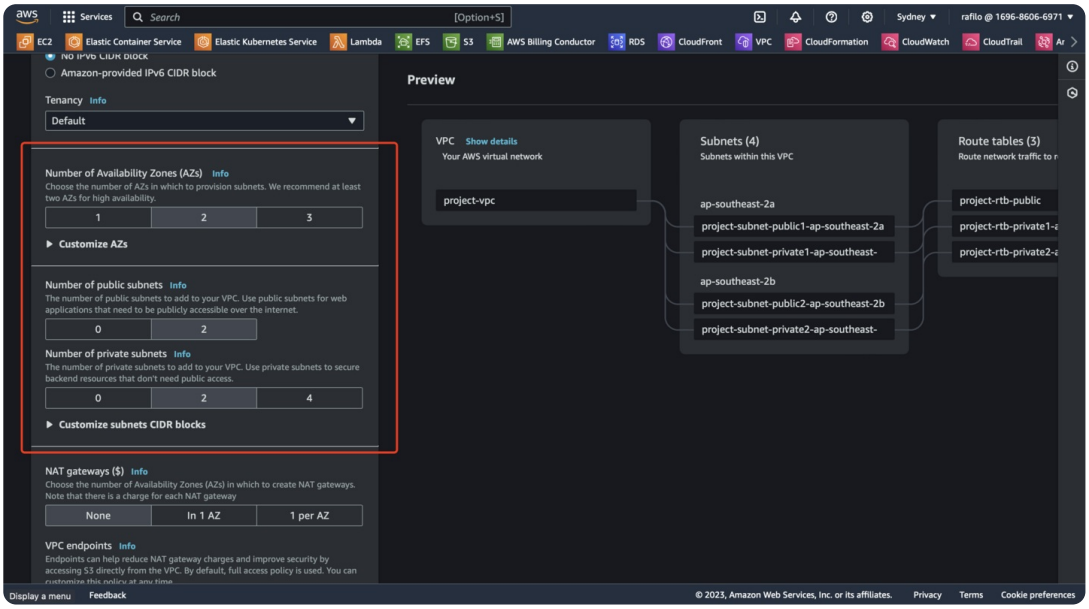
4. Enter the VPC name under the "Auto-generate" checkbox, and keep the "Auto-generate" box checked

5. 

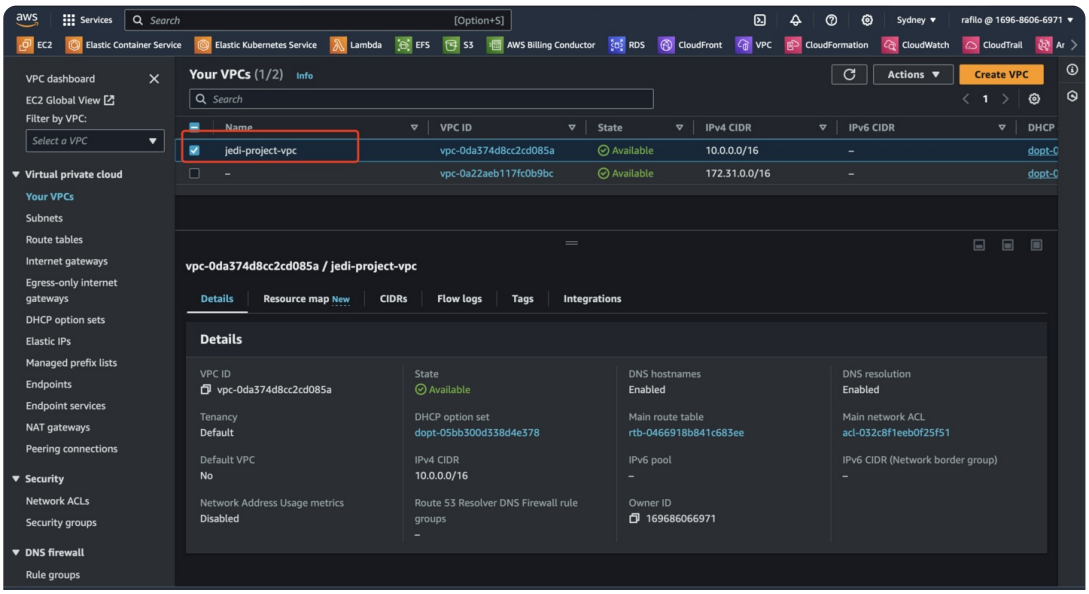


[Optional] If you want to configure the CIDR block to your desired IPs, modify the "IPv4 CIDR block" part

6. Navigate to "Number of Availability zones" and set it to 2. Same for "Number of public subnets" and "Number of private subnets"



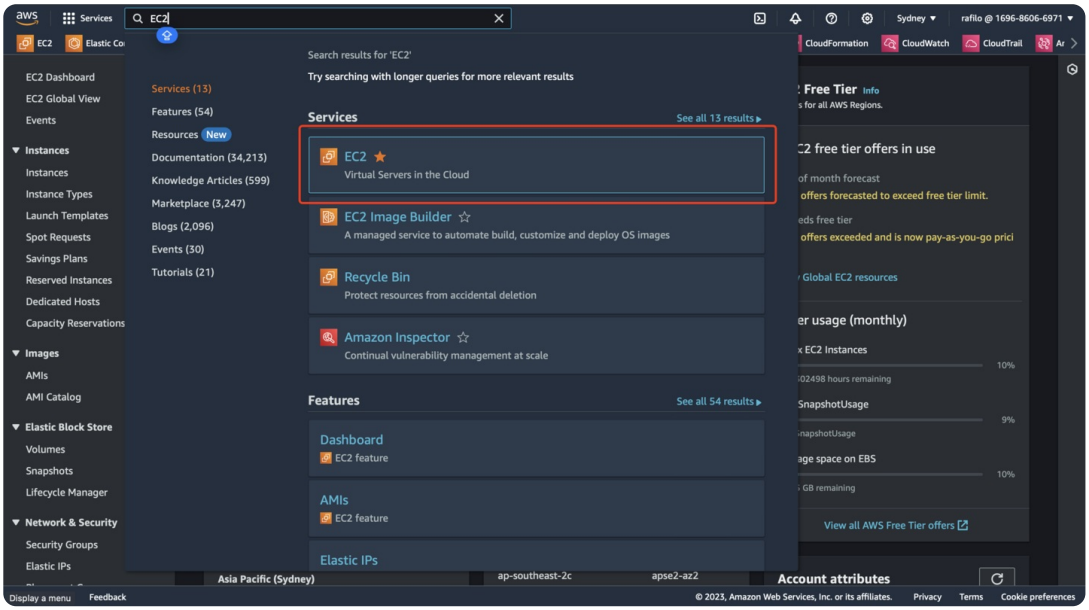
7. keep other settings default and choose "create VPC", it will take some time to automatically set up the VPC
8. Navigate to "Your VPC" panel, Now you should see your VPC in the dashboard



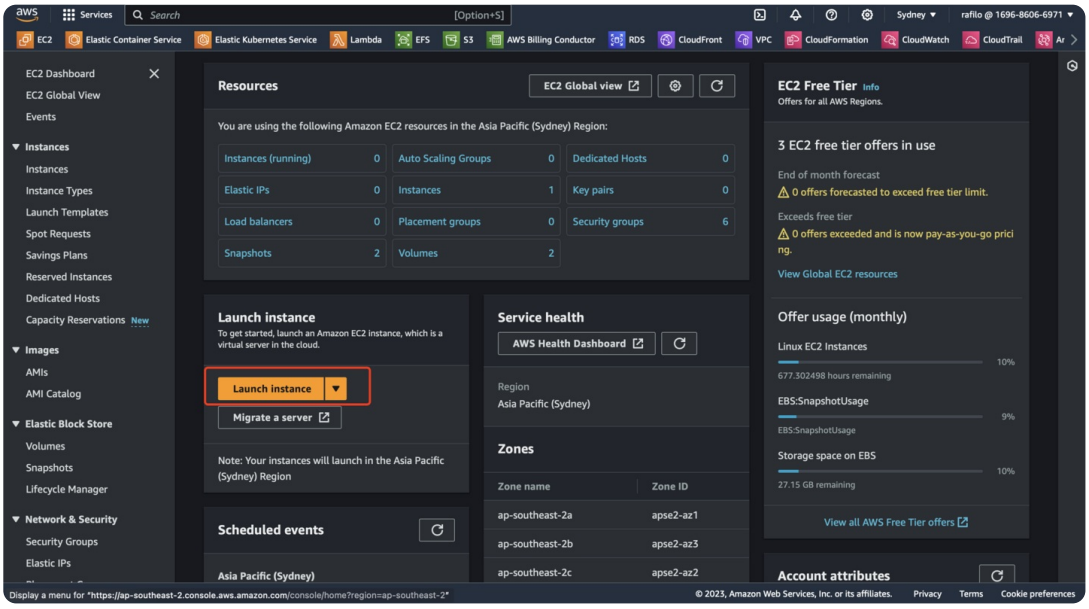
Setting up VPC with **"VPC and more"** options will save the time of creating subnets, internet gateways and routing tables for the VPC. If you want to have more controls when creating subnets and routing tables, you can choose **"VPC only"** and then manually attach subnets, internet gateways and routing tables to it. As in this project, since there is no convolute routing required, we can take advantage of using "VPC and more" to create the VPC.

2. Launch EC2 instance for hosting application in VPC

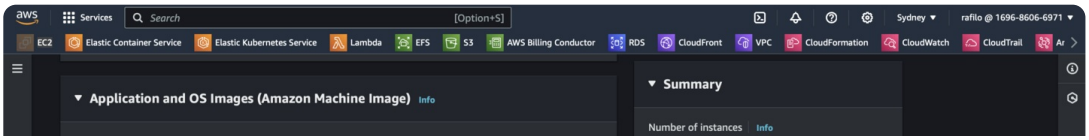
1. Search "EC2" on the navigation bar and navigate to the EC2 panel

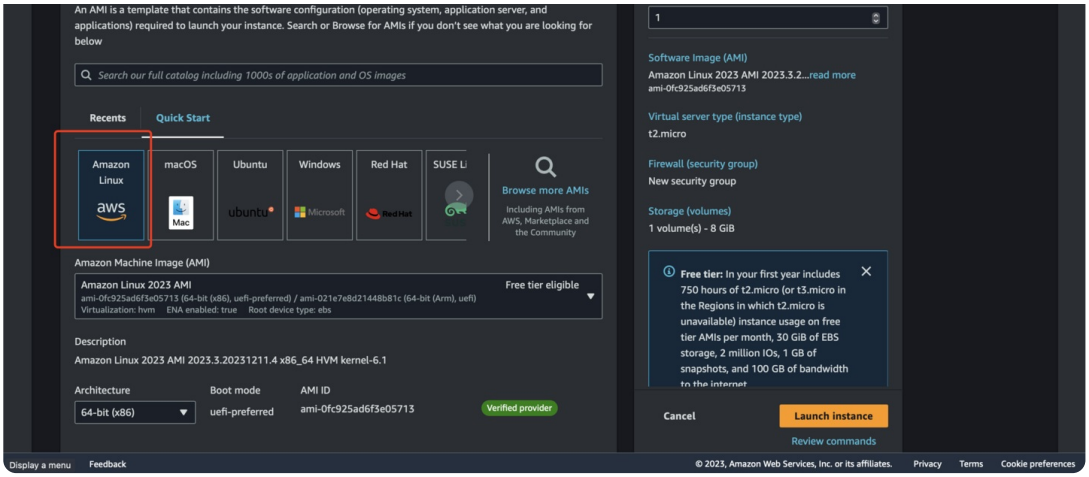


2. select "Launch Instance"

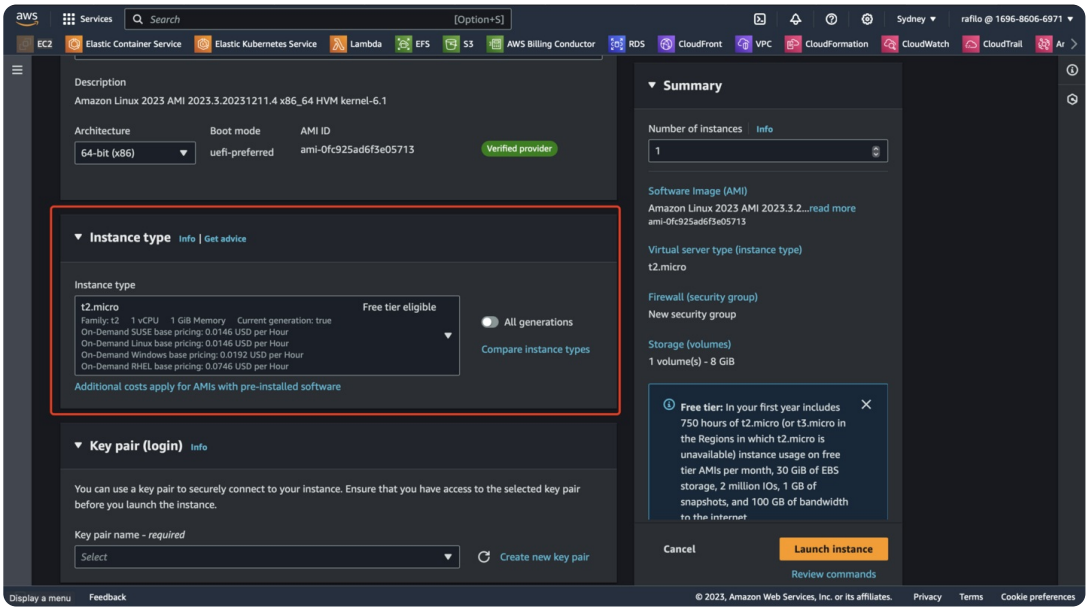


3. Enter the name of your EC2 instance in the "Name" part
4. Choose "Amazon Linux" in "Application and OS Images" part

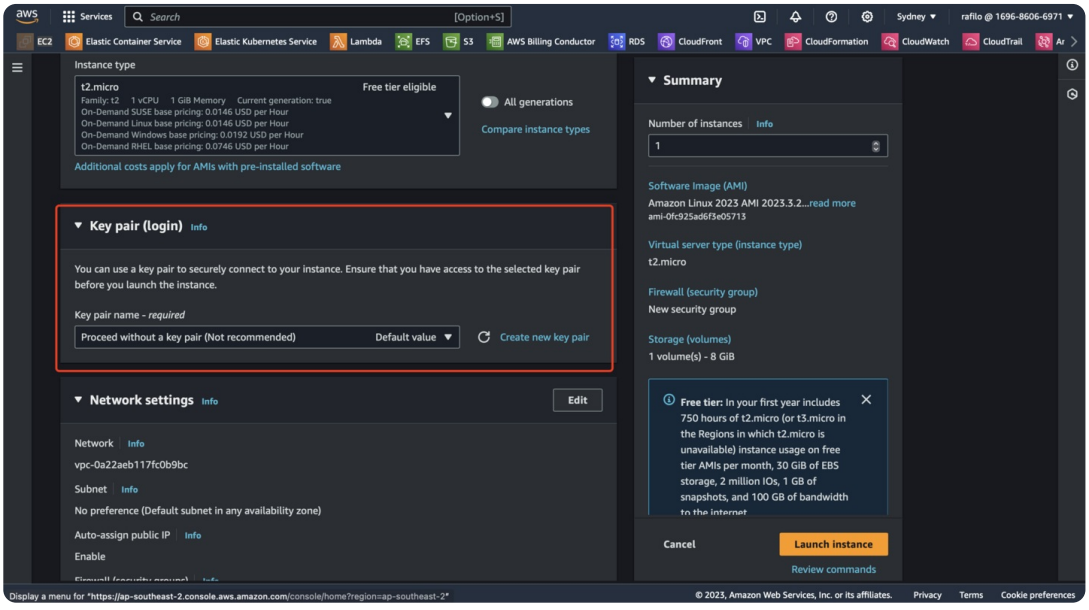




5. Choose "t2.micro" in "Instance type" part

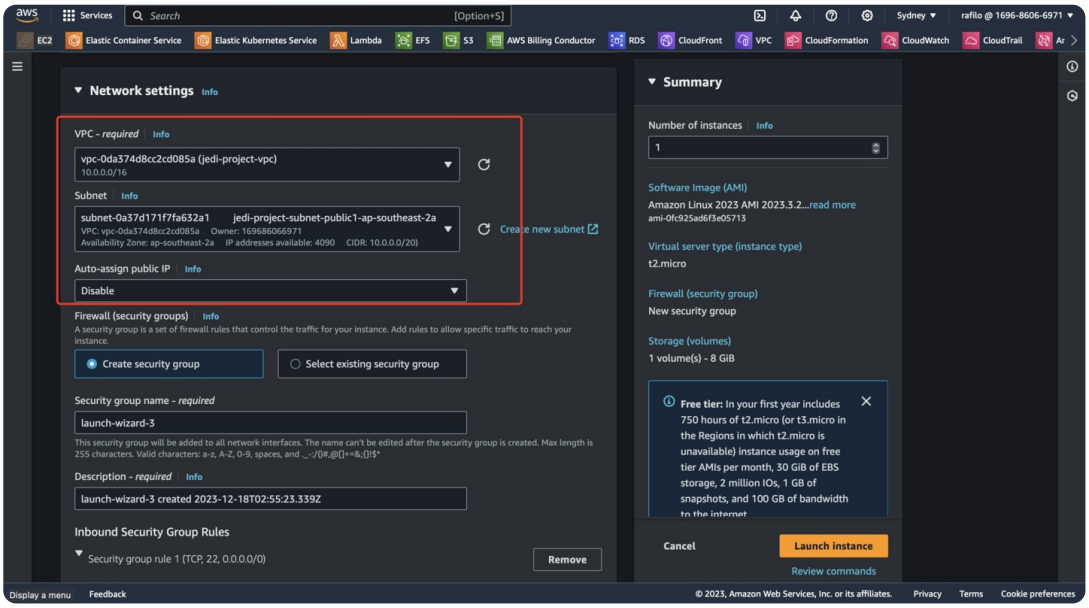


6. Choose "Proceed without a key pair (not recommended) in "Key pair (login)" part

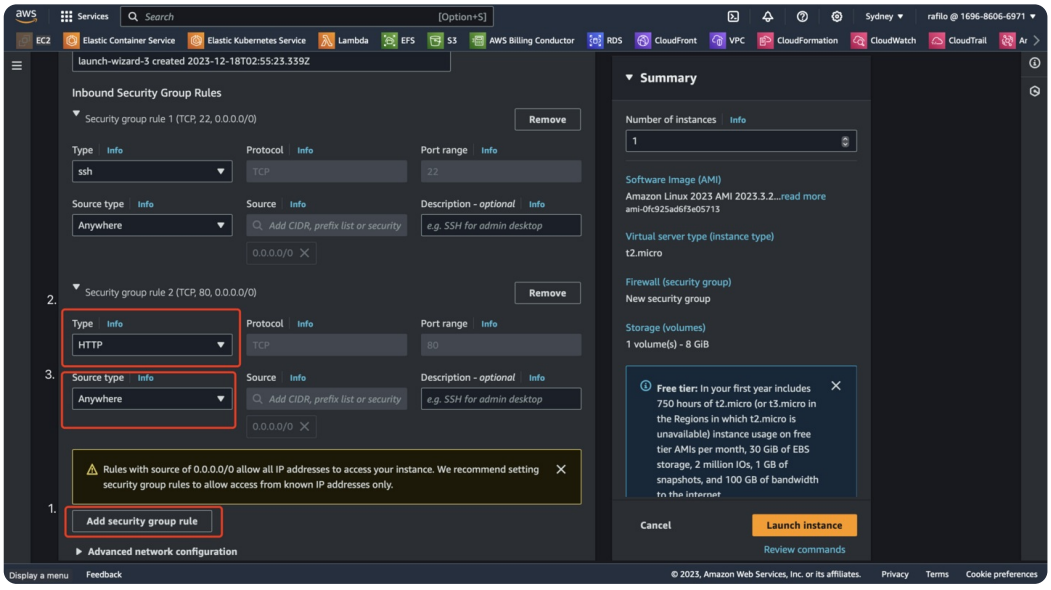


7. Locate to "Network Settings" part, click "Edit"
8. In "VPC-required" part, select the VPC you created in the previous step. In "Subnet" part,

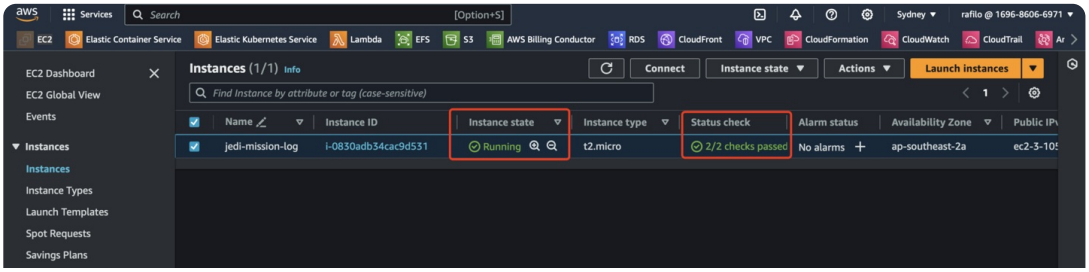
select one of the **public subnets** of your VPC. In "Auto-assign public IP" part, select "Enable"

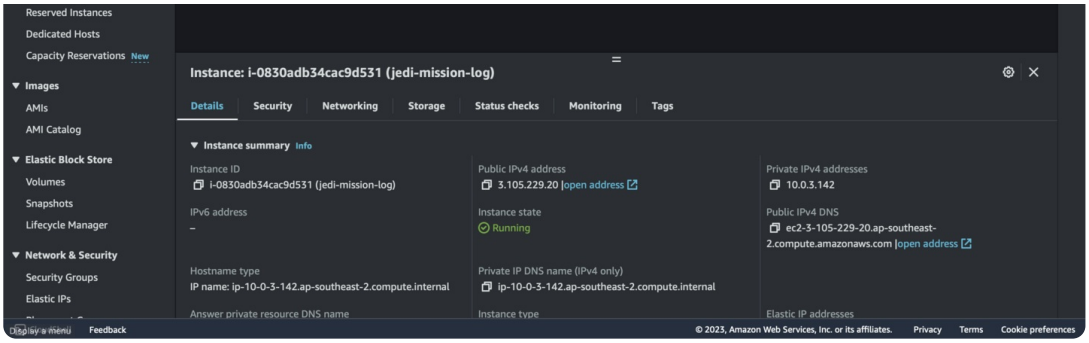


9. In "Inbound Security Group Rules":
    1. click "Add security group rule"
    2. Select "HTTP" for "type" in the newly created security group rule
    3. Select "Anywhere" for "Source type"



10. keep other fields default, then choose "Launch instance". It will take some time to launch the instance
11. Navigate to "Instance" from the navigation panel on the left, and wait until the Instance state becomes "running" and the status check is finished.





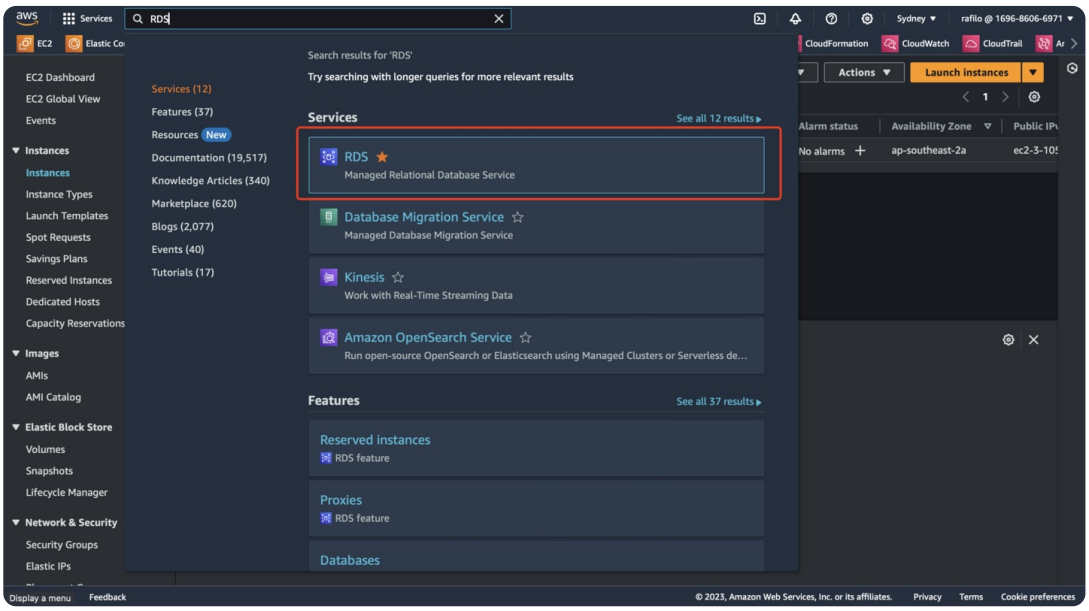
Notes:

1. we are not using the key pair for logging in because we are mainly utilizing the AWS Management panel to connect to our EC2 instance for this project. If you need to connect to the EC2 instance through SSH, then the keypair is required.

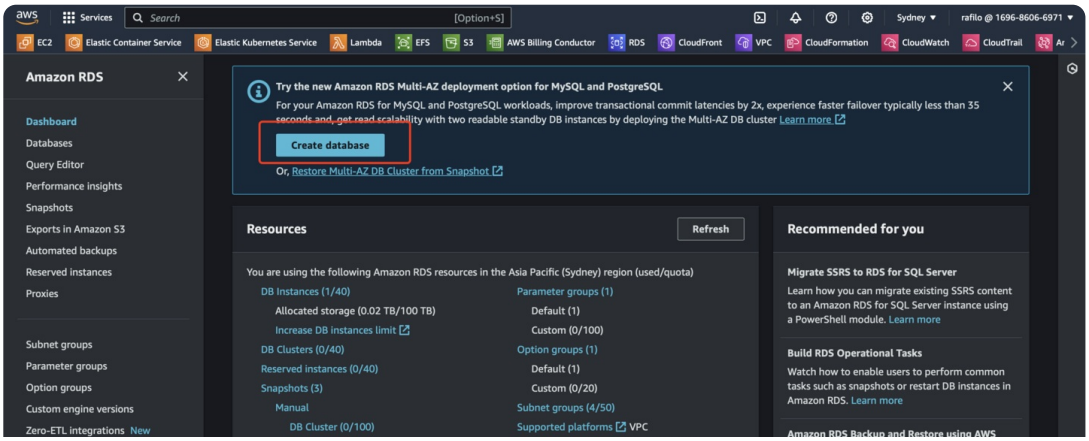
2. We are enabling auto-assign public IP for the instance since we are going to access our application directly through the public IP. In the real-world project, it'll be better to assign the application with a private IP and route it through your desired domain name.

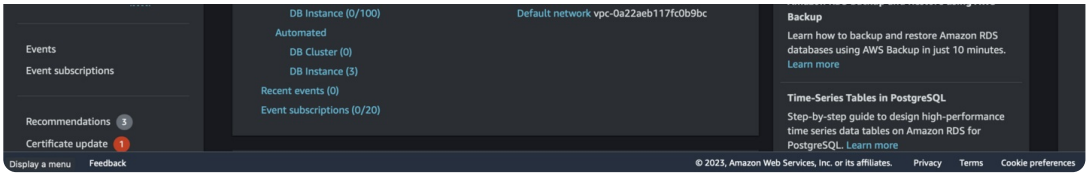
3. Create a PostgreSQL Database with RDS

1. Search "RDS" on the navigation bar and navigate to the RDS panel

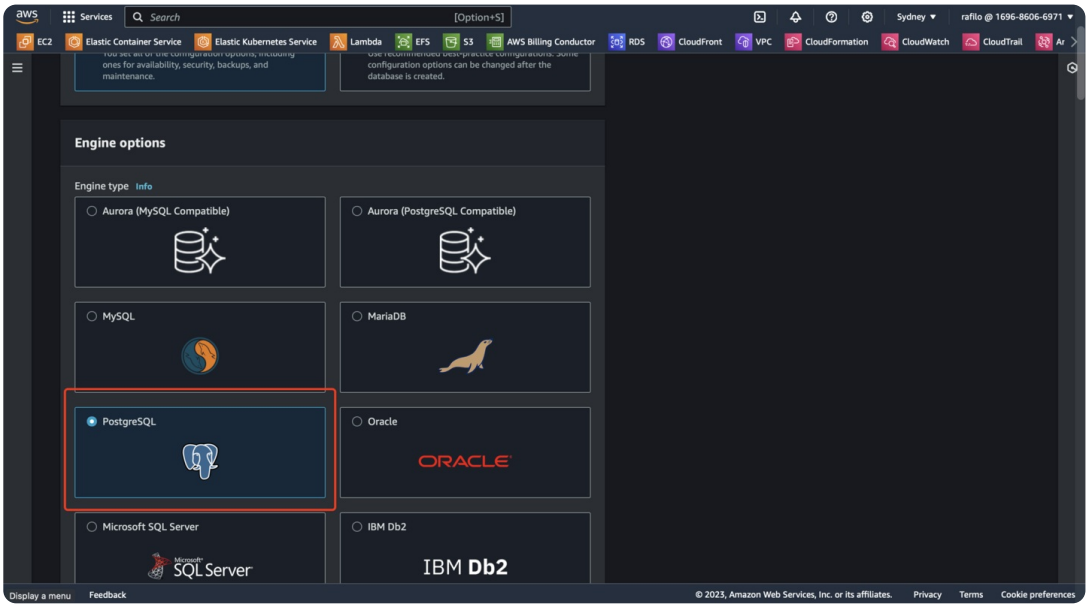


2. choose "Create Database"

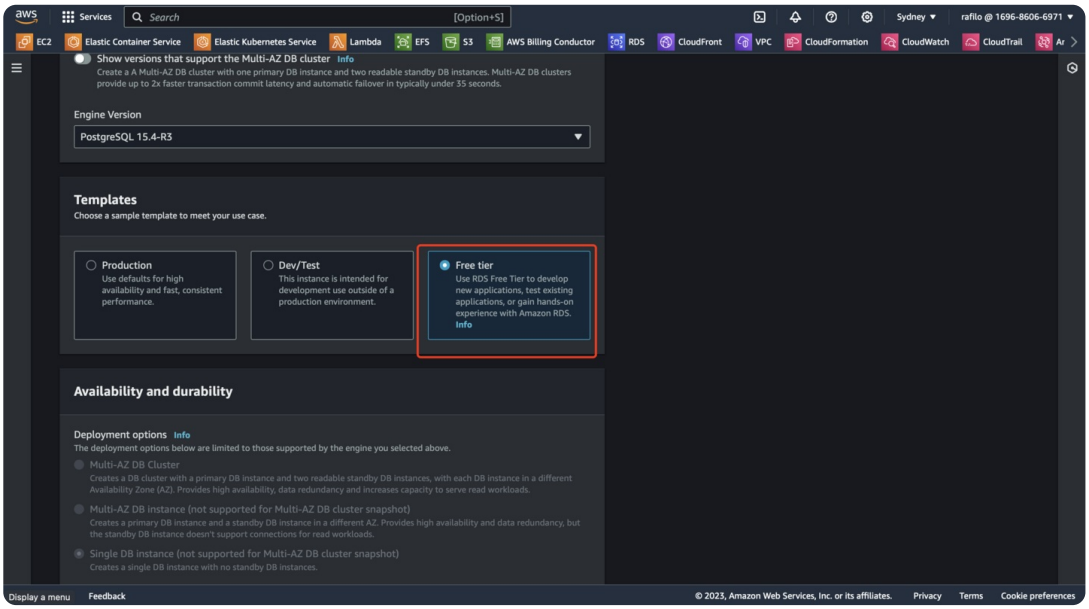




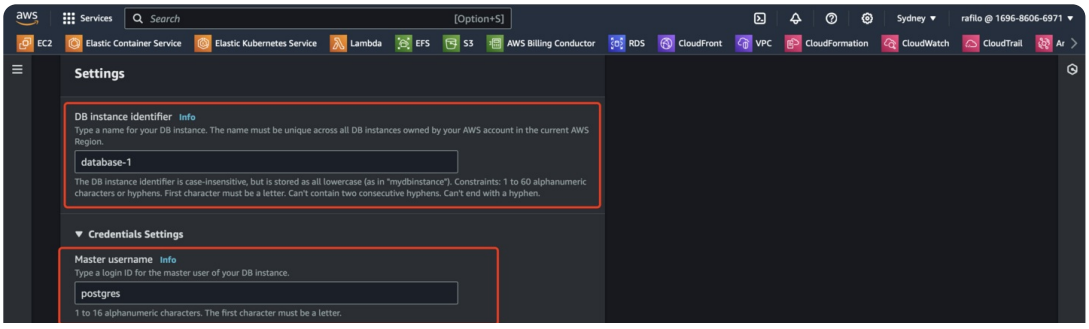
3. In the Create Database page, select PostgreSQL for "Engine type"

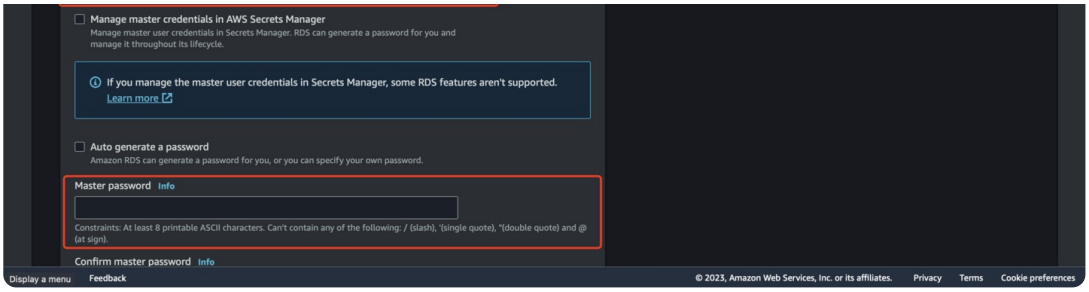


4. Navigate to "Template" part, select "Free tier"

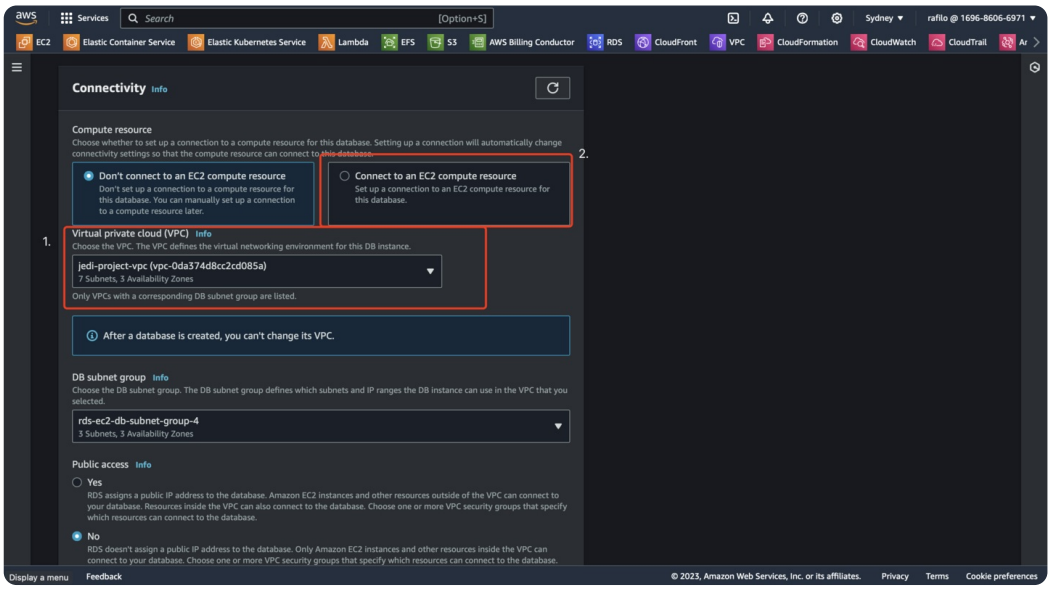


5. Navigate to "Settings" part, enter the database identifier name, the master username and password

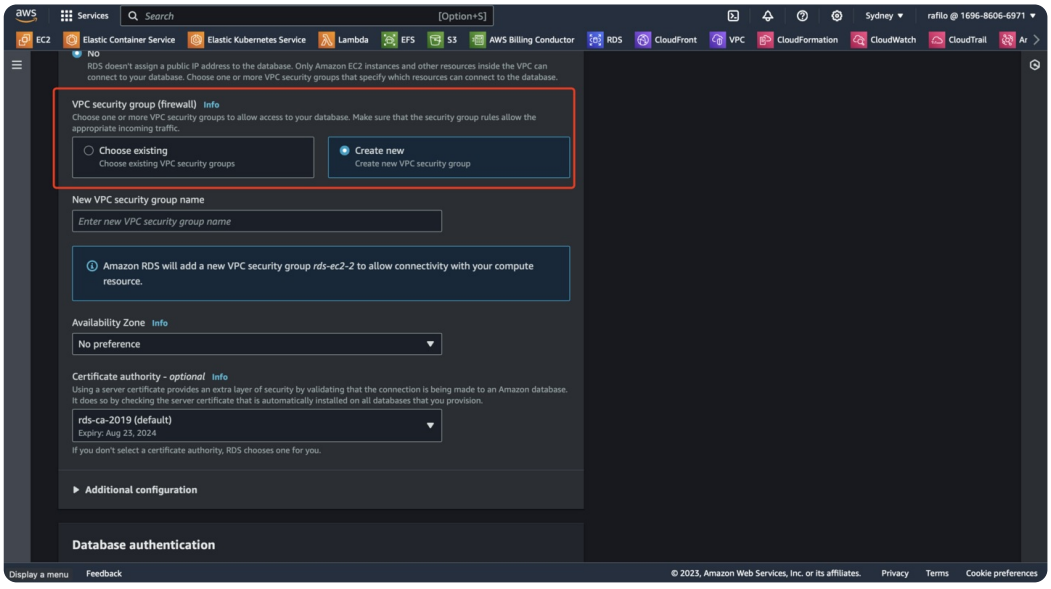




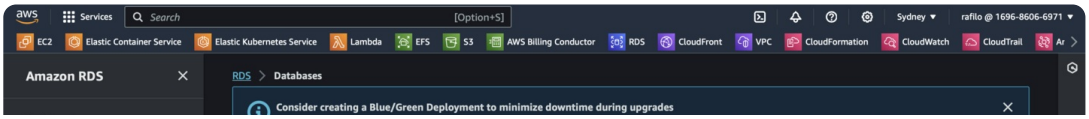
6. Navigate to "Connectivity" part:
    1. first select the VPC that you've created in the previous step
    2. then select "Connect to an EC2 resource" for "Compute resource" part

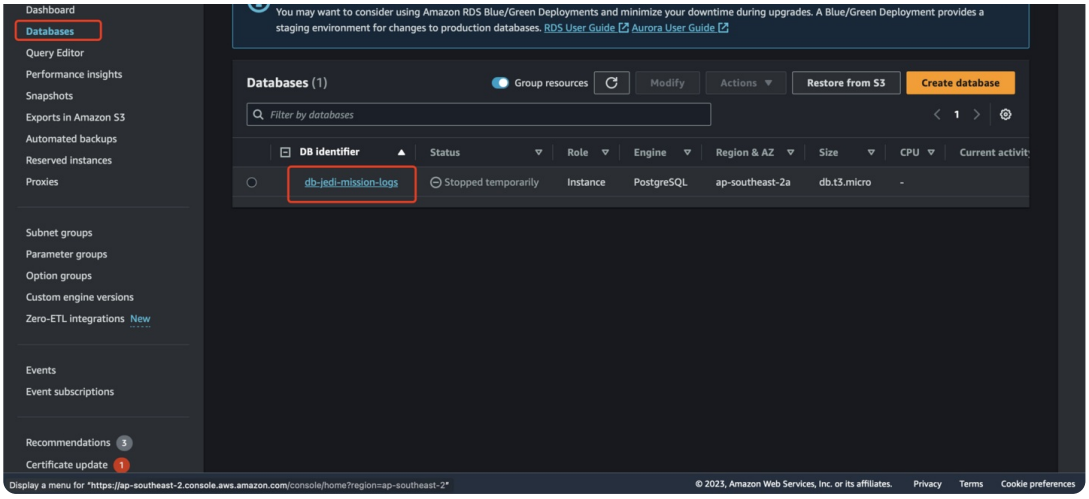


    3. navigate to "VPC security group" part, select "Create New"

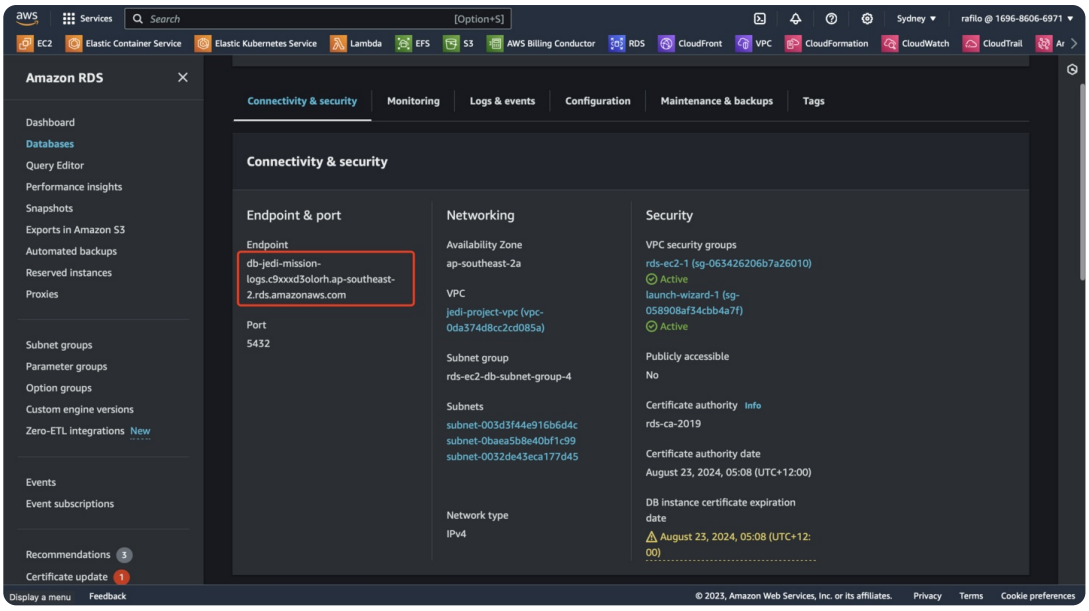


7. keep other settings default, then click "Create database" at the bottom of the page
8. It will take a while for AWS to set up the database. Navigate to "Databases" from the left menu panel, you should see your database in the dash board



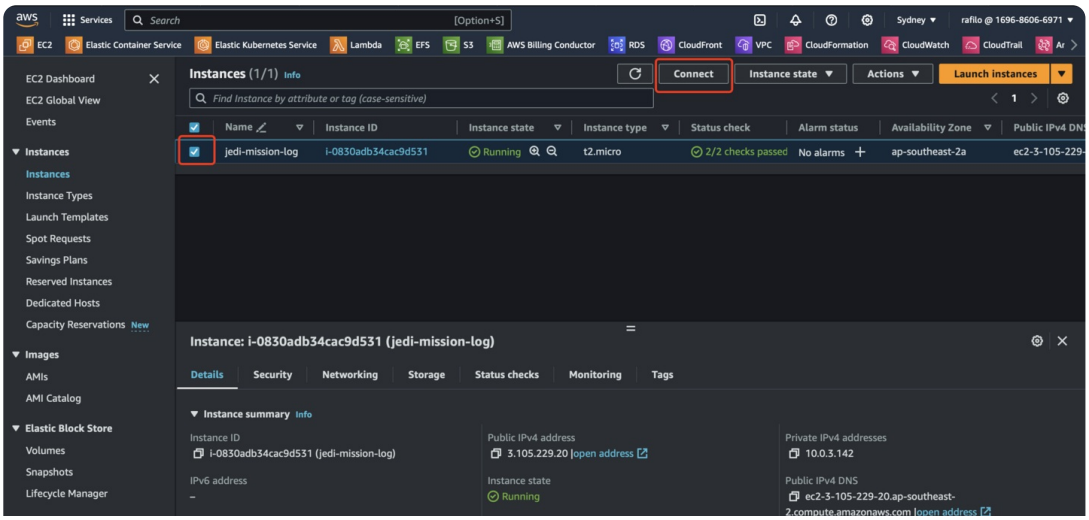


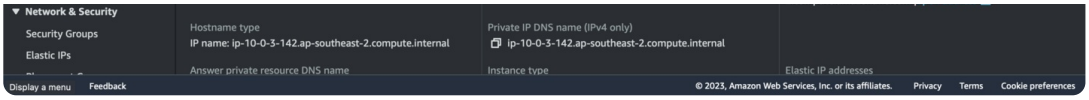
9. Click the name of your database, scroll down and choose "Connectivity & Security". You should see your database endpoint in the panel. Copy it down as we'll need it for our application to connect to our database



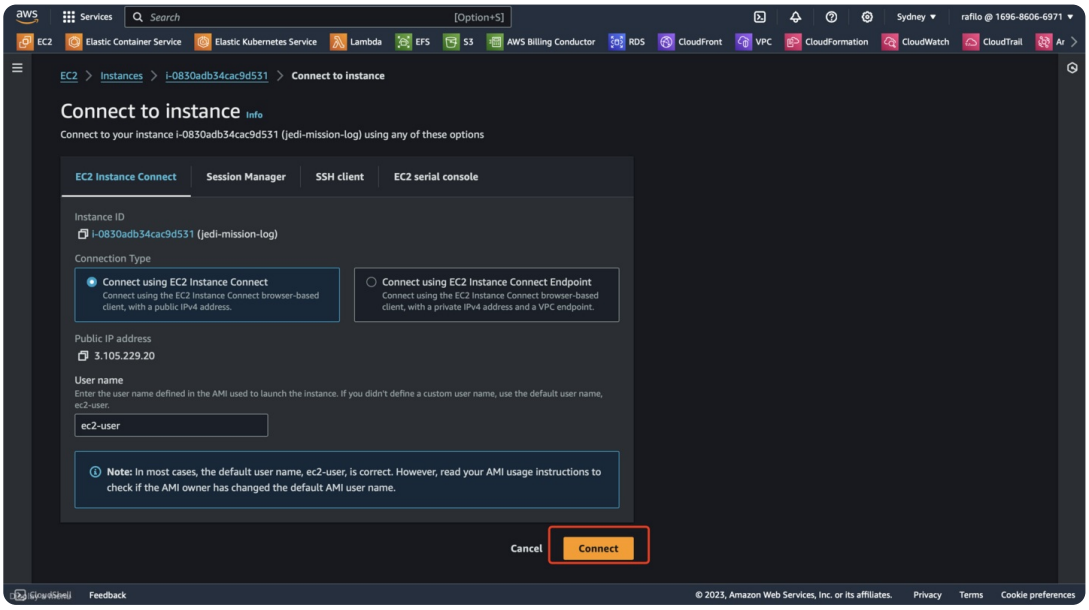
4. Install Docker to start the application

1. Navigate to EC2 panel, check the box beside your current running EC2 instance created in previous step, then click "Connect"

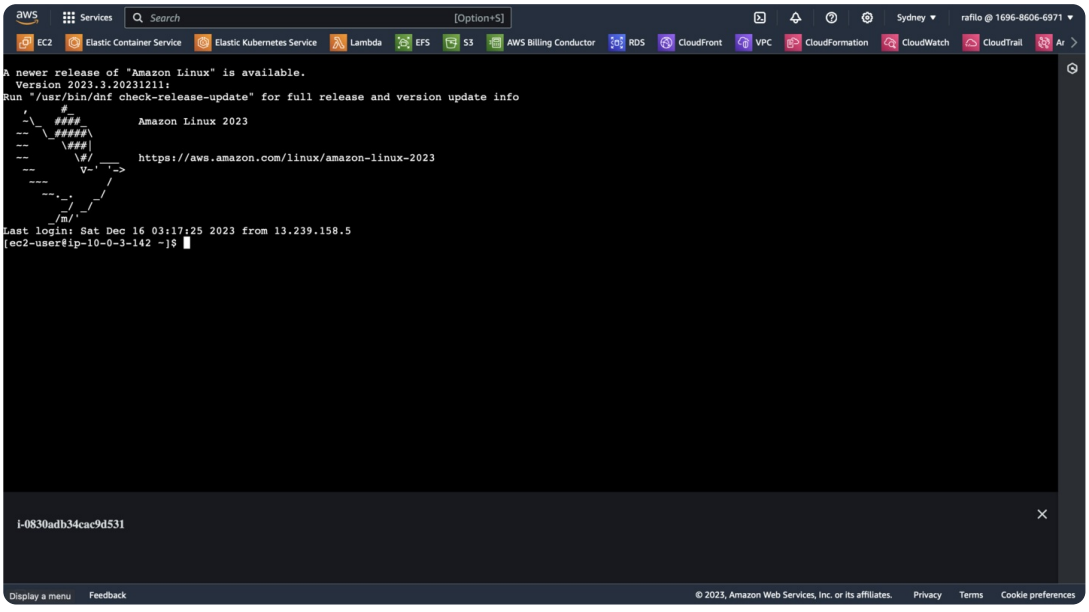




2. In the "Connect to instance" page, click "Connect", it'll open a new tab for your displaying your EC2 Instance



3. If the connection is successful, you should see the following shell in your tab. Otherwise, check whether you've enabled HTTP access for both your VPC and Instance



4. Now we need to install everything we need for our EC2 instance
    1. first, update the `yum` package manager and then install docker:

    ```
    sudo yum update
    sudo yum install docker
    ```

    2. then, start the docker service

    ```
    sudo usermod -a -G docker ec2-user
    sudo service docker start
    ```

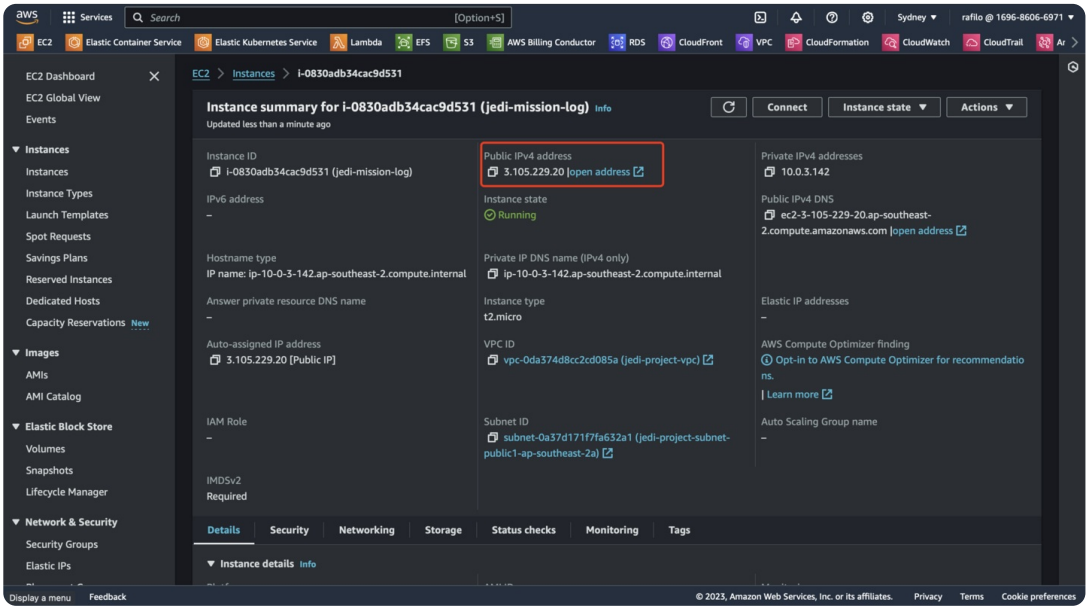
3. after starting up the docker service, pull the image of our application from docker repository

```
sudo docker pull public.ecr.aws/z7j4c9h0/nextwork/coursework/real-world-projects/jedi-mission-logs:latest
```

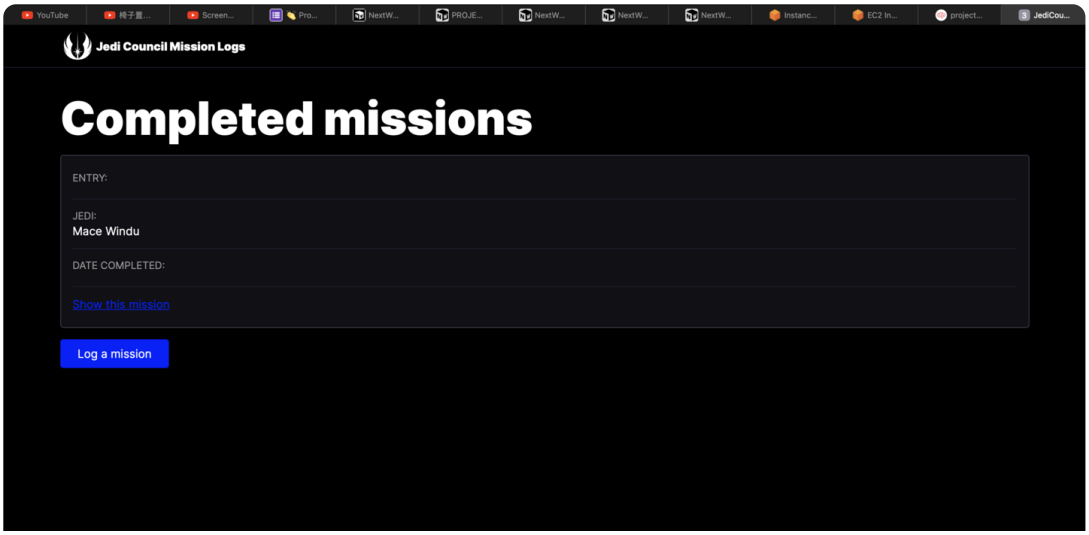
4. finally, start up our application and connect it with our RDS-managed database

```
docker run -d -p 80:3000 -e DATABASE_URL=postgres://<your username>:<your password here>@<DB instance URL here>:5432/jedi-mission-logs public.ecr.aws/z7j4c9h0/nextwork/coursework/real-world-projects/jedi-mission-logs:latest
```

5. In the last command, replace **<your username>** with your database master username that you entered when creating the database in RDS, replace **<your password here>** with your database password, and replace **<DB instance URL here>** with your database endpoint
6. Navigate back to your EC2 Instance, and click your EC2 Instance ID.
7. In the Instance summary page, copy the public IP for your EC2 Instance and open it in another tab



8. Enters the Jedi Council Mission logs panel...

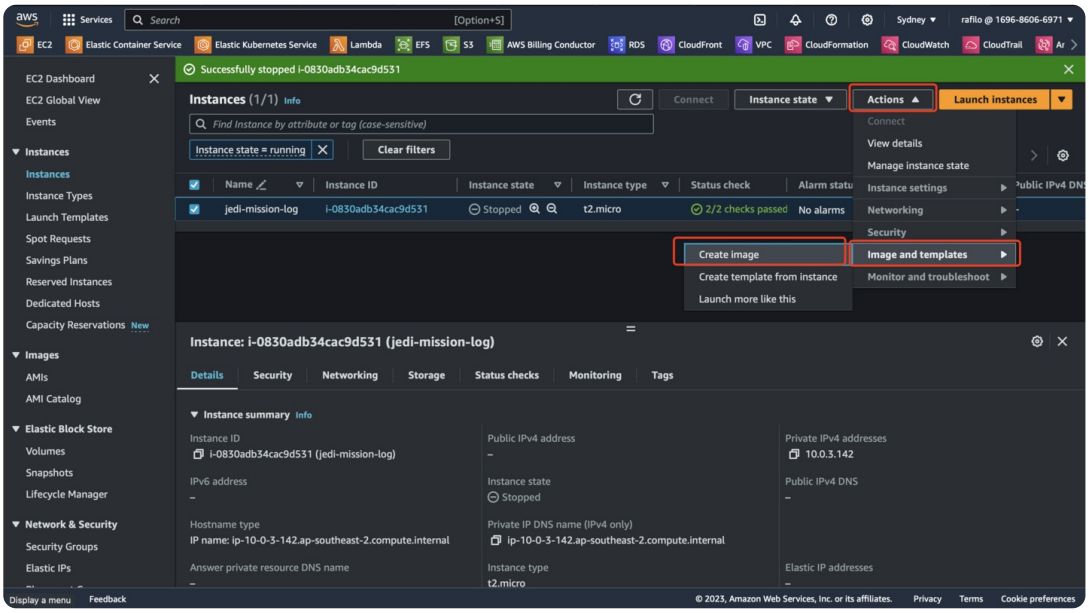


Note:

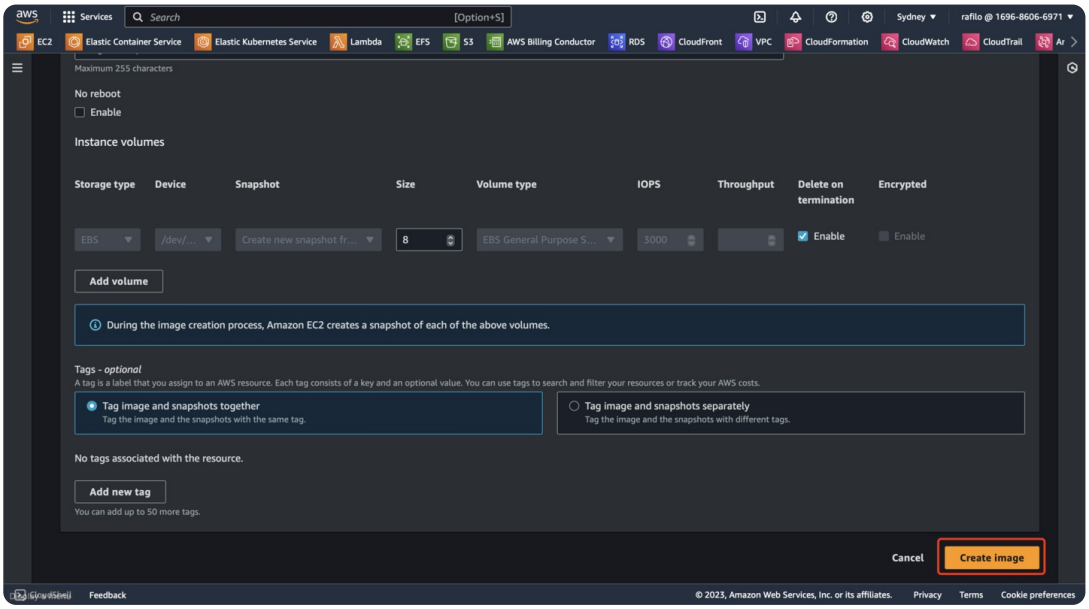
In step 7, you might encounter the "Database not found" error on the page. Click the "create database" button on the page and then refresh. Then you might encounter another "need migration" error, click "migrate" button on the page and then refresh.

5. Create an Image and Launch Template from the EC2 instance

1. Navigate to your EC2 instance page, select your instance and choose "Actions → Image and Templates → Create Image"

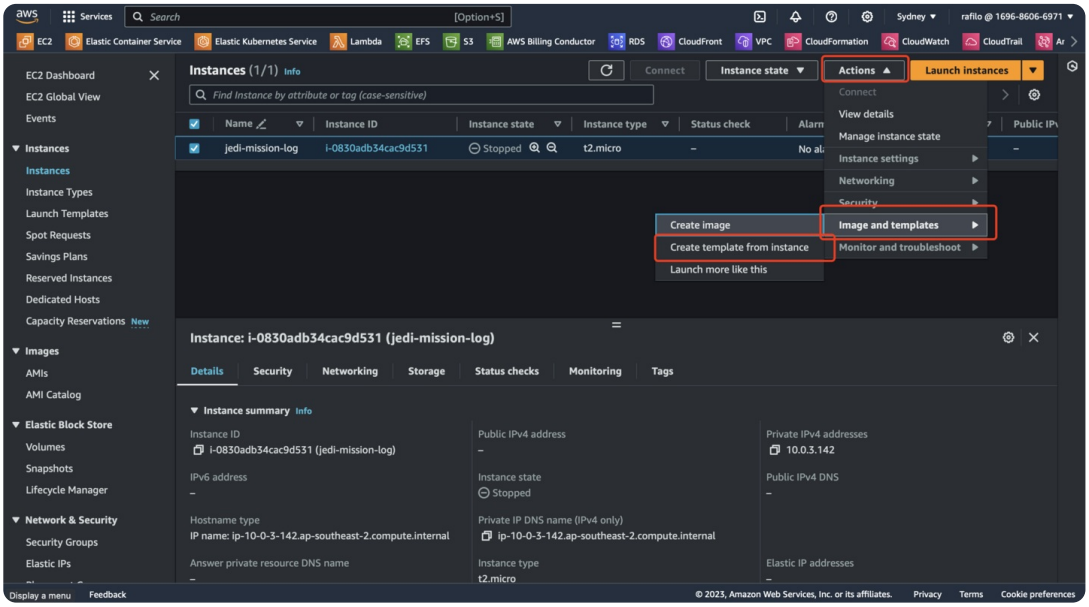


2. enter your image name and keep other things default, then click "Create Image". It'll take a long time to create your image

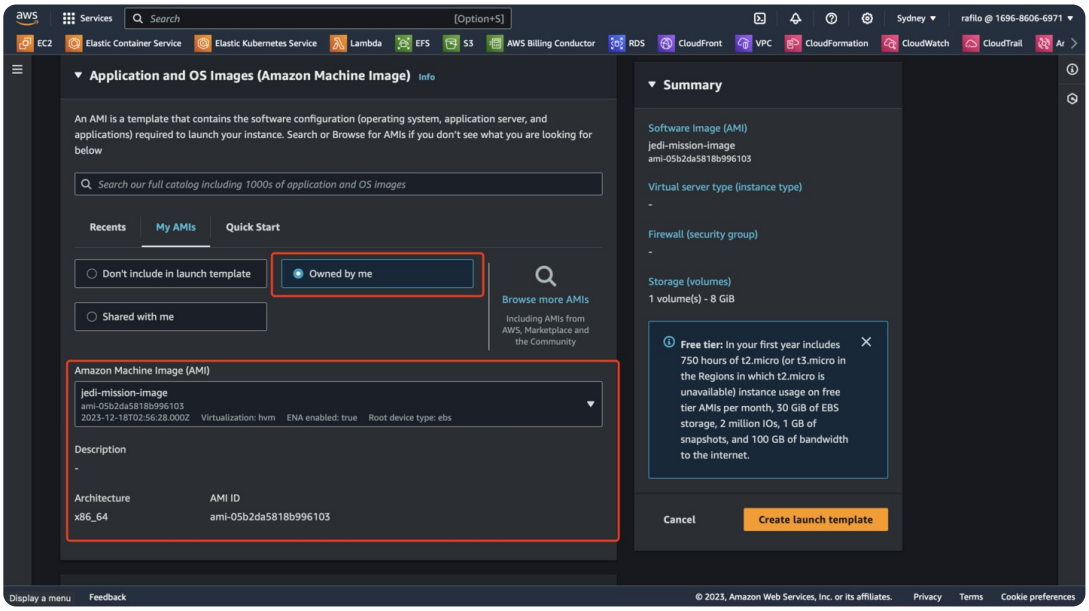


3. After your image is created, navigate back to your EC2 instance page, select your instance

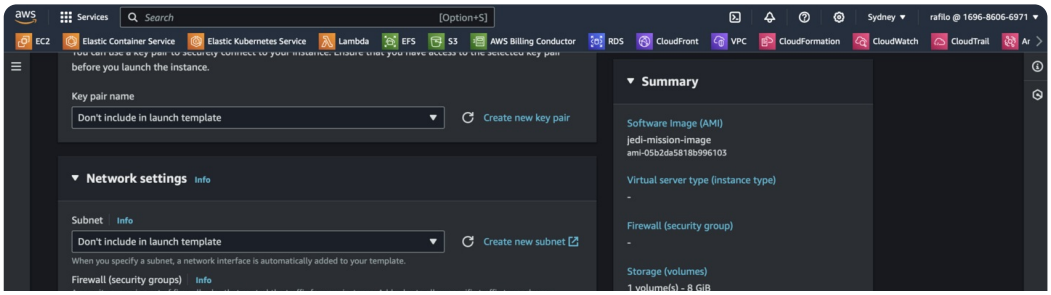
and choose "Actions → Image and Templates → Create template from instance"

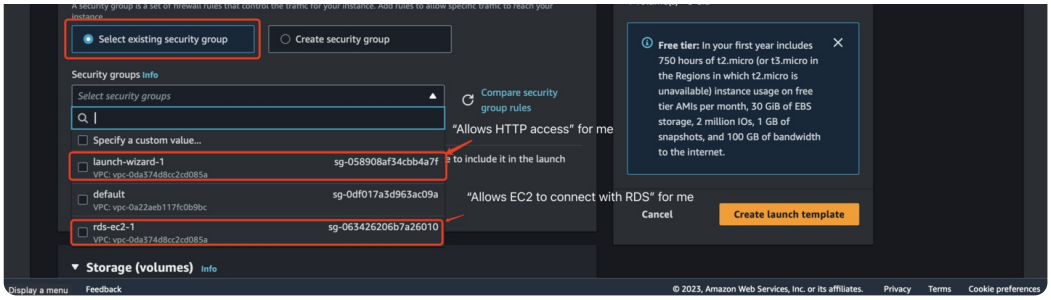


4. In the Create Launch Template page, enter the launch template name and check "Auto scaling guidance".
5. Navigate to "Application and OS Images" part, choose "My AMIs". Inside "My AMIs", select the image that you've created in previous step

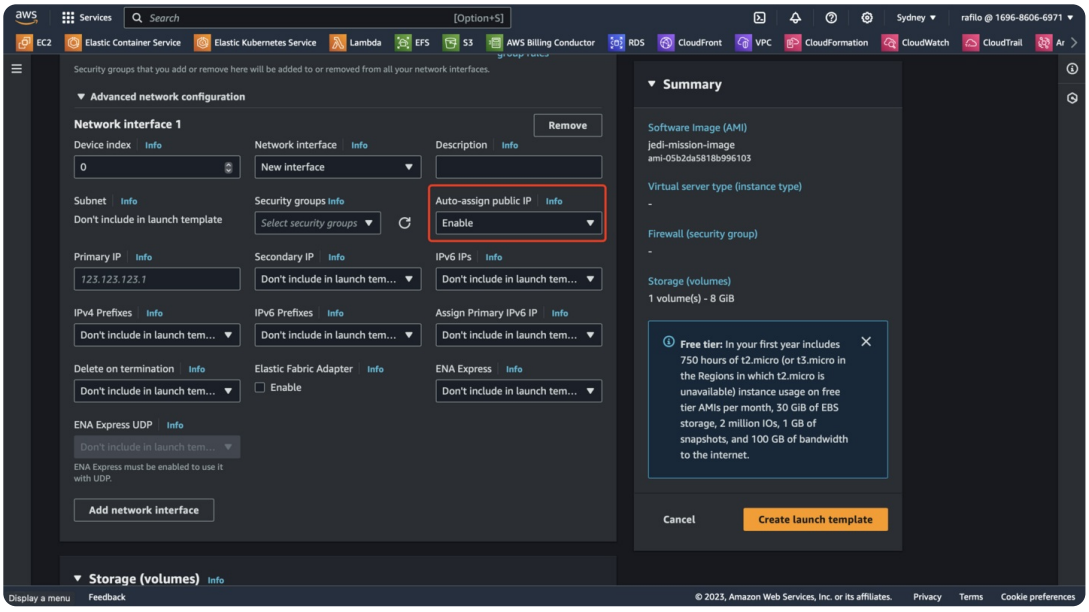


6. Navigate to "Network settings" part, select "Select existing security group", then choose the following security group that:
    1. allows HTTP access for your EC2 instance
    2. allows your EC2 instance to connect with RDS database





7. If you are not sure which security group is responsible for which, you can create new ones that satisfy the two functions mentioned above
8. Expand "Advanced network configuration", click "Add new network interface". In the newly created network interface, select enable for "auto-assign public IP"



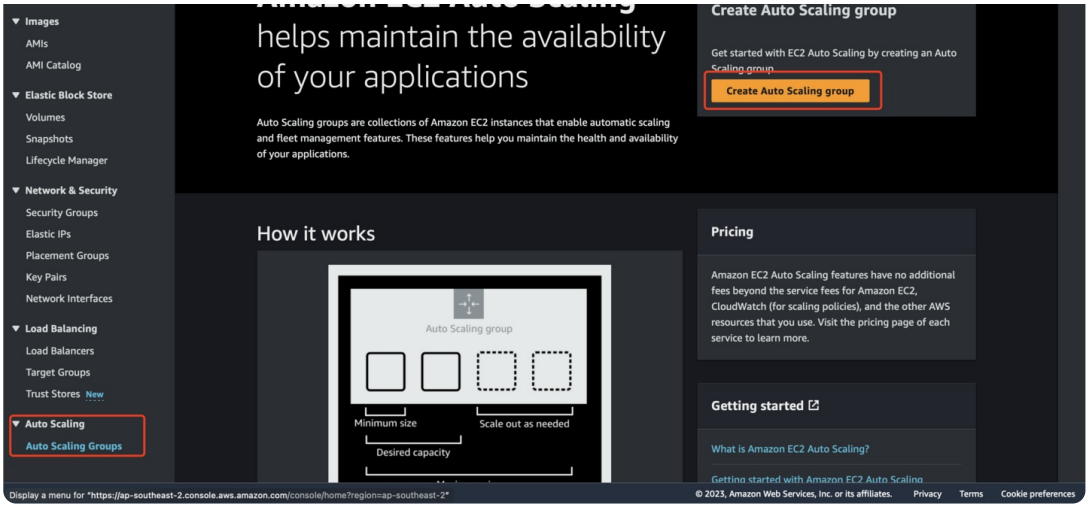
9. Scroll down and expand "Advanced details", navigate to "User data - optional" part, enter the following code, replace the last line with the one that you've used to start up the application in the previous step in EC2 instance:

```
#!/bin/bash
sudo docker pull public.ecr.aws/z7j4c9h0/nextwork/coursework/real-world-projects/jedi-mission-logs:latest
sudo docker run -d -p 80:3000 -e DATABASE_URL=postgres://root:admin123@db-jedi-mission-logs.c9xxxd3olorh.ap-southeast-2.rds.amazonaws.com:5432/jedi-mission-logs public.ecr.aws/z7j4c9h0/nextwork/coursework/real-world-projects/jedi-mission-logs:latest
```

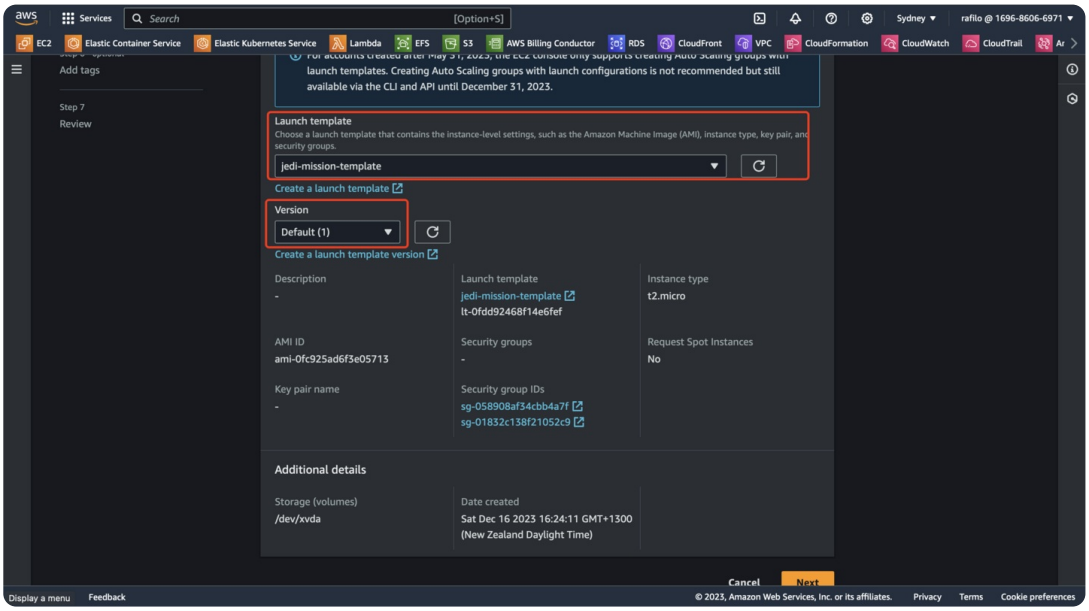
10. Follow the guidance for creating auto scaling launch template , and then click "Create launch template"

6. Create Auto Scaling Group for the application

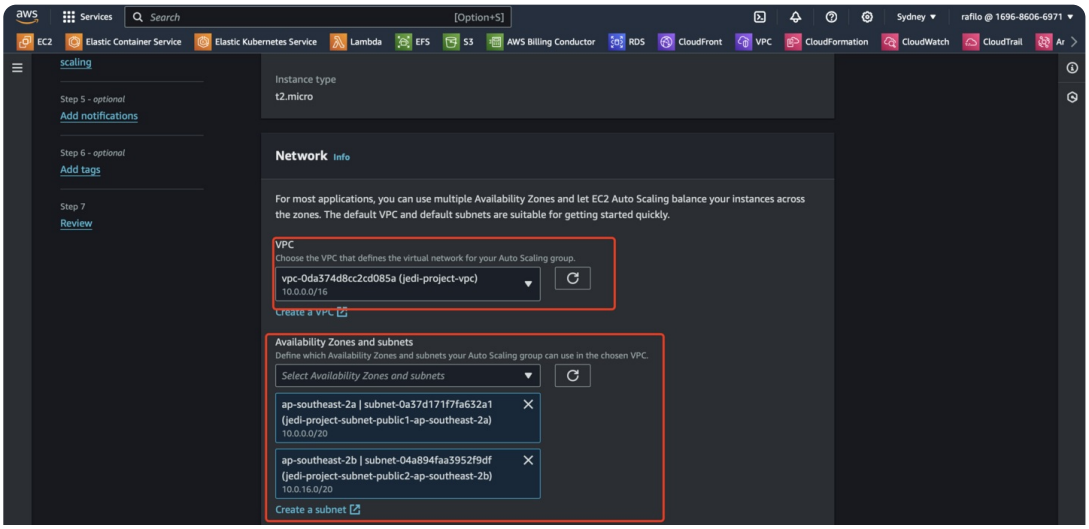
1. Navigate to "Auto Scaling groups" on left menu panel, then click "Create Auto Scaling group" in the dashboard

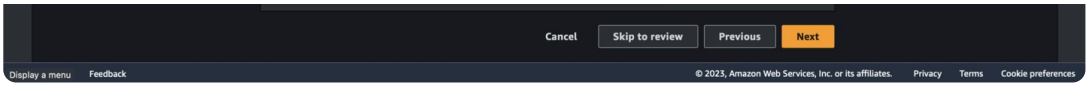


2. In the Create Auto Scaling group page, enter your auto scaling group name. Then navigate to "Launch Template" part, select the launch template that you've created in the previous step, and set the version to your latest one. Finally click "Next"

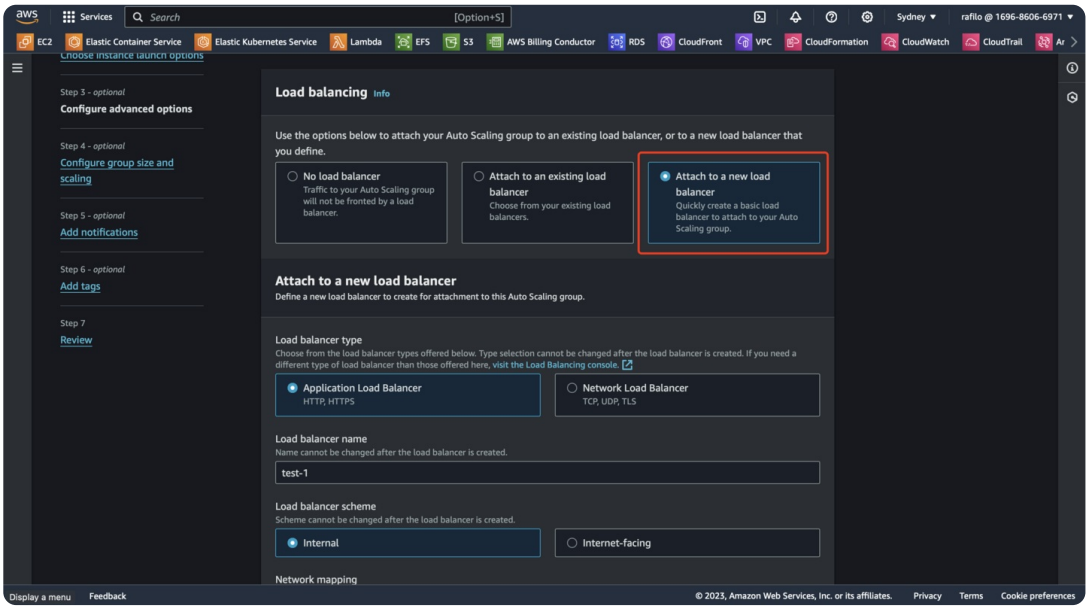


3. In the Choose instance launch options page, navigate to "Network" part, select your VPC in the "VPC" part, then select at least 2 public subnets in "Availability Zones and subnets" part.

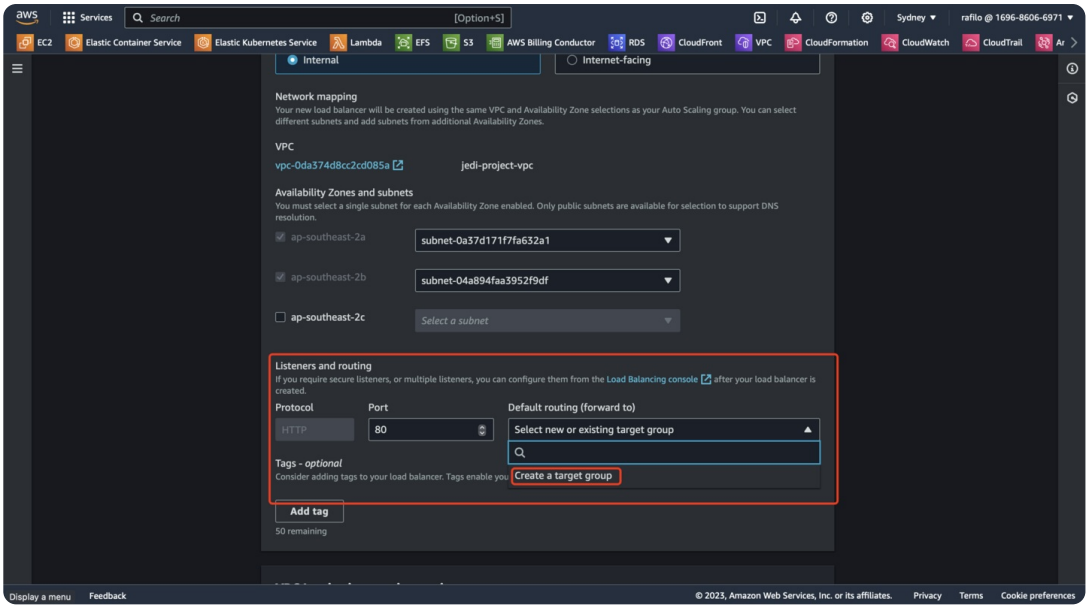




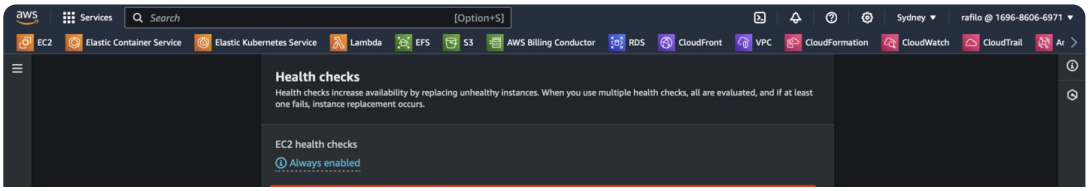
4. In the Configure advanced options page, select "Attach to a new load balancer" for Load balancing part.

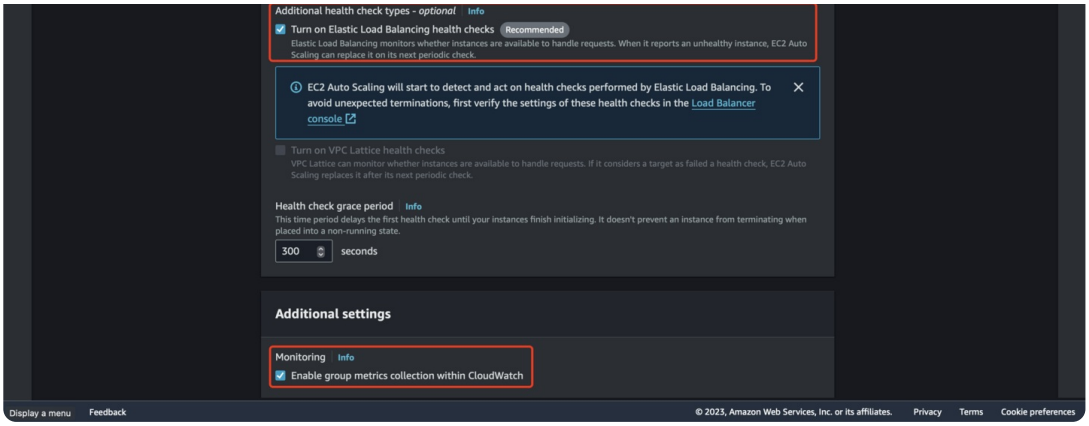


5. Navigate to "Attach to a new load balancer" part, choose Application Load Balancer for Load balancer type. then enter the name for the load balancer
6. Navigate to "Listeners and routing" part, choose "create a target group" for "Default routing (forward to)" part, then enter the name for the target group

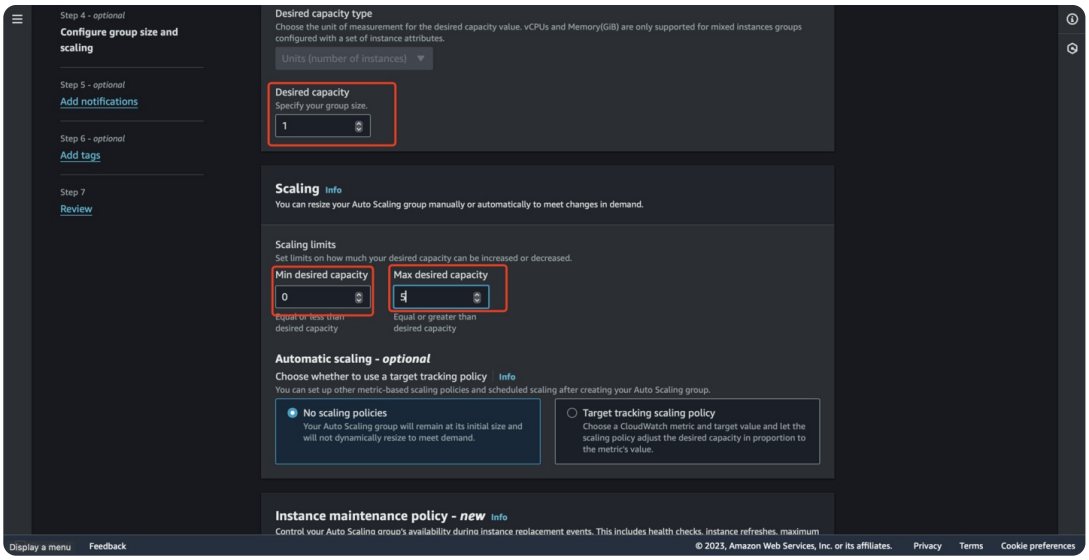


7. Navigate to "Health checks" part, check the box next to "Turn on Elastic Load Balancing health checks". Then navigate to "Additional settings" part, check the box under "monitoring" part. Finally click "Next".

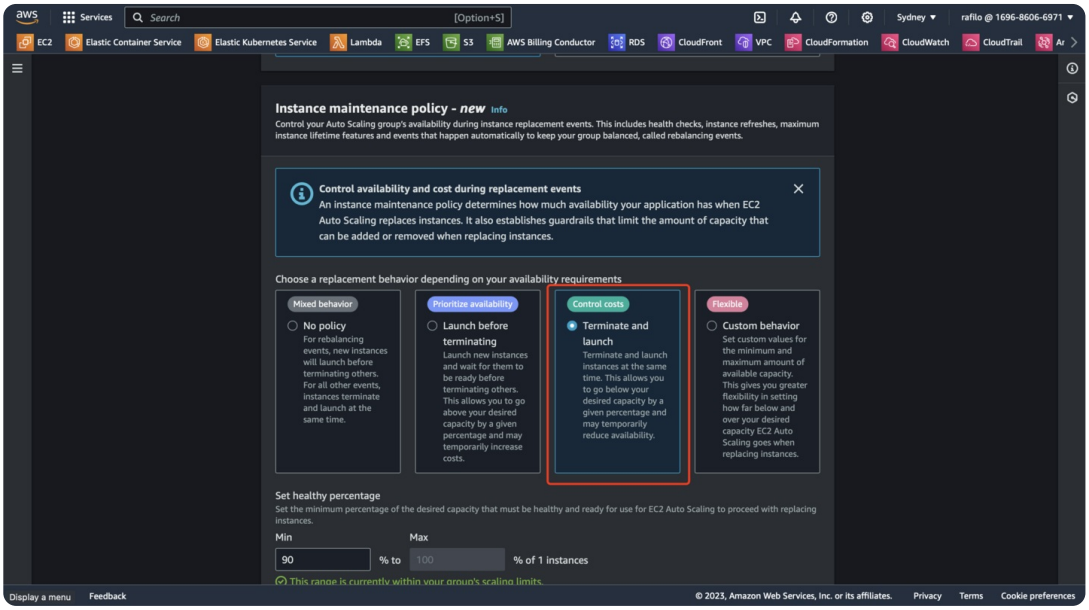




8. In "Configure group size and scaling - optional" page, navigate to "Scaling" part. In our project, we are using the minimum of 0 instances, the maximum of 5 instances, and the desired capacity of 1 instance.



9. Navigate to "Instance maintenance policy" part, select "Terminate and lunch", then scroll down to the bottom of the page, click "Next".



10. Click "Next" for the following two pages, and finally click "Create Auto Scaling group" on

the bottom of Review page

11. Now your auto scaling group will be automatically running and creating instances if the instances doesn't hit the desire instances, and will horizontally scale the number of instances based on how much CPU is occupied for each instances.

Demo

https://youtu.be/8SUo4yz7HiM
Demo link