

# 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-managed services as well as reduce the cost of Capital expenditure

i 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.

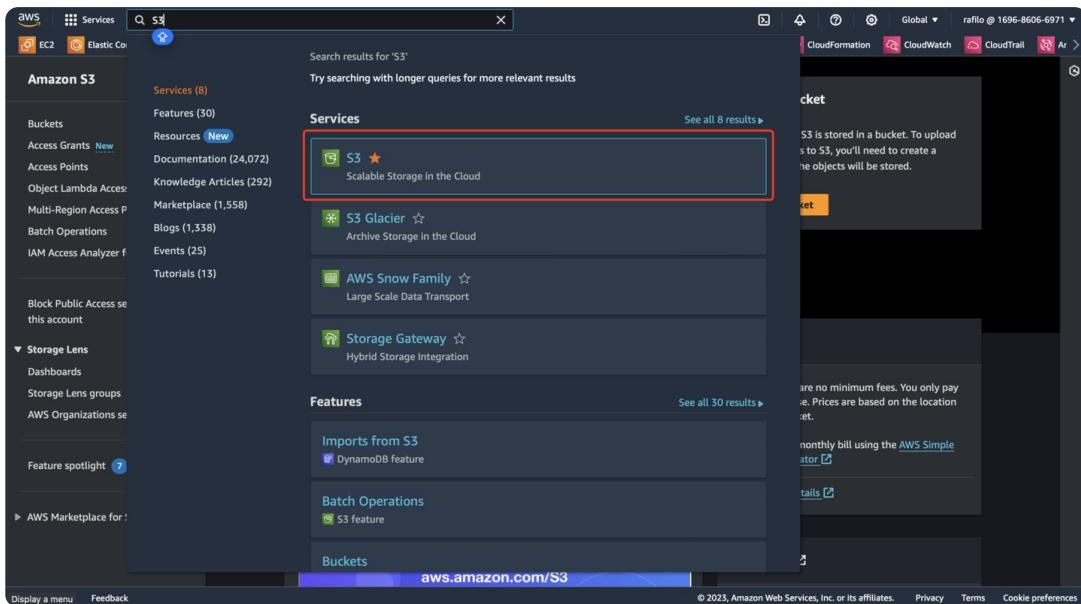
## Hosting static website with AWS S3 and CloudFront

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

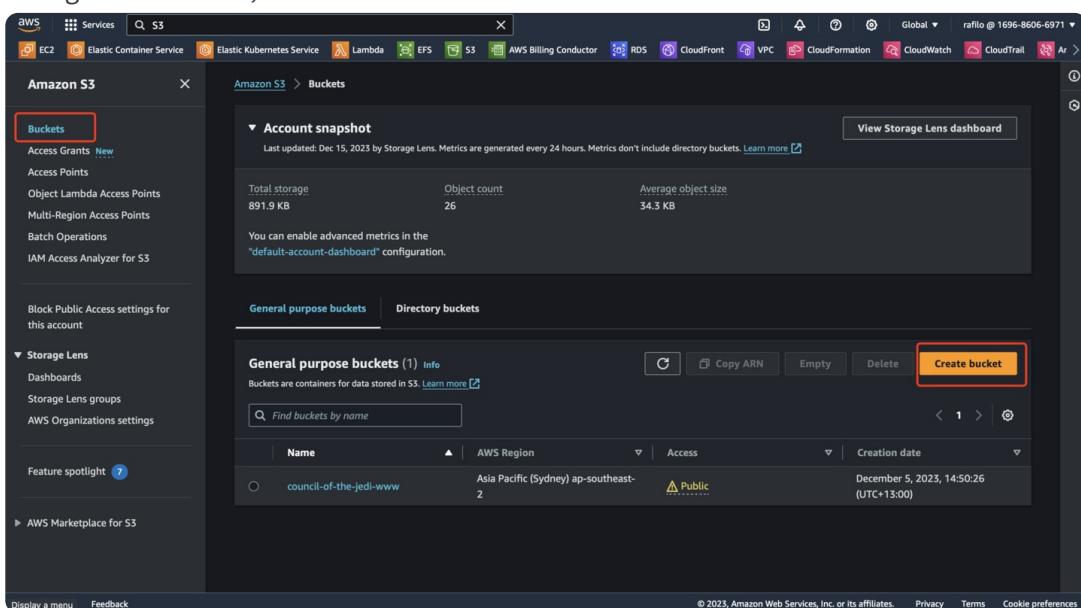
1. Create an S3 Bucket and Upload your static website to S3 Bucket
2. Hosting your static website with S3 Static website hosting service
3. Configure CloudFront for more convenient and faster access to your static website for people from all over the world

## 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>/*"
    }
  ]
}
```

```
{
  "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`

<b>Index document</b>	Specify the home or default page of the website.
<code>index.html</code>	
<b>Error document - optional</b>	This is returned when an error occurs.
<code>404.html</code>	

Click "Save Changes"

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

The screenshot shows the AWS S3 console with the 'Static website hosting' configuration for a bucket named 'jedi'. The 'Enable' radio button is selected under 'Static website hosting'. Under 'Hosting type', the 'Host a static website' radio button is selected. A callout box highlights the note: 'For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access.' The 'Index document' field contains 'index.html' and the 'Error document - optional' field contains '404.html', both of which are highlighted with red boxes.

8. Enters the Jedi Council.....



### **Extra notes: Difference between ACLs and bucket policies**

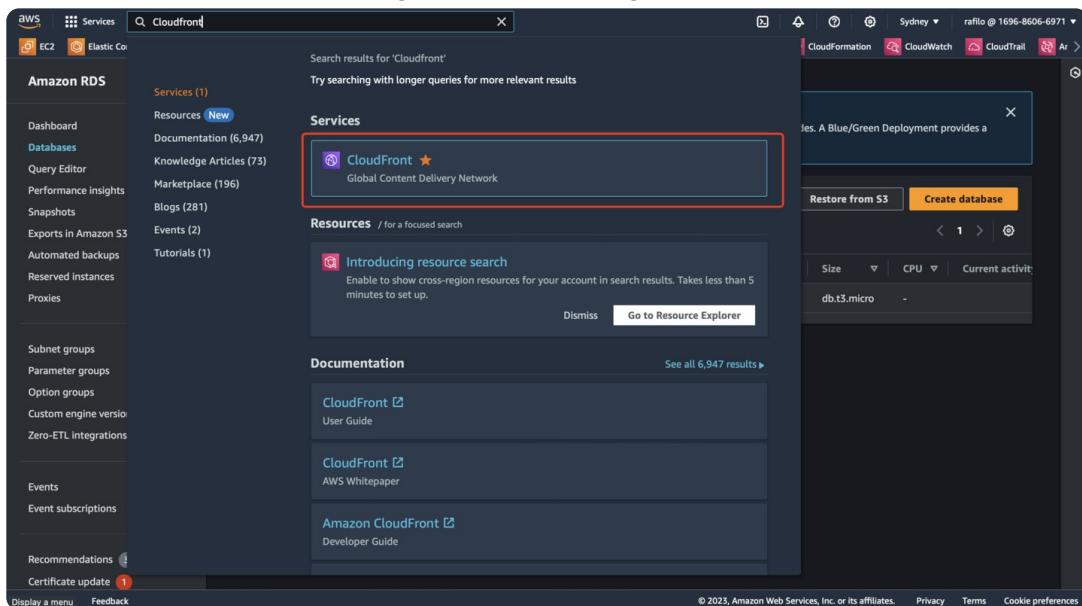
Access Control Lists (ACLs) are legacy (but not deprecated), and bucket/IAM policies are recommended by AWS.

**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

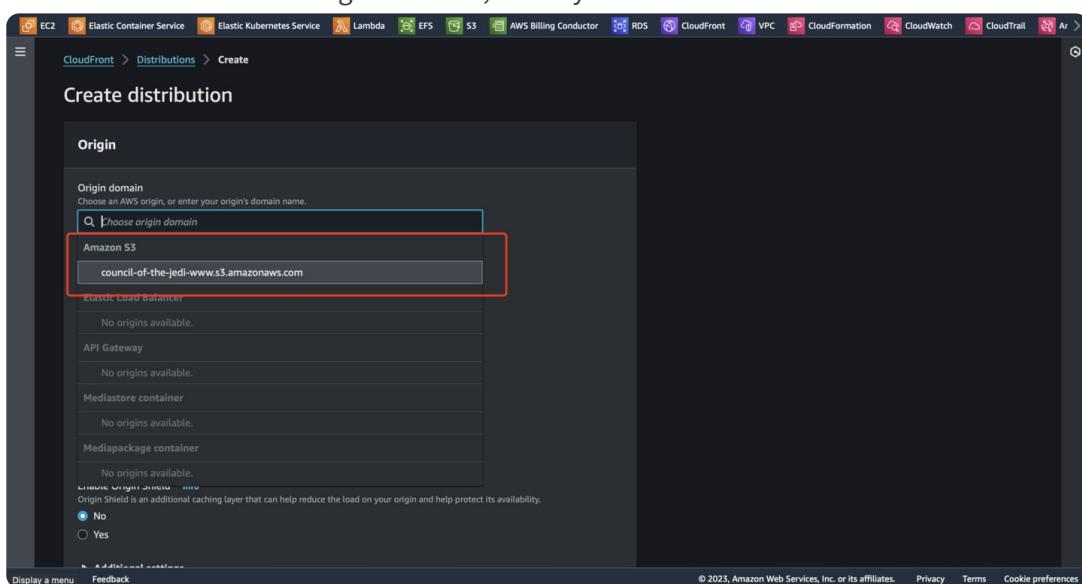
### 1. Search "CloudFront" on the navigation bar and navigate to CloudFront panel



### 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

The screenshot shows the 'Default cache behavior' configuration in the AWS CloudFront console. The 'Viewer' section is highlighted with a red box. Under 'Viewer protocol policy', the 'HTTPS only' option is selected. Under 'Allowed HTTP methods', the 'GET, HEAD' option is selected. Other options like 'HTTP and HTTPS' and 'Redirect HTTP to HTTPS' are also listed.

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

The screenshot shows the 'Web Application Firewall (WAF)' configuration page. The 'Enable security protections' option is selected and highlighted with a red box. The 'Do not enable security protections' option is also visible. Below this, there is a 'Use monitor mode' section and a 'Included security protections' section listing various protection types.

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

The screenshot shows the 'Distribution details' page for a CloudFront distribution named 'E24XBUIL2R68JE'. The 'Distribution domain name' field, containing 'd1nxv5jxh9uy.cloudfront.net', is highlighted with a red box. Other fields shown include ARN and Last modified date.

The screenshot shows the 'Settings' tab of a CloudFront distribution. It includes sections for 'Description', 'Price class' (set to 'Use all edge locations (best performance)'), 'Supported HTTP versions' (set to 'HTTP/2, HTTP/1.1, HTTP/1.0'), 'Alternate domain names', and 'Standard logging' (with 'Off' selected for both 'Cookie logging' and 'Default root object'). There is also a note about 'Alternate domain names' being disabled. At the bottom, there are links for 'Display a menu', 'Feedback', and copyright information.

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

**i** 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

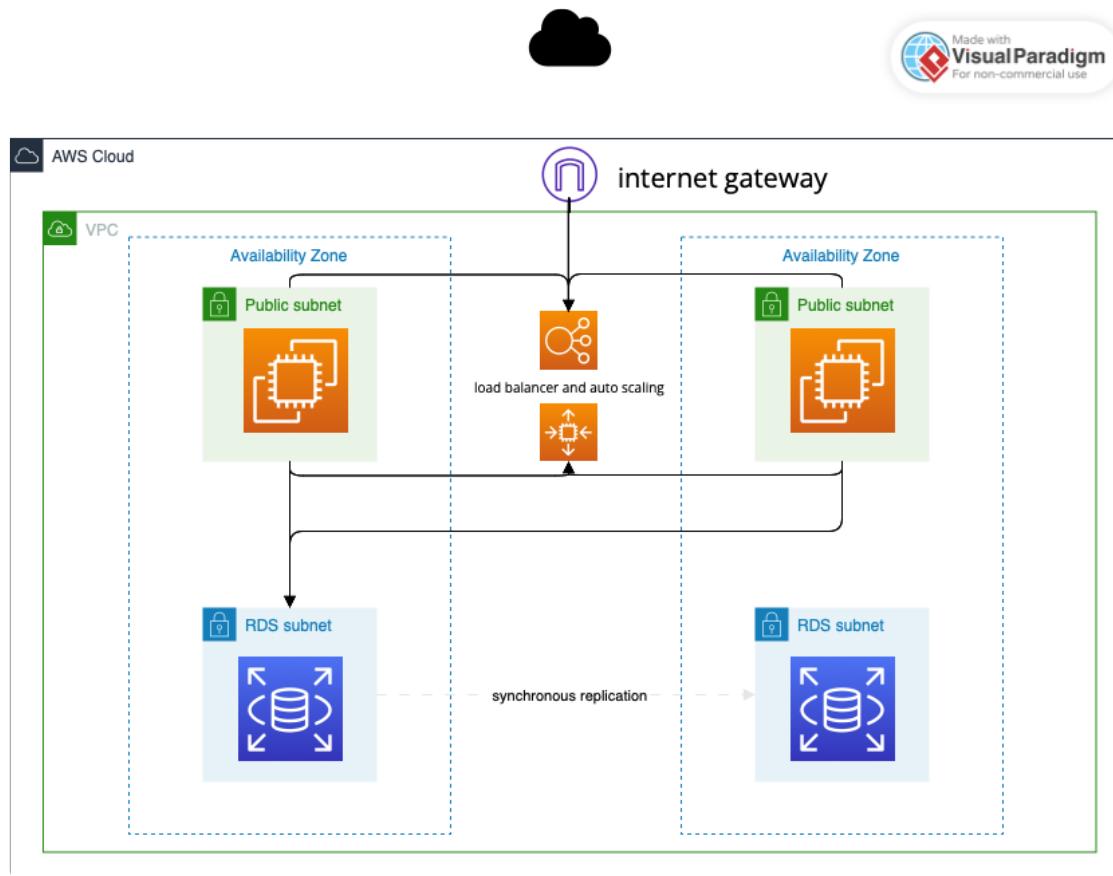
Hosting a FullStack application with AWS is more complicated compared to hosting a static website. It consists of the following steps:

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

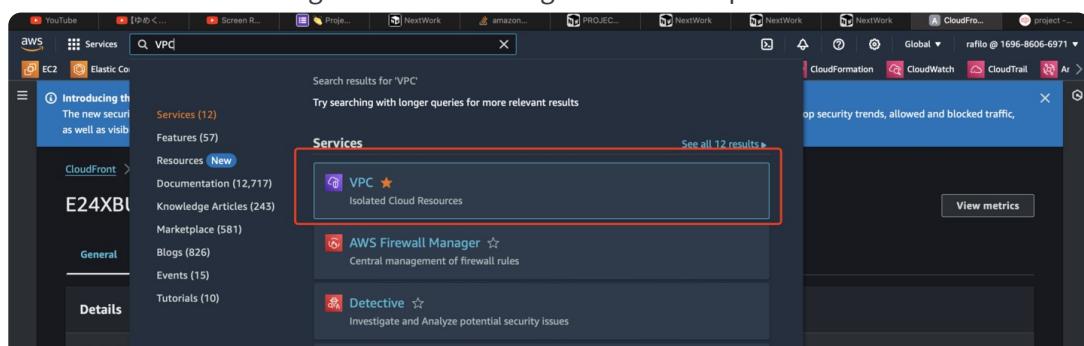


ⓘ You may wonder... "Why do we need 2 public and private subnets and 2 availability zones?"

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>

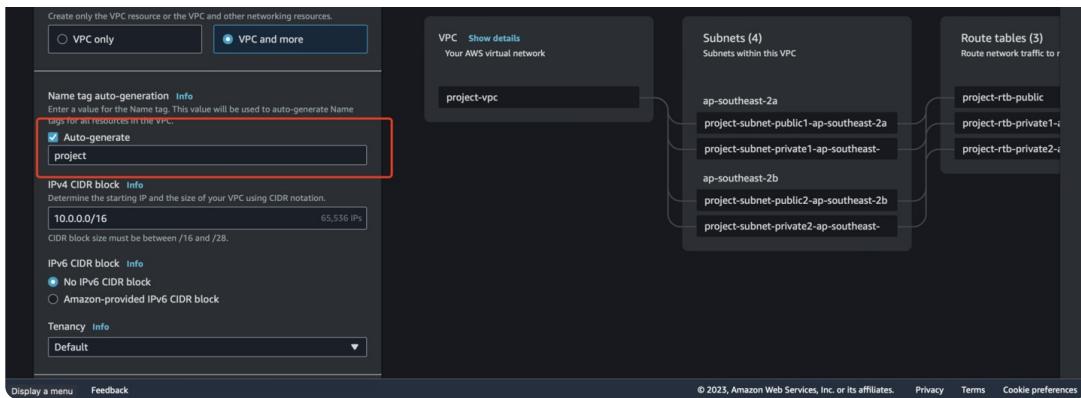
## 1. Search "VPC" on the navigation bar and navigate to the VPC panel



## 2. choose "Create VPC"

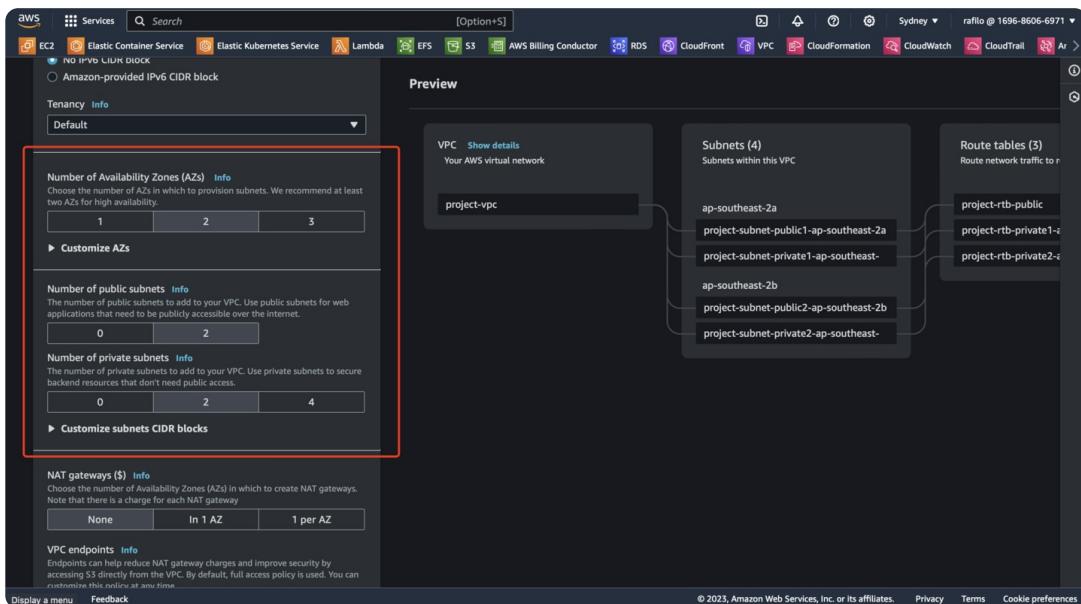
## 3. In Create VPC page, select "VPC and more" in "Resources to create" part

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



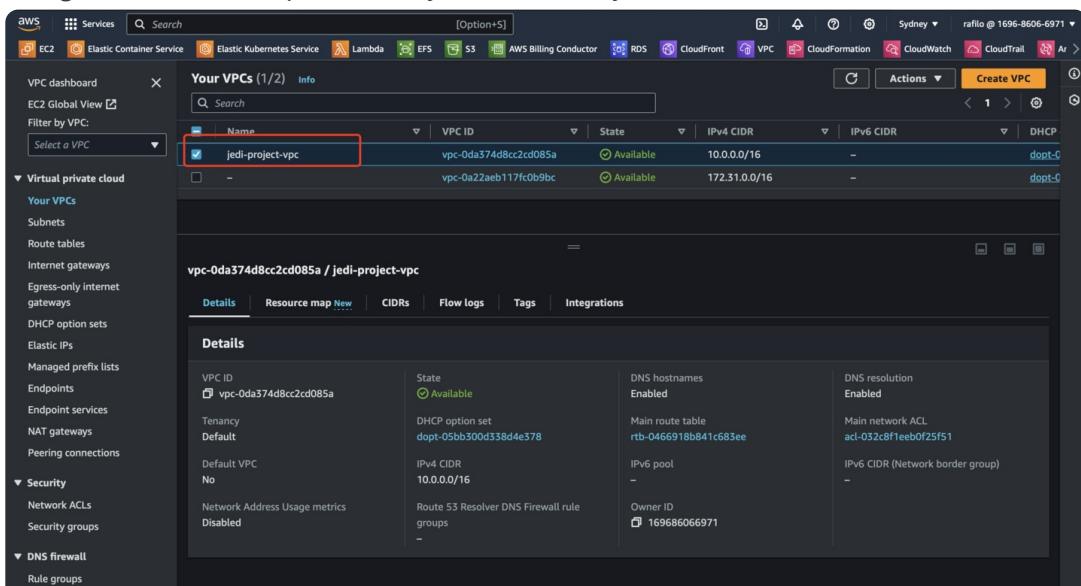
[Optional] If you want to configure the CIDR block to your desired IPs, modify the "IPv4 CIDR block" part

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



**i** 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

The screenshot shows the AWS CloudShell interface with a search bar at the top containing 'EC2'. Below the search bar, there is a 'Search results for 'EC2'' section with a message 'Try searching with longer queries for more relevant results'. A red box highlights the 'EC2' card, which is described as 'Virtual Servers in the Cloud'. Other cards visible include 'EC2 Image Builder', 'Recycle Bin', 'Amazon Inspector', and 'Elastic IPs'. To the right of the search results, there is a sidebar titled 'Free Tier Info' showing usage statistics for EC2 instances, snapshots, and storage space on EBS. At the bottom, there are account attributes and a 'View all AWS Free Tier offers' link.

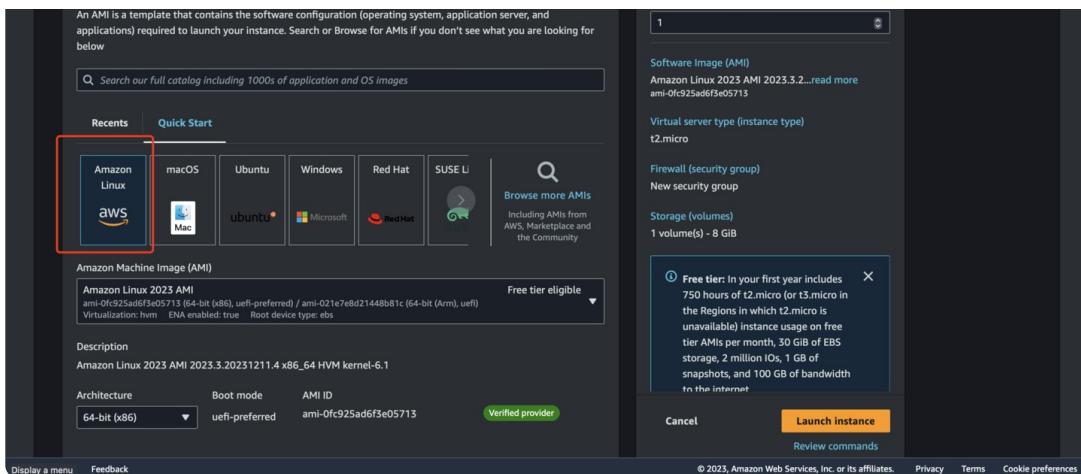
### 2. select "Launch Instance"

The screenshot shows the AWS EC2 Dashboard. On the left, there is a sidebar with various EC2-related options like Instances, Images, and Elastic Block Store. The main area is titled 'Resources' and shows metrics for running instances, auto scaling groups, dedicated hosts, elastic IPs, instances, key pairs, load balancers, placement groups, security groups, snapshots, and volumes. In the center, there is a 'Launch instance' button highlighted with a red box. Below it, there is a note stating 'Note: Your instances will launch in the Asia Pacific (Sydney) Region'. To the right, there is a 'Service health' section with a 'AWS Health Dashboard' link and a 'Zones' table showing zone names and IDs. The right side of the screen contains a sidebar with 'EC2 Free Tier Info' and offer usage statistics for Linux EC2 instances, EBS snapshot usage, and storage space on EBS. At the bottom, there are account attributes and a 'View all AWS Free Tier offers' link.

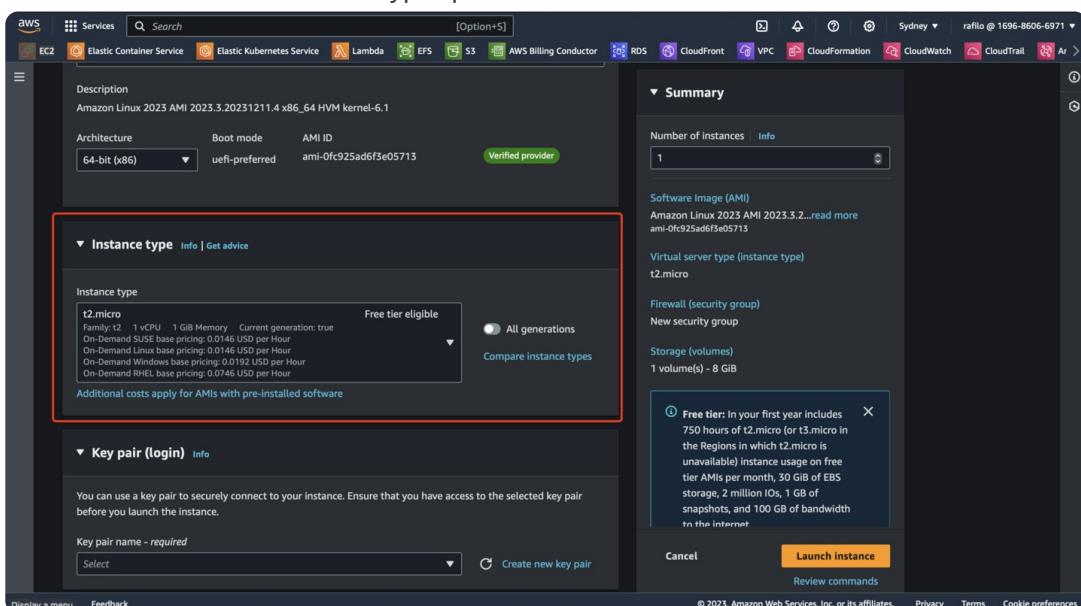
### 3. Enter the name of your EC2 instance in the "Name" part

### 4. Choose "Amazon Linux" in "Application and OS Images" part

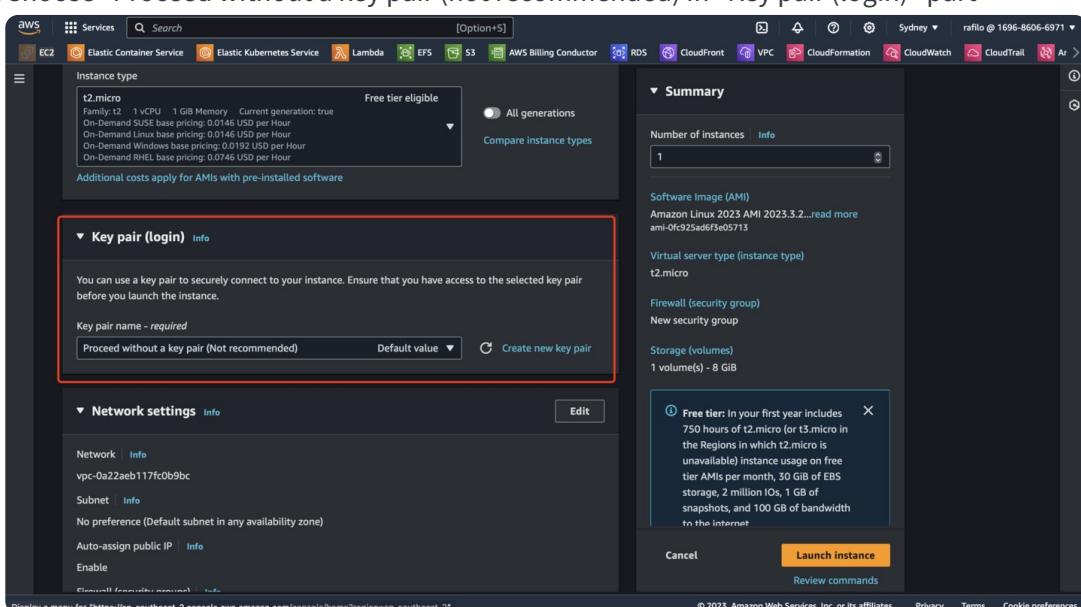
The screenshot shows the 'Application and OS Images (Amazon Machine Image)' page. The left sidebar has a single item 'Amazon Machine Images'. The main area is titled 'Summary' and shows the number of instances. At the bottom, there is a 'Number of instances' link.



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



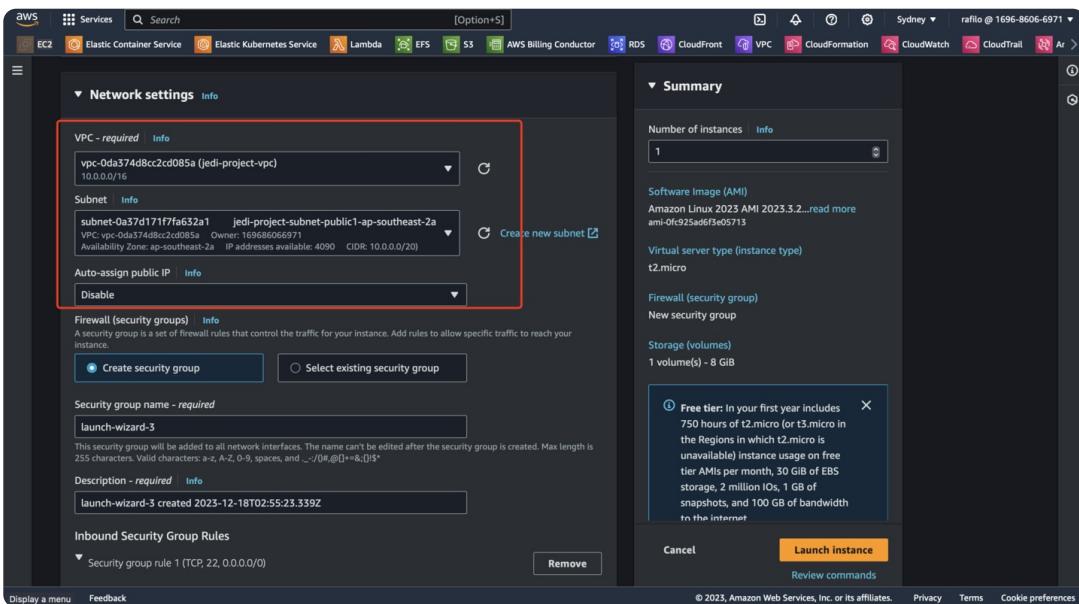
## 6. Choose "Proceed without a key pair (not recommended)" in "Key pair (login)" part



## 7. Locate to "Network Settings" part, click "Edit"

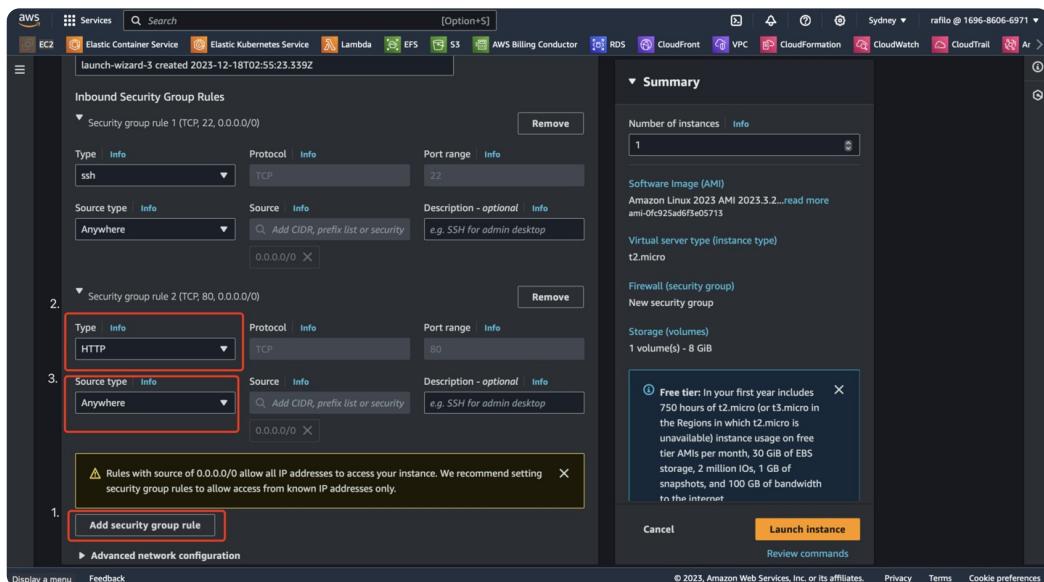
8. In "VPC-required" part, select the VPC you created in the previous step. In "Subnet" part,

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



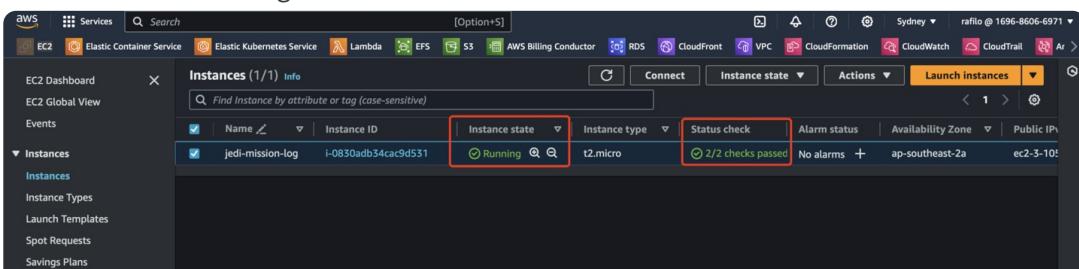
### 9. In "Inbound Security Group Rules":

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.



Reserved Instances  
Dedicated Hosts  
Capacity Reservations [New](#)

**Images**  
AMIs  
AMI Catalog

**Elastic Block Store**  
Volumes  
Snapshots  
Lifecycle Manager

**Network & Security**  
Security Groups  
Elastic IPs

Display a menu [Feedback](#)

**Instance: i-0830adb34cac9d531 (jedi-mission-log)**

**Details** Security Networking Storage Status checks Monitoring Tags

**Instance summary** Info

Instance ID: i-0830adb34cac9d531 (jedi-mission-log)  
IPv6 address: -  
Hostname type: IP name: ip-10-0-3-142.ap-southeast-2.compute.internal  
Answer private resource DNS name

Public IPv4 address: 3.105.229.20 [\[open address\]](#)  
Instance state: Running  
Private IP DNS name (IPv4 only): ip-10-0-3-142.ap-southeast-2.compute.internal  
Instance type

Private IPv4 addresses: 10.0.3.142  
Public IPv4 DNS: ec2-3-105-229-20.ap-southeast-2.compute.amazonaws.com [\[open address\]](#)

Elastic IP addresses

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

### **i 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

aws Services  Sydney rafilo @ 1696-8606-6971

EC2 Elastic Co.

EC2 Dashboard  
EC2 Global View  
Events

**Instances**  
Instances  
Instance Types  
Launch Templates  
Spot Requests  
Savings Plans  
Reserved Instances  
Dedicated Hosts  
Capacity Reservations

**Images**  
AMIs  
AMI Catalog

**Elastic Block Store**  
Volumes  
Snapshots  
Lifecycle Manager

**Network & Security**  
Security Groups  
Elastic IPs

Display a menu [Feedback](#)

Search results for 'RDS'  
Try searching with longer queries for more relevant results

**Services** See all 12 results ▾

- RDS** ★ Managed Relational Database Service
- Database Migration Service Managed Database Migration Service
- Kinesis Work with Real-Time Streaming Data
- Amazon OpenSearch Service Run open-source OpenSearch or Elasticsearch using Managed Clusters or Serverless de...

**Features** See all 37 results ▾

- Reserved instances RDS feature
- Proxies RDS feature
- Databases

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

### 2. choose "Create Database"

aws Services  [Option+S] Sydney rafilo @ 1696-8606-6971

EC2 Elastic Container Service Elastic Kubernetes Service Lambda EFS S3 AWS Billing Conductor RDS CloudFront VPC CloudFormation CloudWatch CloudTrail Ar

**Amazon RDS**

Dashboard Databases Query Editor Performance insights Snapshots Exports in Amazon S3 Automated backups Reserved instances Proxies

Subnet groups Parameter groups Option groups Custom engine versions Zero-ETL integrations [New](#)

Try the new Amazon RDS Multi-AZ deployment option for MySQL and PostgreSQL.  
For your Amazon RDS for MySQL and PostgreSQL workloads, improve transactional commit latencies by 2x, experience faster failover typically less than 35 seconds and get read scalability with two readable standby DB instances by deploying the Multi-AZ DB cluster [Learn more](#).  
**Create database**  
Or, Restore Multi-AZ DB Cluster from Snapshot

**Resources** Refresh

You are using the following Amazon RDS resources in the Asia Pacific (Sydney) region (used/quota)

DB Instances (1/40)	Parameter groups (1)
Allocated storage (0.02 TB/100 TB)	Default (1)
Increase DB instances limit <a href="#">[+]</a>	Custom (0/100)
DB Clusters (0/40)	Option groups (1)
Reserved instances (0/40)	Default (1)
Snapshots (3)	Custom (0/20)
Manual	Subnet groups (4/50)
DB Cluster (0/100)	Supported platforms <a href="#">[+]</a> VPC

**Recommended for you**

- Migrate SSRS to RDS for SQL Server
- Build RDS Operational Tasks
- Amazon RDS Backup and Restore using AWS

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

Manage master credentials in AWS Secrets Manager  
Manage master user credentials in Secrets Manager. RDS can generate a password for you and manage it throughout its lifecycle.

If you manage the master user credentials in Secrets Manager, some RDS features aren't supported.  
[Learn more](#)

Auto generate a password  
Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), 'single quote), \"double quote) and @ (at sign)

Confirm master password [Info](#)

Display a menu Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## 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

Compute resource  
Choose whether 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 can 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.

Virtual private cloud (VPC) [Info](#)  
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

jedi-project-vpc (vpc-0da374d8cc2cd085a)  
7 Subnets, 3 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

DB subnet group [Info](#)  
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

rds-ec2-db-subnet-group-4  
3 Subnets, 3 Availability Zones

Public access [Info](#)  
 Yes  
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

No  
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

Display a menu Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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

VPC security group (firewall) [Info](#)  
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

Choose existing  
Choose existing VPC security groups

Create new  
Create new VPC security group

New VPC security group name

Amazon RDS will add a new VPC security group rds-ec2-2 to allow connectivity with your compute resource.

Availability Zone [Info](#)  
No preference

Certificate authority - optional [Info](#)  
Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.

rds-ca-2019 (default)  
Expiry: Aug 23, 2024

If you don't select a certificate authority, RDS chooses one for you.

Additional configuration

Database authentication

Display a menu Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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

Amazon RDS [X](#) RDS > Databases

Consider creating a Blue/Green Deployment to minimize downtime during upgrades

The screenshot shows the AWS RDS Dashboard. On the left sidebar, under the 'Databases' section, there are various options like 'Query Editor', 'Performance insights', and 'Exports in Amazon S3'. The main area displays a table titled 'Databases (1)'. The table has columns for 'DB identifier', 'Status', 'Role', 'Engine', 'Region & AZ', 'Size', 'CPU', and 'Current activity'. The single row in the table corresponds to the database 'db-jedi-mission-logs', which is highlighted with a red box. The status is 'Stopped temporarily', the engine is 'PostgreSQL', and the instance is 'ap-southeast-2a'.

- 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

The screenshot shows the 'Connectivity & security' tab for the database 'db-jedi-mission-logs'. The 'Endpoint' field is highlighted with a red box and contains the value 'db-jedi-mission-logs.ckxxxd0lorh.ap-southeast-2.rds.amazonaws.com'. The 'Networking' and 'Security' sections are also visible, showing details like Availability Zone (ap-southeast-2a), VPC (jedi-project-vpc), Subnet group (rds-ec2-db-subnet-group-4), Subnets (subnet-003d3f44e916b6d4c, subnet-0baea5b8e40bf1c99, subnet-0032de43eca177d45), and VPC security groups (rds-e2-1, launch-wizard-1).

#### 4. Install Docker to start the application

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

The screenshot shows the EC2 Instances page. The table lists one instance named 'jedi-mission-log' with the instance ID 'i-0830adb34cac9d531'. The 'Name' column has a checked checkbox, and the 'Actions' column has a 'Connect' button highlighted with a red box. Below the table, the instance details are shown, including its public IPv4 address (3.105.229.20) and private IPv4 address (10.0.3.142).

Hostname type: IP name: ip-10-0-3-142.ap-southeast-2.compute.internal  
 Private IP DNS name (IPv4 only): ip-10-0-3-142.ap-southeast-2.compute.internal

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

Instance ID: i-0830adb34cac9d531 (jedi-mission-log)  
 Connection Type:  Connect using EC2 Instance Connect  
 Public IP address: 3.105.229.20  
 User name: ec2-user  
 Note: In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

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

Amazon Linux 2023  
 https://aws.amazon.com/linux/amazon-linux-2023  
 Last login: Sat Dec 16 03:17:25 2023 from 13.239.158.5  
 [ec2-user@ip-10-0-3-142 ~]\$

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

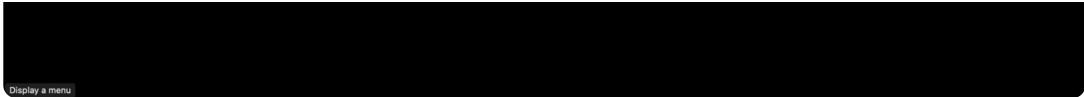
```
docker run -d -p 80:3000 -e DATABASE_URL=postgres://<your username>:<your password here>@<DB instance URL here>:5432/jedi-mission-logs
public.ecr.aws/z7j4c9h0/nextwork/coursework/real-world-projects/jedi-mission-logs:latest
```

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

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, and Elastic IPs. The main area displays an instance summary for 'i-0830adb34cac9d531 (jedi-mission-log)'. The instance ID is listed under 'Instance ID'. The 'Public IPv4 address' field contains '3.105.229.20' with a link to 'open address'. The 'Instance state' is shown as 'Running'. Other details include Hostname type (IP name: ip-10-0-3-142.ap-southeast-2.compute.internal), Private IP DNS name (ip-10-0-3-142.ap-southeast-2.compute.internal), Instance type (t2.micro), VPC ID (vpc-0da374d8cc2cd085a (jedi-project-vpc)), Subnet ID (subnet-0a37d171f7fa652a1 (jedi-project-subnet-public1-ap-southeast-2a)), and IAM Role (Required). At the bottom, there are tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags, with 'Details' being the active tab. Below the tabs, there's a section for 'Instance details' with a link to 'Info'.

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

The screenshot shows a web application titled 'Jedi Council Mission Logs'. The main heading is 'Completed missions'. Below the heading is a form with three input fields: 'ENTRY:' containing 'Mace Windu', 'JEDI:' containing 'Mace Windu', and 'DATE COMPLETED:' containing 'Mace Windu'. Below the form is a blue button labeled 'Log a mission'. The page has a dark background with white text and light-colored buttons.

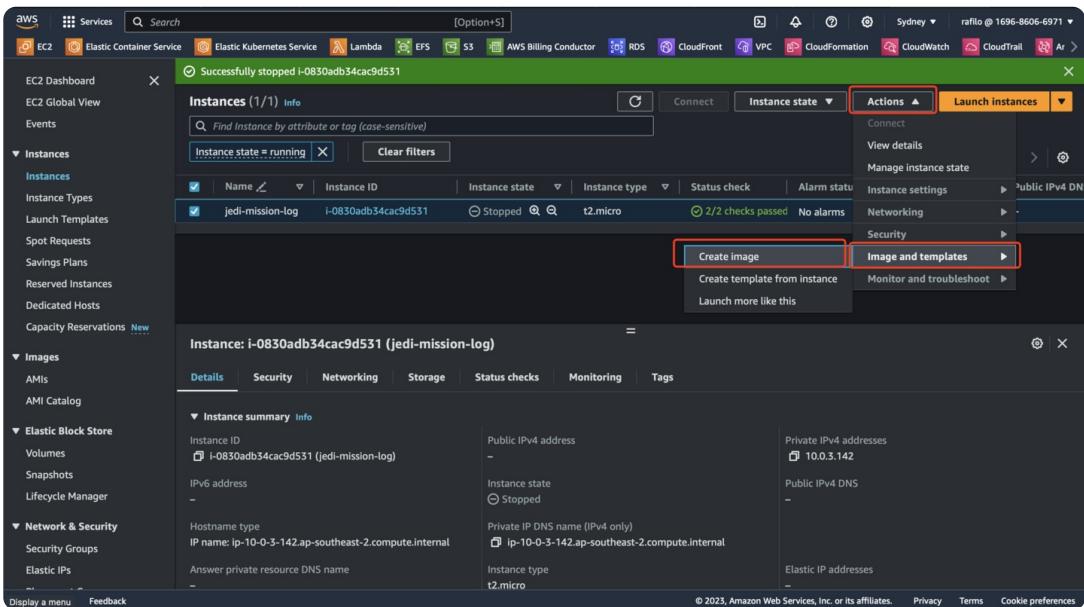


**Note:**

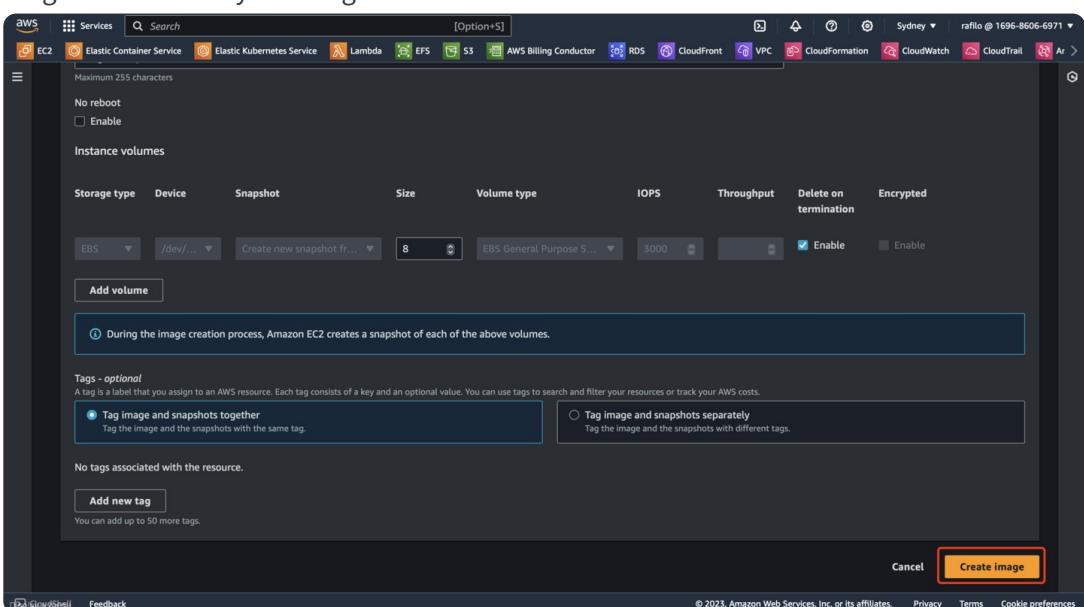
In step 7, you might encounter the "Database not found" error on the page. Click the "create database" button on the page and then refresh. Then you might encounter another "need migration" error, click "migrate" button on the page and then refresh.

## 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"

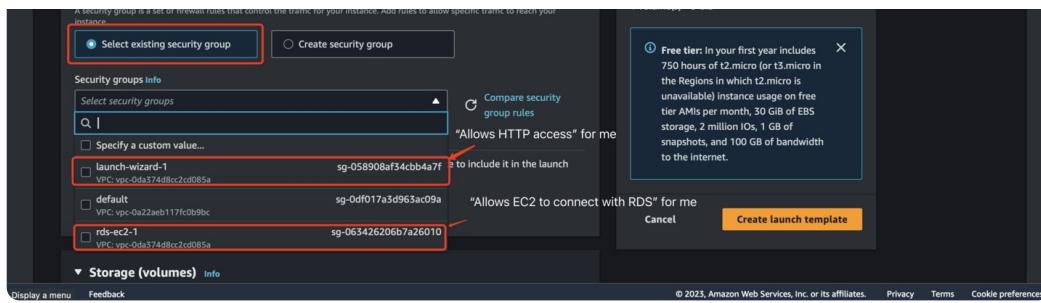
The screenshot shows the AWS EC2 Instances page. A specific instance, 'jedi-mission-log', is selected. The 'Actions' dropdown menu is open, and the 'Image and templates' option is highlighted with a red box. Other options like 'Create image' and 'Create template from instance' are also visible.

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

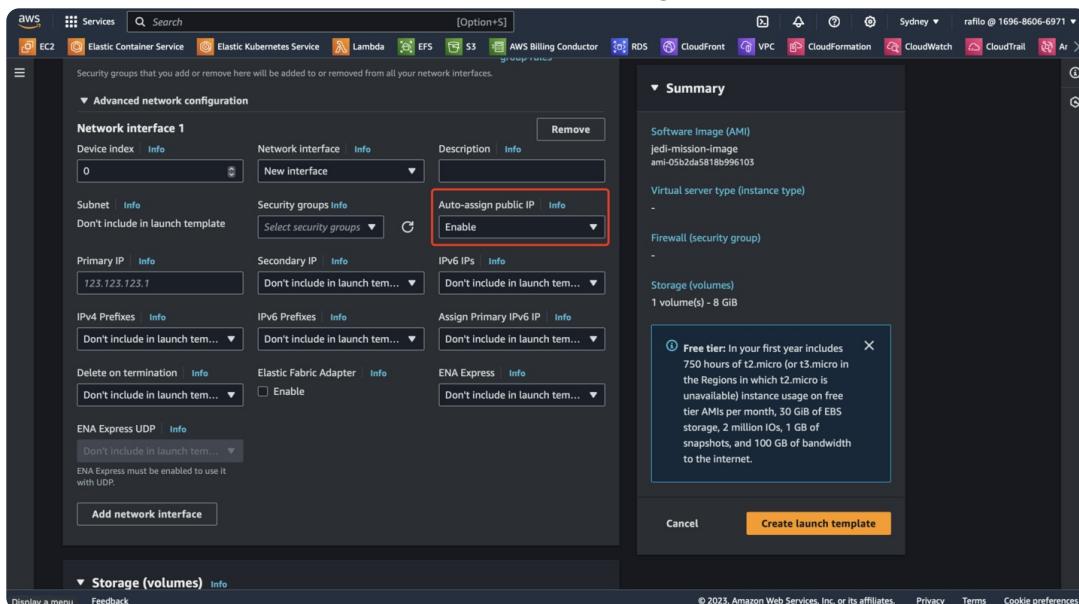
The screenshot shows the AWS Application and OS Images (Amazon Machine Image) page. The 'My AMIs' tab is selected. A specific AMI, 'jedi-mission-image', is highlighted with a red box. A modal window titled 'Create launch template' is open on the right side, containing details about the free tier and a 'Create launch template' button.

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

The screenshot shows the AWS Network settings page for a launch template. The 'Subnet' section is displayed, with the 'Don't include in launch template' option selected. A note below states: 'When you specify a subnet, a network interface is automatically added to your template.'



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"

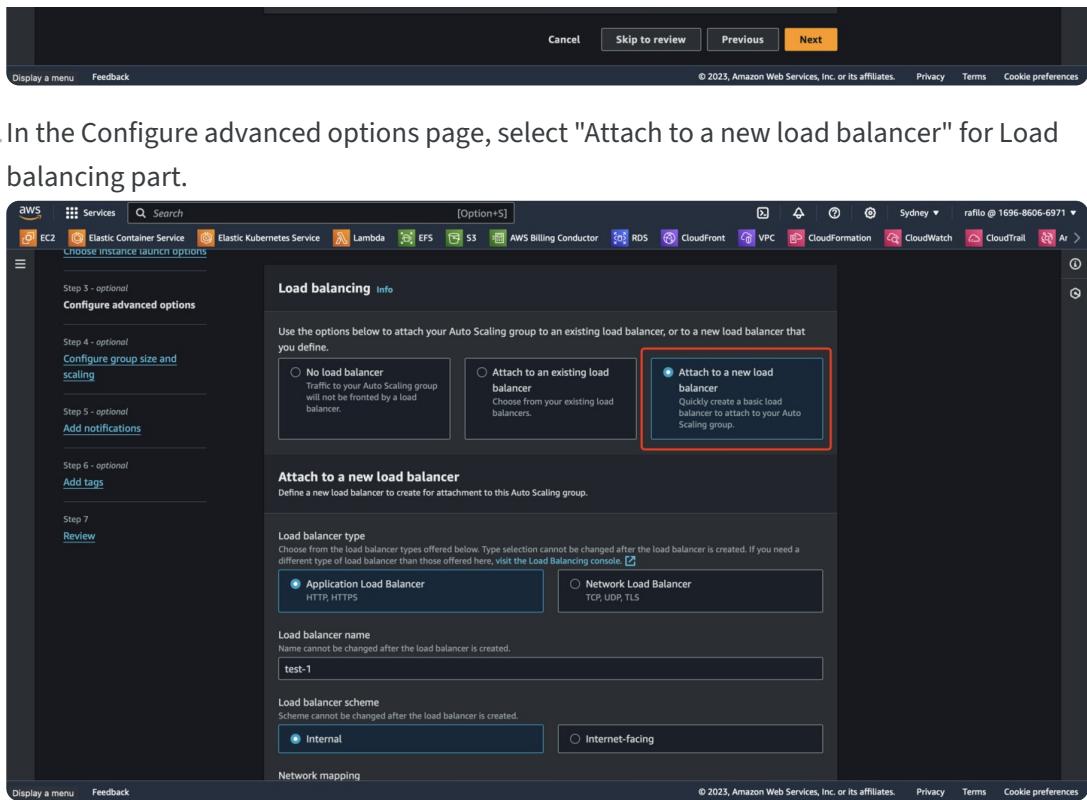
## 6. Create Auto Scaling Group for the application

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"

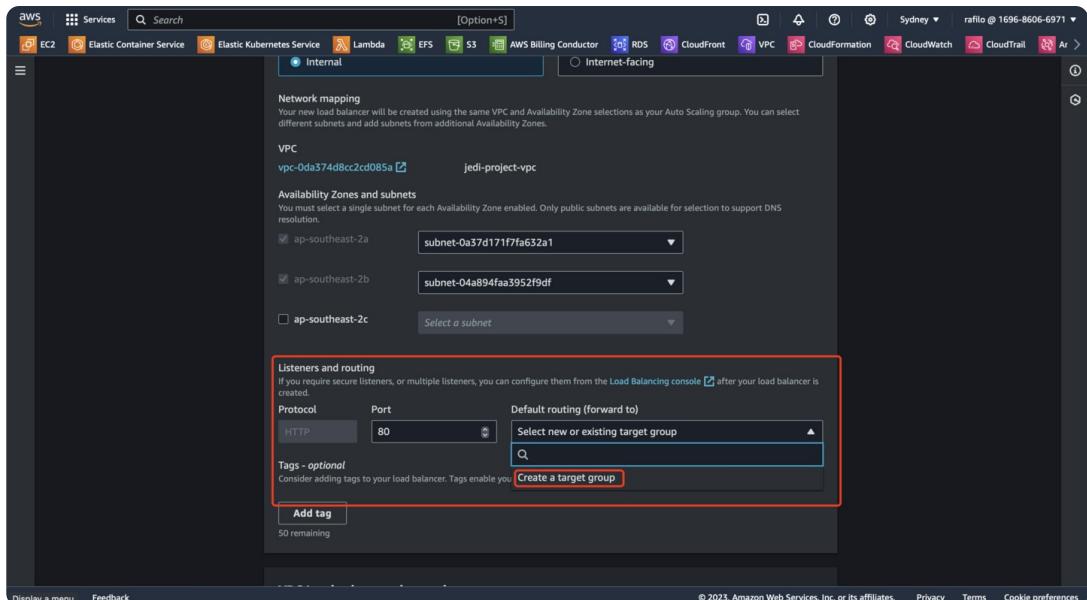
3. In the Choose instance launch options page, navigate to "Network" part, select your VPC in the "VPC" part, then select at least 2 public subnets in "Availability Zones and subnets" part.



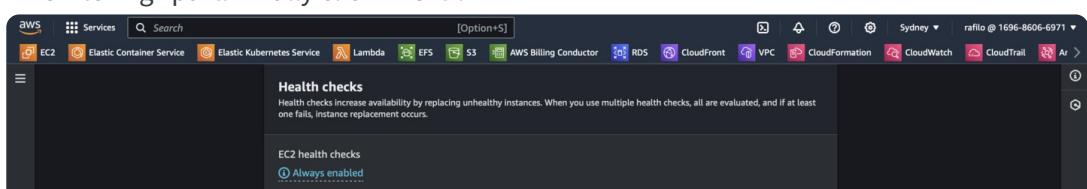
4. In the Configure advanced options page, select "Attach to a new load balancer" for Load balancing part.

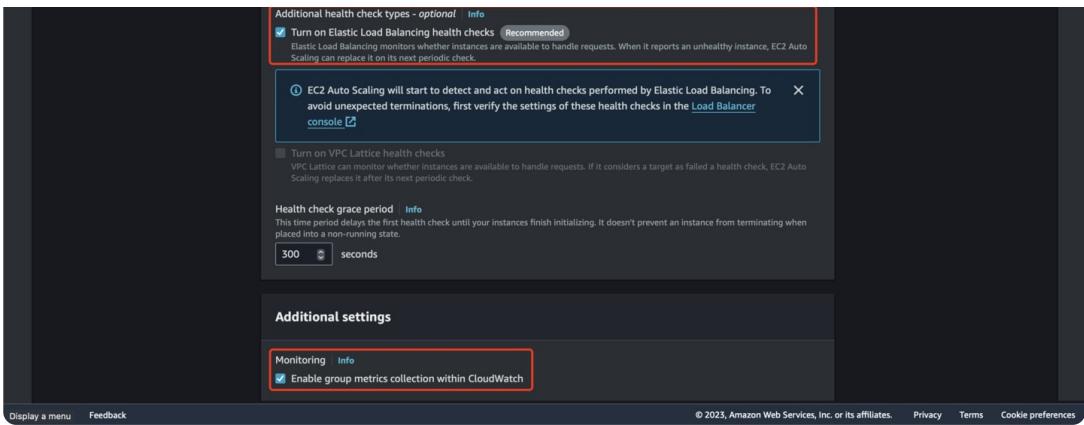
5. Navigate to "Attach to a new load balancer" part, choose Application Load Balancer for Load balancer type. then enter the name for the load balancer

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

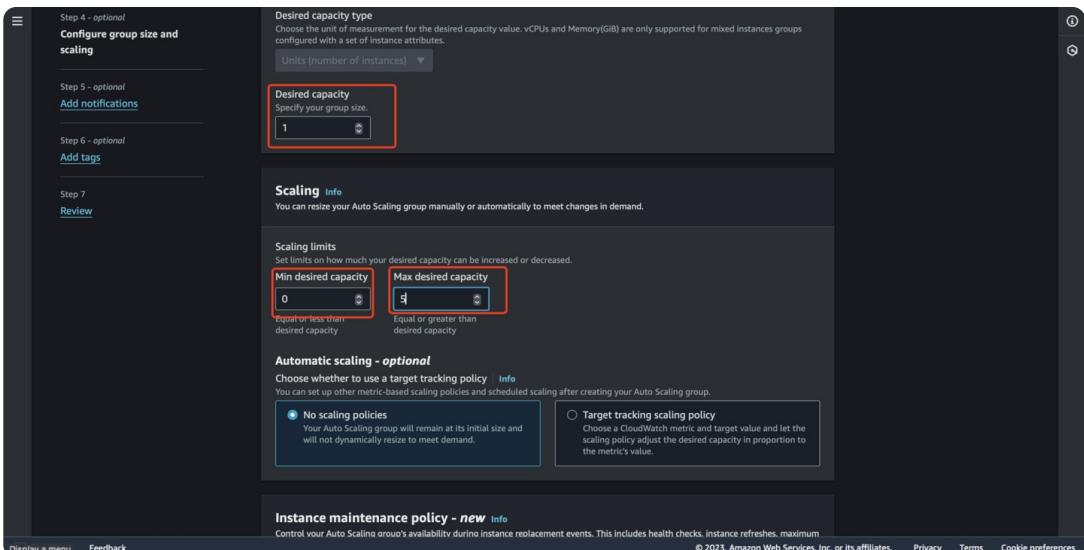


7. Navigate to "Health checks" part, check the box next to "Turn on Elastic Load Balancing health checks". Then navigate to "Additional settings" part, check the box under "monitoring" part. Finally click "Next".

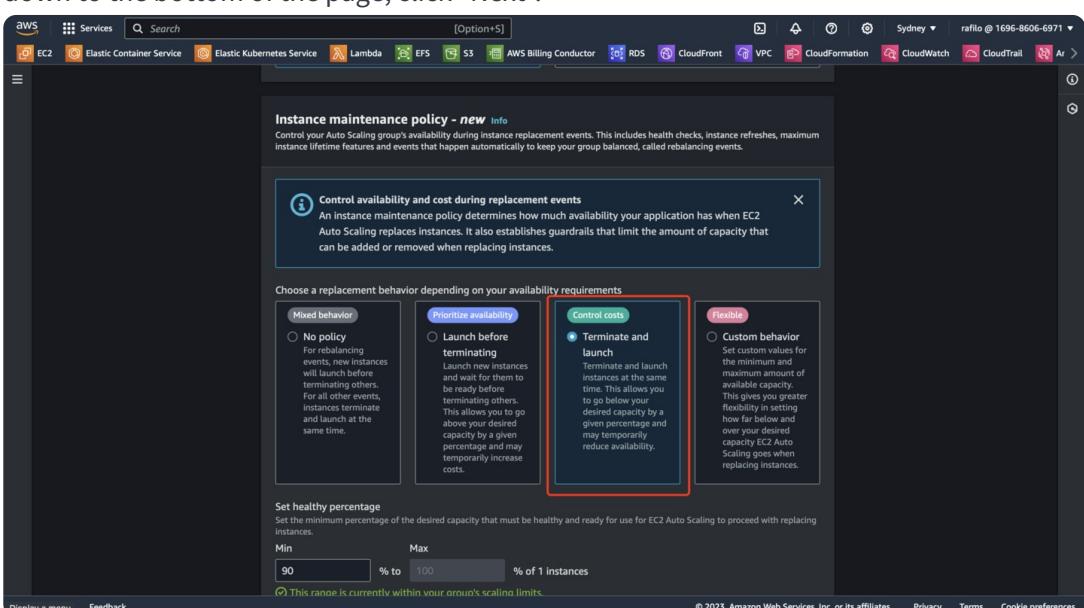




8. In "Configure group size and scaling - optional" page, navigate to "Scaling" part. In our project, we are using the minimum of 0 instances, the maximum of 5 instances, and the desired capacity of 1 instance.



9. Navigate to "Instance maintenance policy" part, select "Terminate and launch", 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](#)