

# ELC Cloud and DevOps Program

---

**Application Hosting on AWS**

## Problem Solution

First Step of solution is to create the VPC for secure design where we are going to launch our database and python application server.

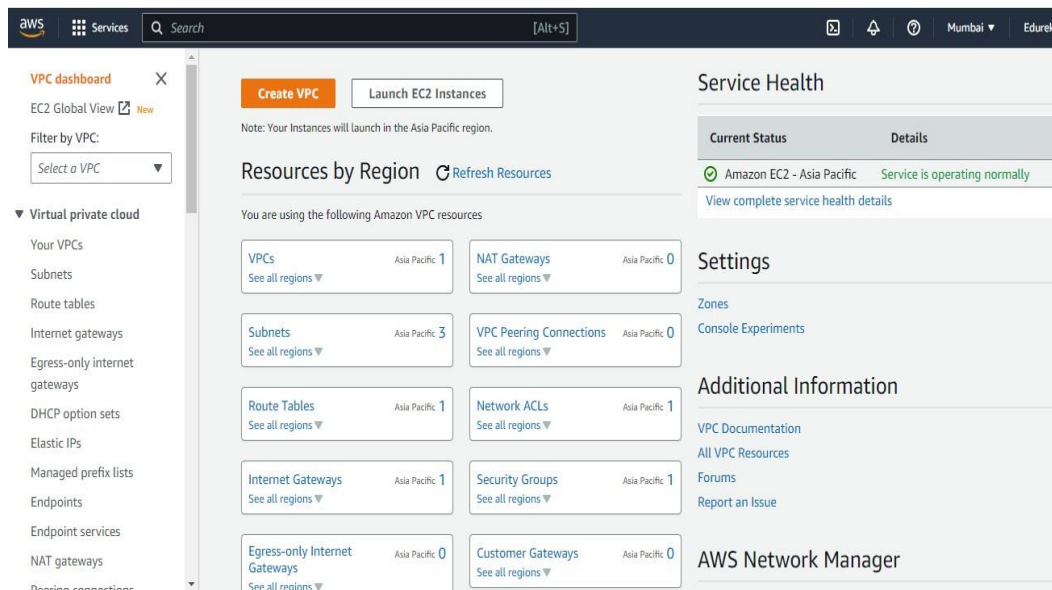


Figure 4.1: VPC dashboard

Create our own VPC provide the name and CIDR value as 0.0.0.0/16 with default tenancy

VPC > Your VPCs > Create VPC

### Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

#### VPC settings

**Resources to create** Info  
Create only the VPC resource or the VPC and other networking resources.

☒ VPC only ☐ VPC and more

**Name tag - optional**  
Creates a tag with a key of 'Name' and a value that you specify.

edurekavpc

**IPv4 CIDR block** Info

☒ IPv4 CIDR manual input ☐ IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.0.0.0/16

**IPv6 CIDR block** Info

☒ No IPv6 CIDR block ☐ IPAM-allocated IPv6 CIDR block ☐ Amazon-provided IPv6 CIDR block ☐ IPv6 CIDR owned by me

**Tenancy** Info

Default

#### Tags

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

**Key**

**Value - optional**

You can add 49 more tags.

Figure 4.2: Create VPC

VPC > Your VPCs > Create VPC > Create VPC resources

### Create VPC workflow

**Creating VPC Resources**

Thank you for using the new create VPC experience. Let us know what you think.

Attach internet gateway to the VPC 48%

**Details**

- ✔ Create VPC: vpc-00181f0398e171ce6 [Link](#)
- ✔ Enable DNS hostnames
- ✔ Enable DNS resolution
- ✔ Verifying VPC creation: vpc-00181f0398e171ce6 [Link](#)
- ✔ Create S3 endpoint: vpce-0da83b36458536e80 [Link](#)
- ✔ Create subnet: subnet-065dfde5a79b8b67e [Link](#)
- ✔ Create subnet: subnet-0eef6cbd85ba34ac7 [Link](#)
- ✔ Create subnet: subnet-077feb44e92bb54f5 [Link](#)
- ✔ Create subnet: subnet-02e1c040c8cf09493 [Link](#)
- ✔ Create internet gateway: igw-0e99f1505cf66ac62 [Link](#)
- ⚙ Attach internet gateway to the VPC
- ⚙ Create route table
- ⚙ Create route
- ⚙ Associate route table
- ⚙ Associate route table
- ⚙ Create route table
- ⚙ Associate route table
- ⚙ Create route table
- ⚙ Associate route table

Figure 4.3: VPC creating resources

Next step in creating vpc is to launch the subnet and we need two subnet on private for Database and Public for Backend API.

In the public subnet we have to launch under our own created VPC in with CIDR as 10.0.1.0/24 And Private subnet with CIDR as 10.0.2.0/24

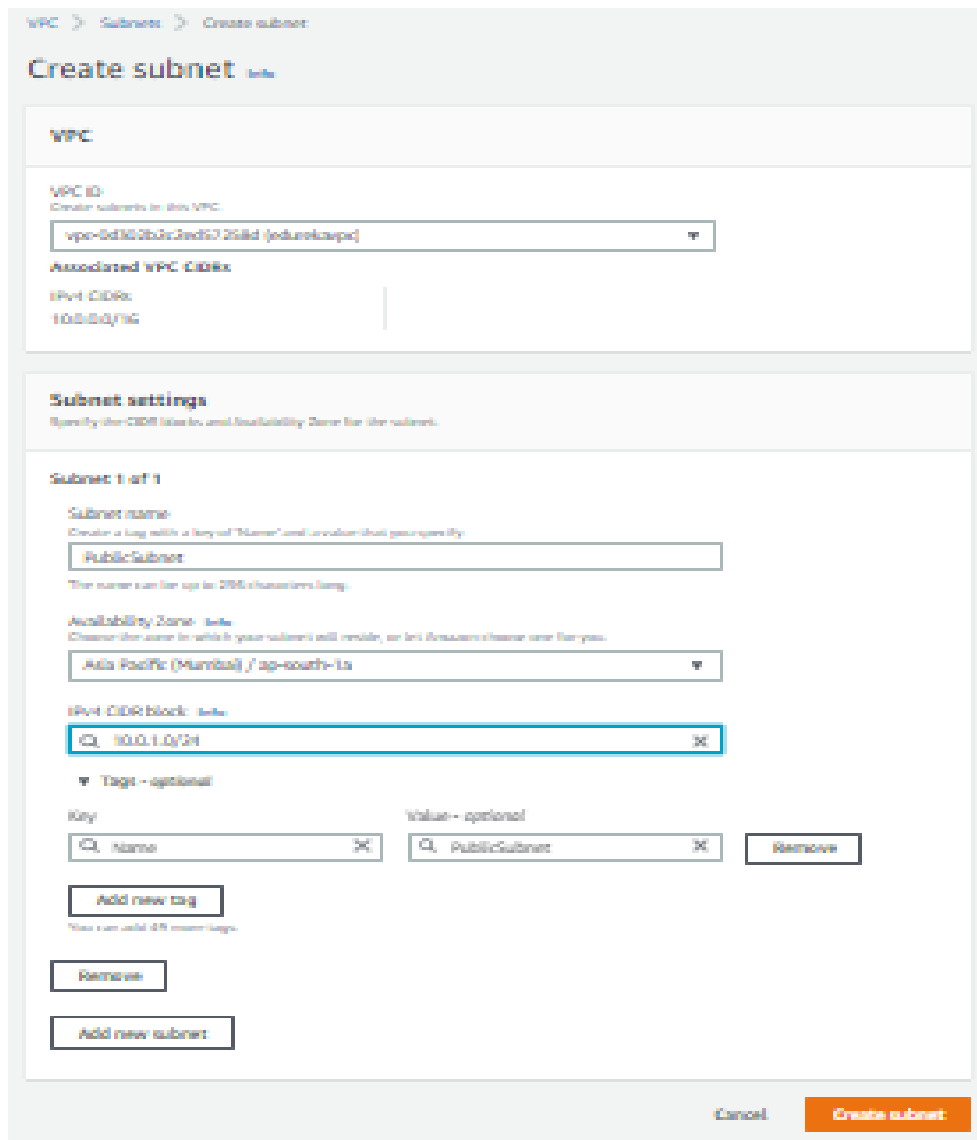


Figure 4.4: Create Public Subnet

Figure 4.5: Create Private Subnet

securing our subnet to connect  
private route table inside our o

VPC > Route tables > Create route table

## Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

### Route table settings

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.

**VPC**  
The VPC to use for this route table.

### Tags

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

Key	Value - optional	
<input type="text" value="Q Name"/>	<input type="text" value="Q PublicRT"/>	<input type="button" value="Remove"/>
<input type="button" value="Add new tag"/>		

You can add 49 more tags.

Figure 4.6: Create route table for Public

VPC > Route tables > Create route table

## Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

### Route table settings

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.

**VPC**  
The VPC to use for this route table.

### Tags

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

Key	Value - optional	
<input type="text" value="Q Name"/>	<input type="text" value="Q PrivateRT"/>	<input type="button" value="Remove"/>
<input type="button" value="Add new tag"/>		

You can add 49 more tags.

Figure 4.7: Create route table for Private

We have to work on now subnet association with both public route table and private route table.

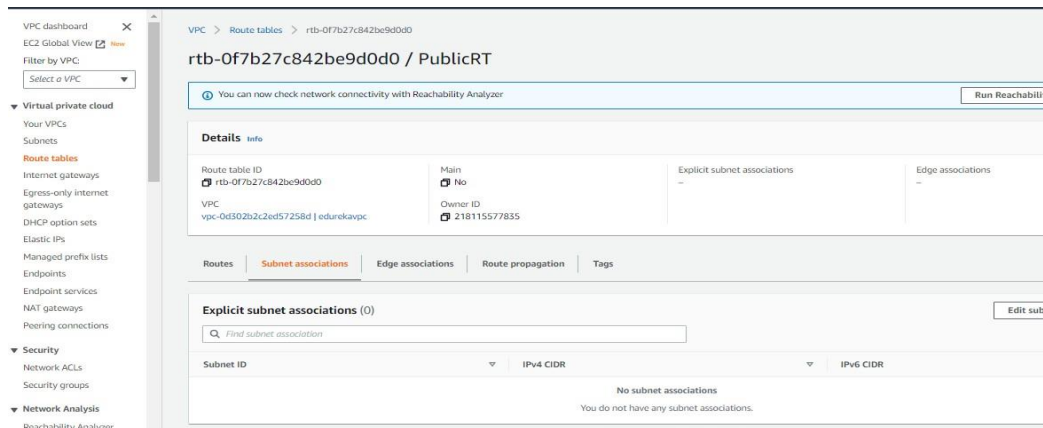


Figure 4.8: Explicit Subnet association

Link public subnet with public route table in the select list we need to select our subnet and click checkbox for association.

Similarly associate private subnet with private route table.

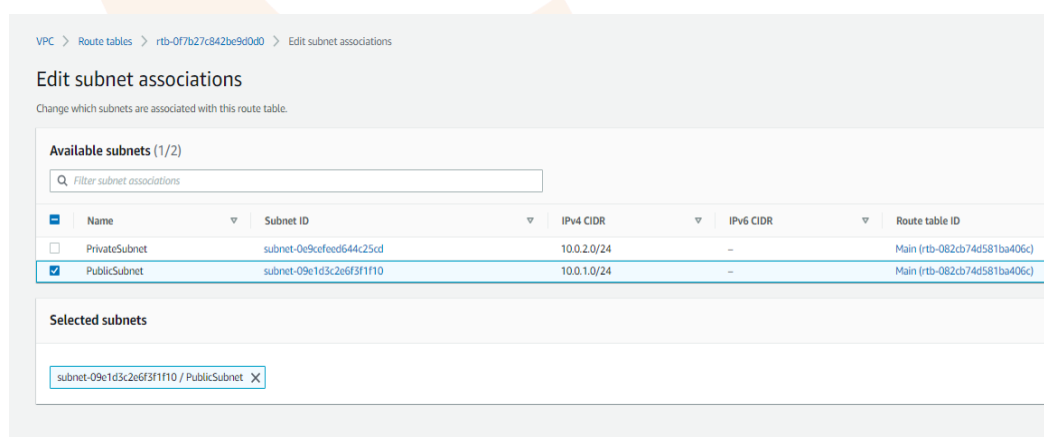


Figure: 4.9: Subnet associations

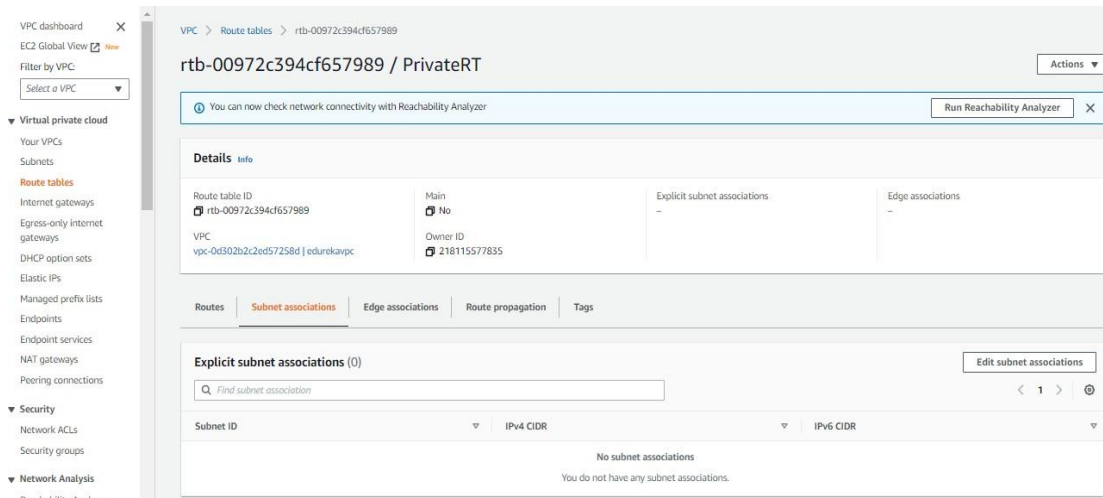


Figure: 4.10: Explicit Subnet associations

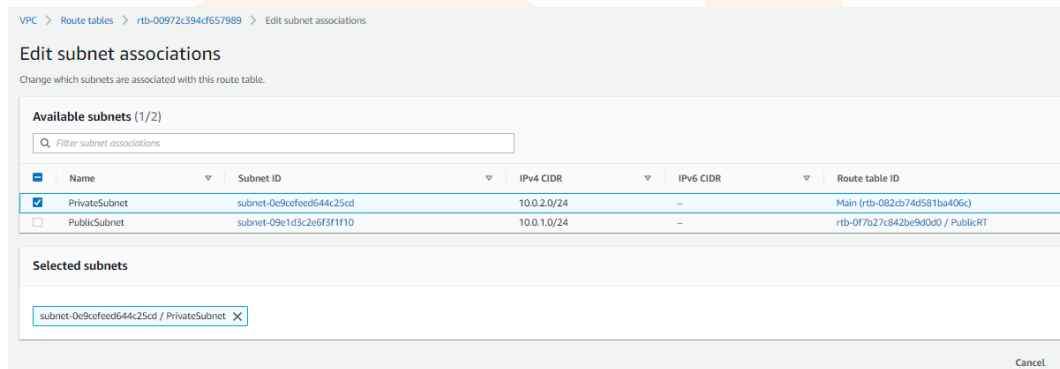


Figure: 4.11: Subnet associations

Now we need to add one internet gateway to the VPC to make it available outside as the public URL.

Select internet gateway and click on action and attach to VPC which you have created.



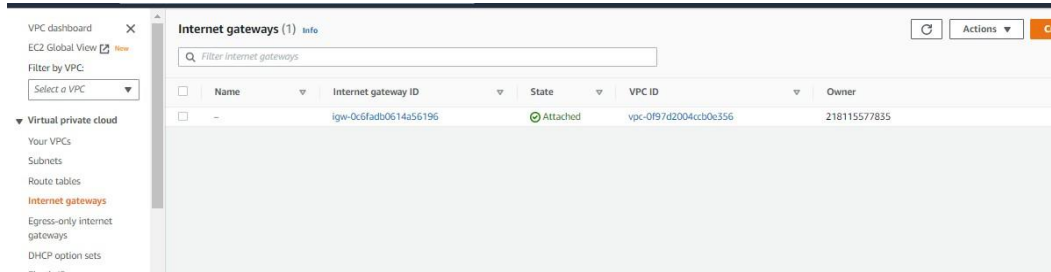


Figure 4.12 Attach Internet Gateway

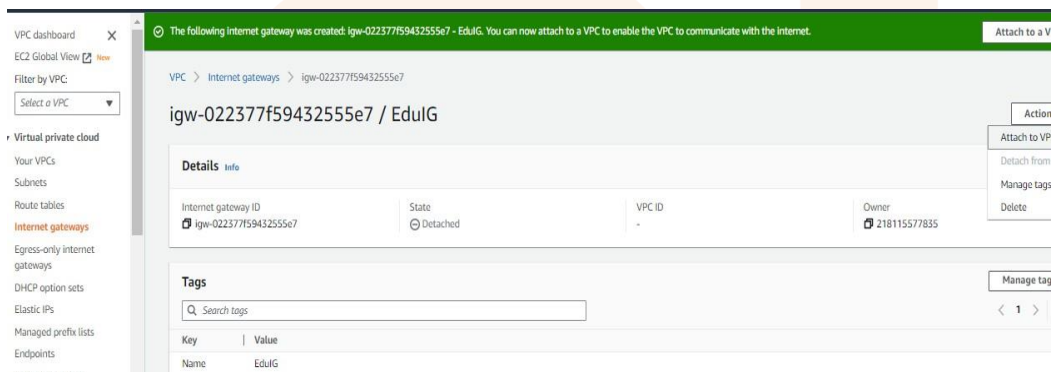


Figure 4.13: Attaching Internet gateway to VPC

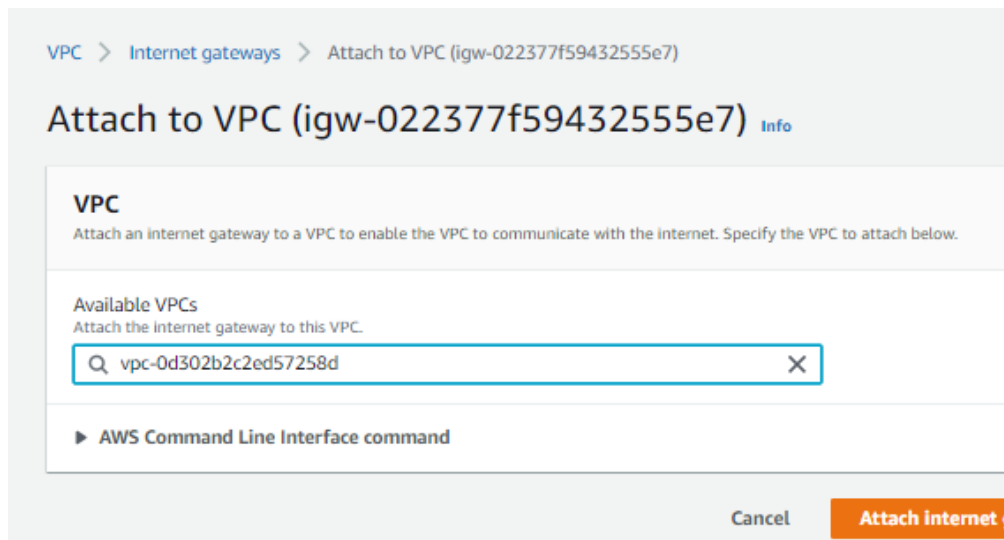


Figure 4.14: Attaching Internet gateway to VPC

By this we are ready for with our VPC configuration with on public and private subnet Now we need one server to deploy the application for our frontend part of application.

So, we are going to launch the elastic beanstalk container where we will deploy our app with continuous deployment mode

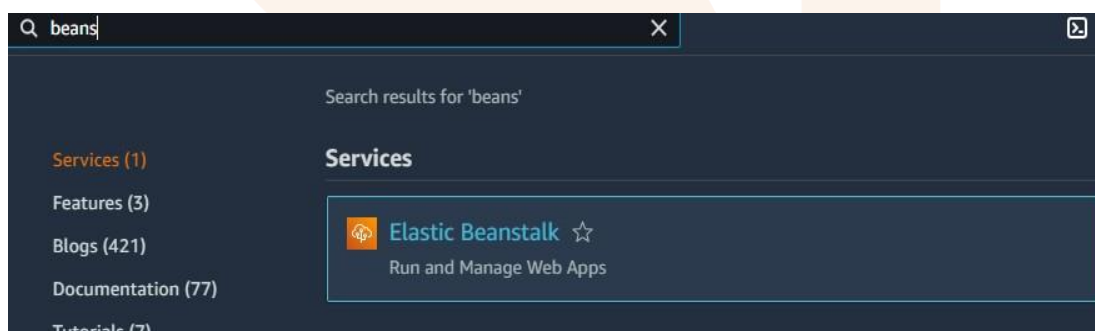


Figure 4.15: Search Elastic Beanstalk

**Elastic Beanstalk** ×

**Create a web app**  
Create a new application and environment with a sample application or your own code. By creating an environment, you allow Amazon Elastic Beanstalk to manage Amazon Web Services resources and permissions on your behalf. [Learn more](#)

**Application information**

Application name  
EduFront  
Up to 128 Unicode characters, not including forward slash (/).

**Application tags**

Apply up to 50 tags. You can use tags to group and filter your resources. A tag is a key-value pair. The key must be unique within the resource and is case-sensitive. [Learn more](#)

[Add tag](#)  
0/50 remaining

**Platform**

Platform  
Node.js ▼

Platform branch  
Node.js 16 running on 64bit Amazon Linux 2 ▼

Platform version  
5.6.3 (Recommended) ▼

**Application code**

☒ **Sample application**  
Get started right away with sample code.

☐ **Upload your code**  
Upload a source bundle from your computer or copy one from Amazon S3.

[Cancel](#) [Configure more options](#) [Create application](#)

Figure 4.16: Create a Web app in Elastic Beanstalk

This will launch one instance over Ec2 as pass service where we can deploy our frontend

Now create a pipe for continuous integration i.e. CI/CD using Codepipeline

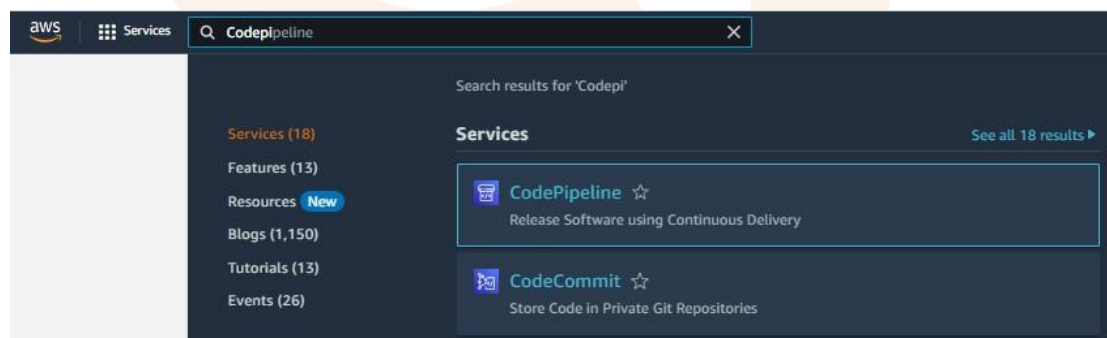


Figure 4.17: Search a CodePipeline

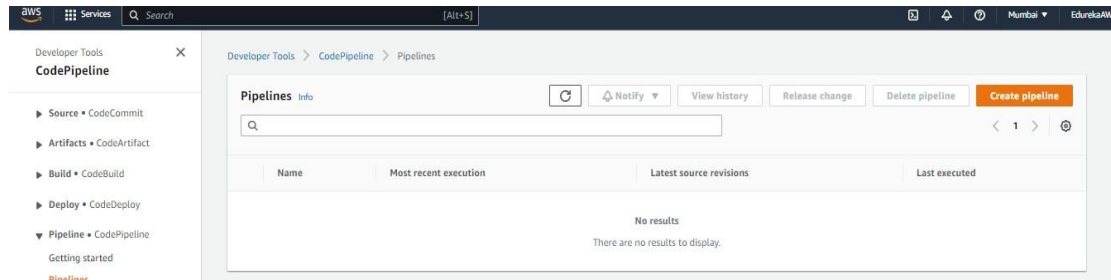


Figure 4.18: Create a CodePipeline

Provide the name to code pipeline for frontend app

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1  
**Choose pipeline settings**

Step 2  
Add source stage

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

### Choose pipeline settings Info

#### Pipeline settings

**Pipeline name**  
Enter the pipeline name. You cannot edit the pipeline name after it is created.

eduCiCD

No more than 100 characters

**Service role**

☒ New service role  
Create a service role in your account

☐ Existing service role  
Choose an existing service role from your account

**Role name**

AWSCodePipelineServiceRole-ap-south-1-eduCiCD

Type your service role name

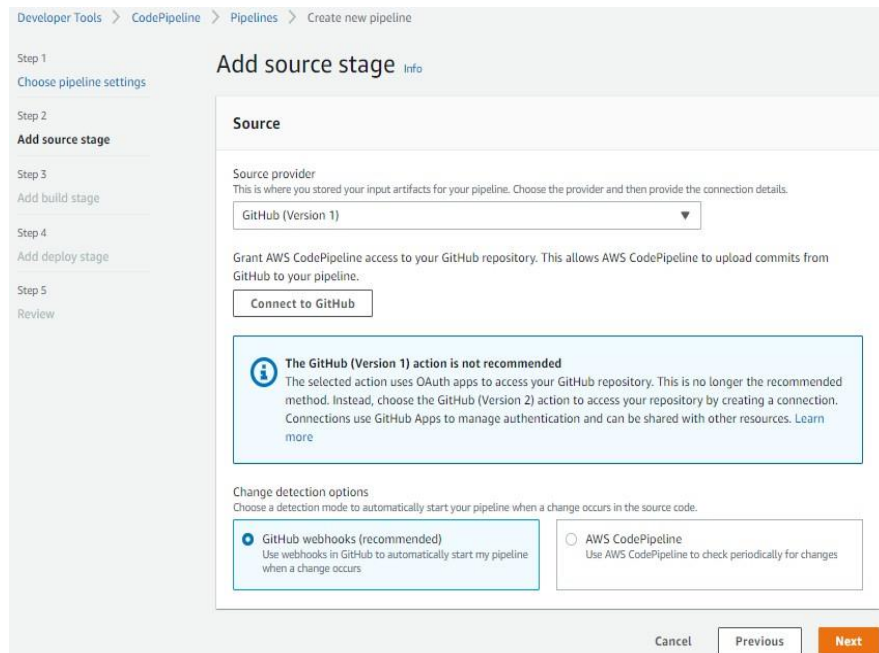
☒ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

► Advanced settings

Cancel Next

Figure 4.19: Configure the CodePipeline

Now in next step we will select the source from where we have to pick the code and deploy.



Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1  
Choose pipeline settings

Step 2  
**Add source stage**

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

### Add source stage Info

#### Source

Source provider  
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (Version 1) ▼

Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.

[Connect to GitHub](#)

**The GitHub (Version 1) action is not recommended**

The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (Version 2) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. [Learn more](#)

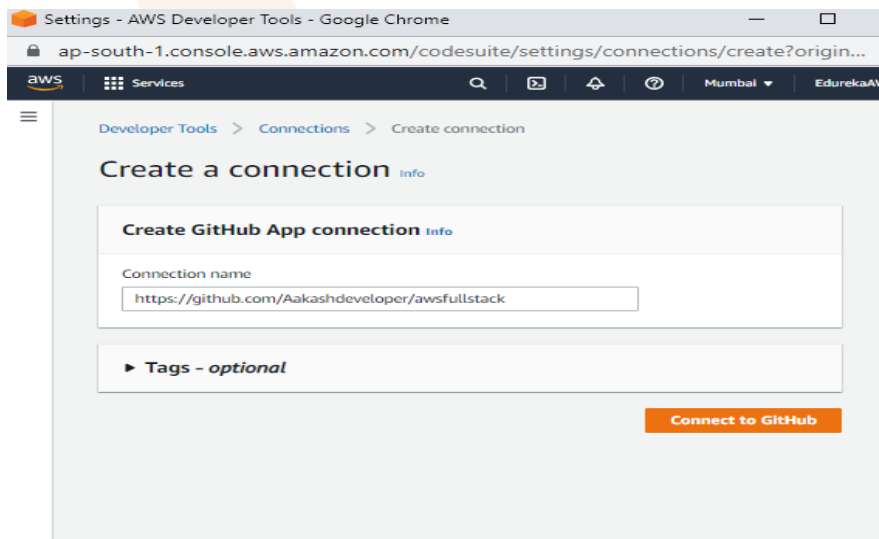
Change detection options  
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

☒ **GitHub webhooks (recommended)**  
Use webhooks in GitHub to automatically start my pipeline when a change occurs

☐ **AWS CodePipeline**  
Use AWS CodePipeline to check periodically for changes

[Cancel](#) [Previous](#) [Next](#)

Figure 4.20: Add the Source Provider



Settings - AWS Developer Tools - Google Chrome

ap-south-1.console.aws.amazon.com/codesuite/settings/connections/create?origin...

aws Services

Developer Tools > Connections > Create connection

### Create a connection Info

#### Create GitHub App connection Info

Connection name

https://github.com/Aakashdeveloper/awsfullstack

► **Tags - optional**

[Connect to GitHub](#)

Figure 4.21: Create GitHub connection

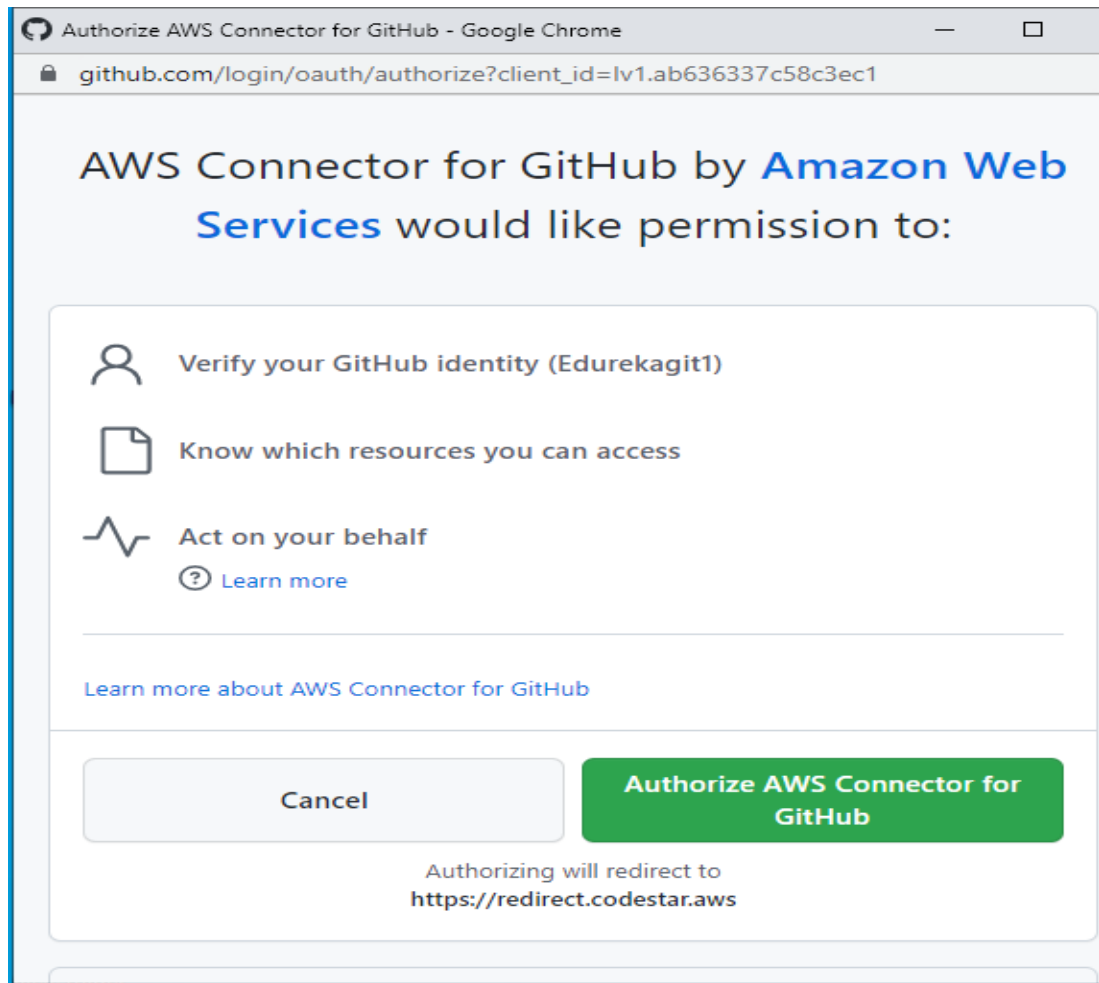


Figure 4.22: Authorize AWS connector for GitHub

Select the repo which you want to link for code pipeline and select the branch.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1  
Choose pipeline settings

Step 2  
**Add source stage**

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

### Add source stage Info

**Source**

**Source provider**  
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (Version 1) ▼

Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.

Connected

✔ You have successfully configured the action with the provider. ✕

**The GitHub (Version 1) action is not recommended**  
The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (Version 2) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. [Learn more](#)

**Repository**  
Edurekagit1/Aakashdeveloper-awsfullstack ✕

**Branch**  
master ✕

**Change detection options**  
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

☒ **GitHub webhooks (recommended)**  
Use webhooks in GitHub to automatically start my pipeline when a change occurs

☐ **AWS CodePipeline**  
Use AWS CodePipeline to check periodically for changes

Cancel Previous **Next**

Figure 4.23: Select the repository and branch

Frontend code is available over the repo: <https://github.com/Edurekagit1/Aakashdeveloper-awsfullstack> This is the react frontend app we need to change the URL of backend python app.

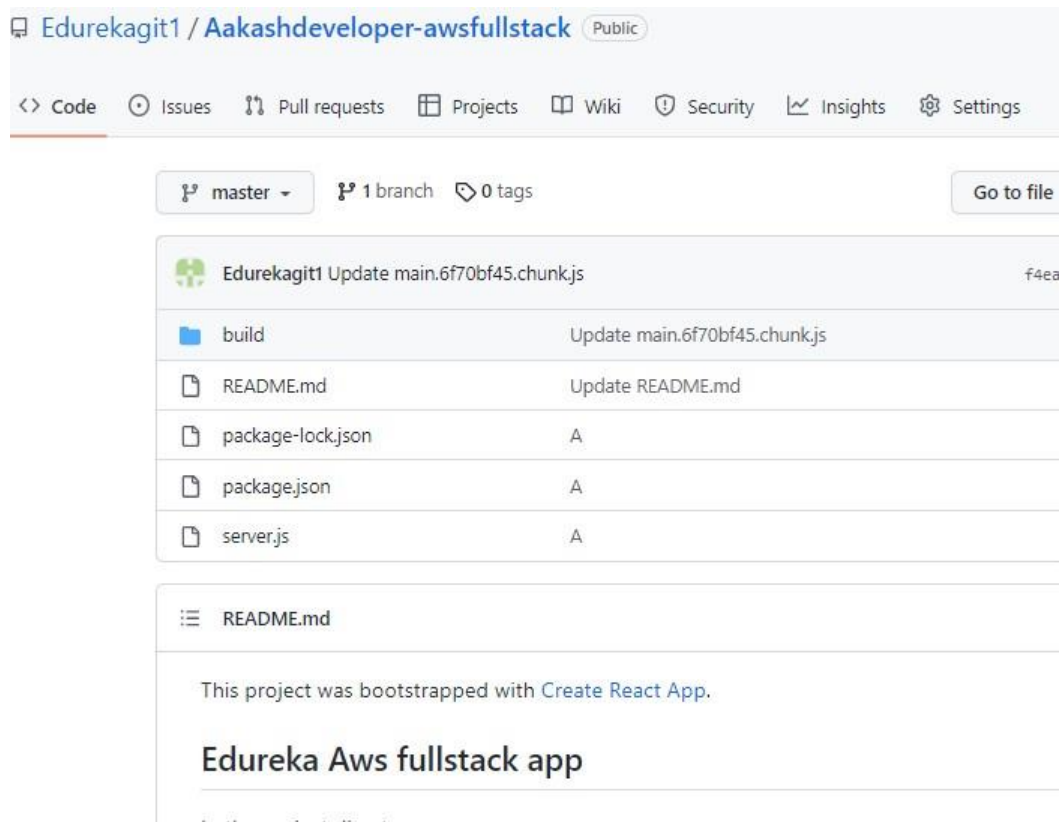


Figure 4.24: Source code in repository

For this demo we can skip the build stage of the application as it more like core script need to build app.

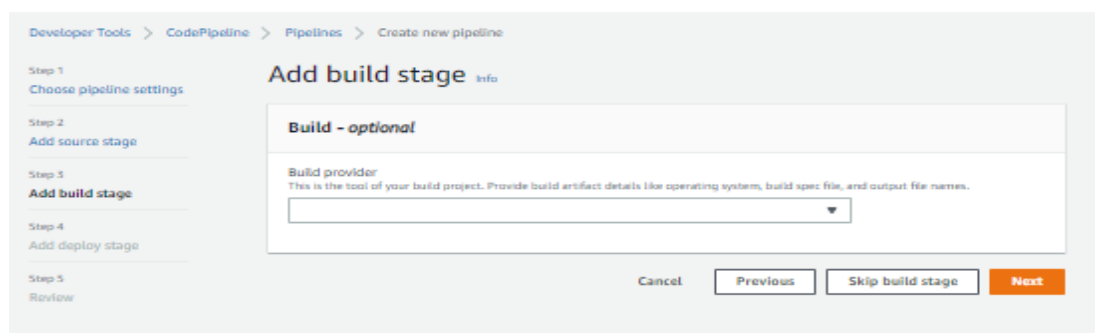


Figure 4.25: Add build stage



Now select the server where we want to deploy the application and i.e. beanstalk that we have launched in earlier step

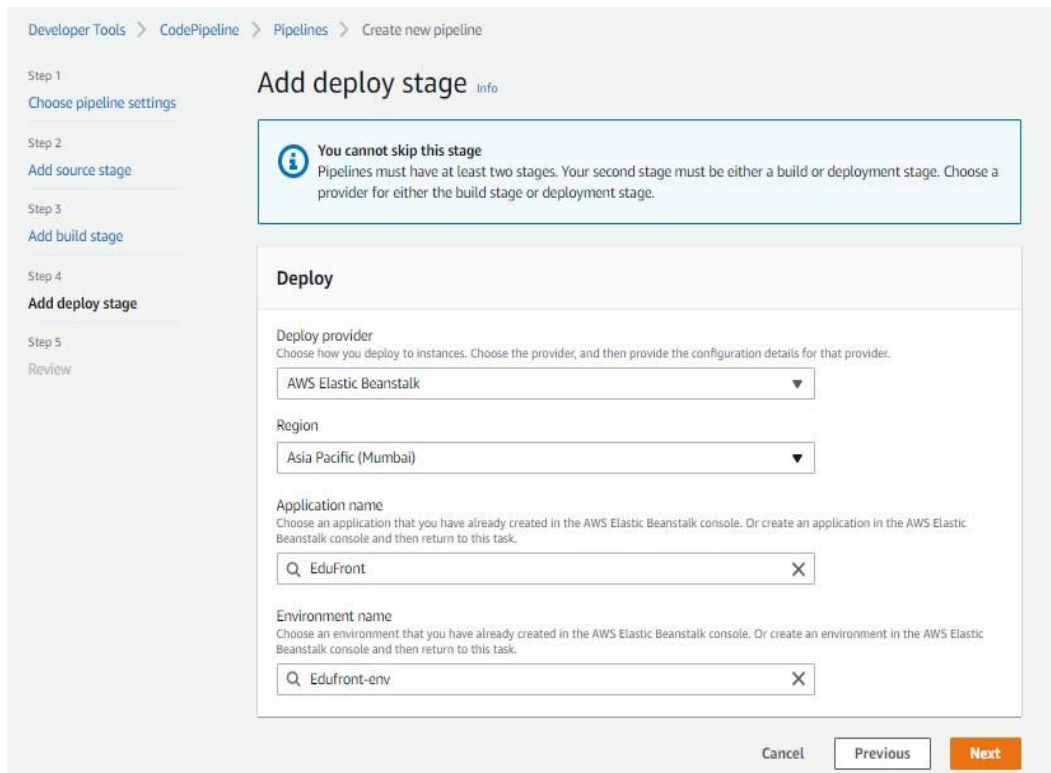


Figure 4.26: Add deploy stage

Once ready our code pipeline is ready to deploy our frontend on the bean stalk.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1: Create pipeline settings

Step 2: Add source stage

Step 3: Add build stage

Step 4: Add deploy stage

**Review**

### Review

Step 1: Choose pipeline settings

**Pipeline settings**

Pipeline name: edureka

Artifact location: A new Amazon S3 bucket will be created as the default artifact store for your pipeline

Service role name: AWSCodePipelineServiceRole-us-east-1-edureka

Step 2: Add source stage

**Source action provider**

Source action provider: GitHub (version 1)

ProviderSourceChanges: false

Repo: AskofDeveloper-questlabs

Owner: Gunkagiri

Branch: master

Step 3: Add build stage

**Build action provider**

Build stage: no build

Step 4: Add deploy stage

**Deploy action provider**

Deploy action provider: AWS Elastic Beanstalk

Application name: edureka

Environment name: edureka-env

Buttons: Cancel, Previous, Create pipeline

Figure 4.27: Review the Pipeline

As soon as the code commit on the master branch this pipeline will execute and deploy code on the server.

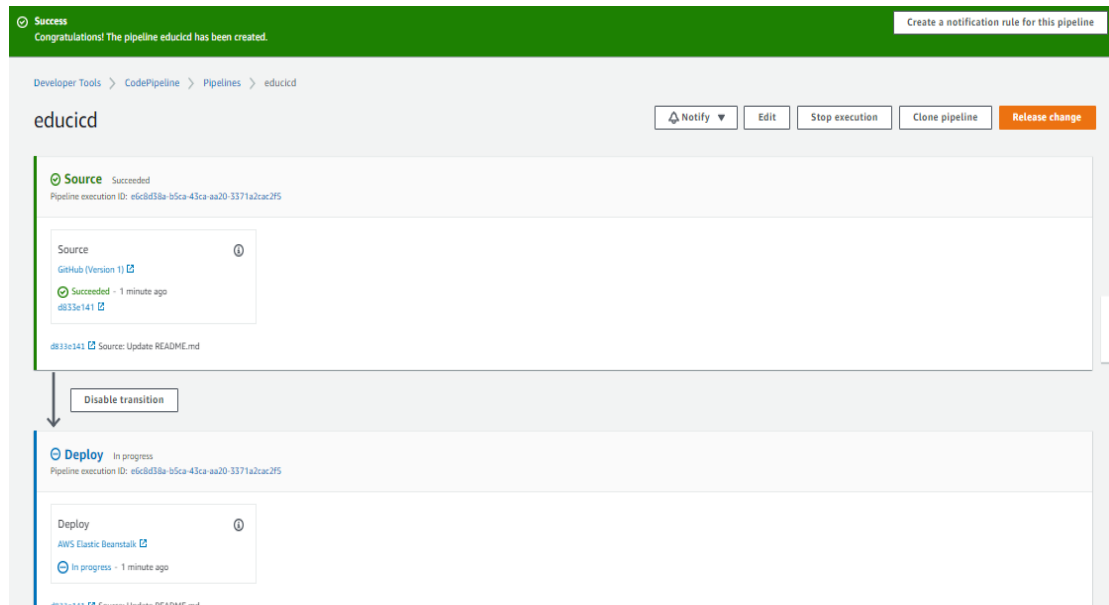


Figure 4.28: CodePipeline source to deploy transition

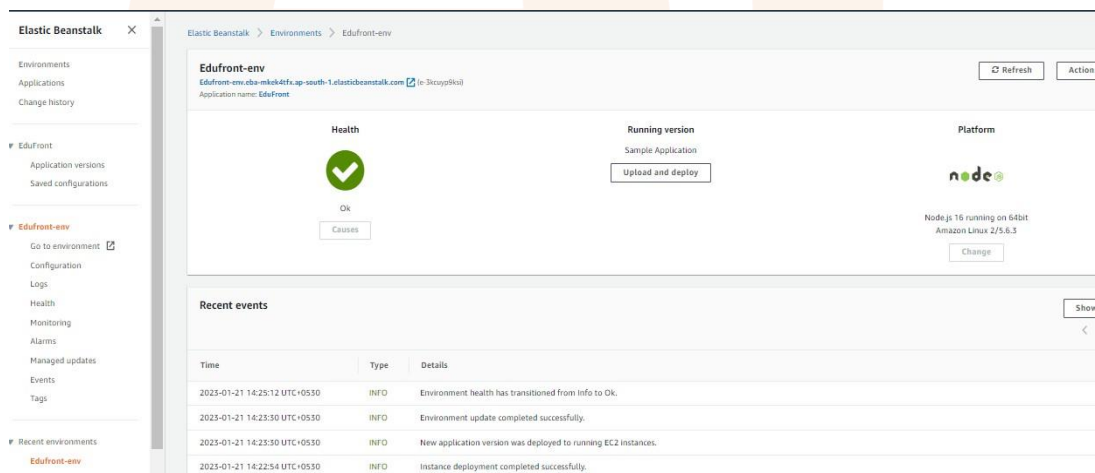


Figure 4.29: Triggered in the Beanstalk

It's time to start launch the Ec2 into our subnet for backend application inside our VPC

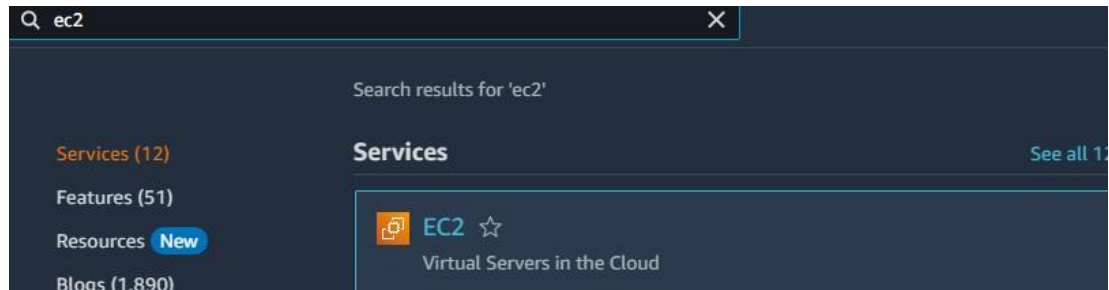


Figure 4.30: Search an EC2

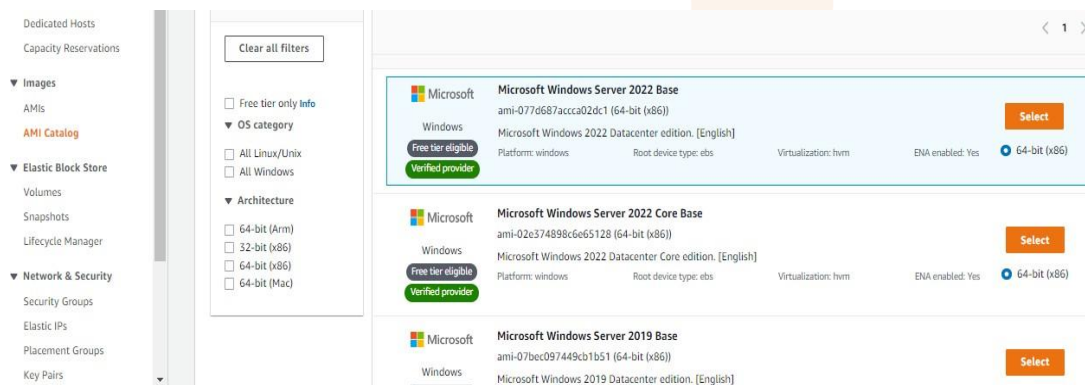


Figure 4.31: Select the AMI

In the configuration tab make sure to check the VPC select should be our VPC and subnet should be public.

**Instance summary for i-0060db2121fd41607 (backend)** Info  
Updated less than a minute ago

Instance ID i-0060db2121fd41607 (backend)	Public IPv4 address 35.154.220.142   open address
IPv6 address -	Instance state Running
Hostname type IP name: ip-10-0-1-227.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-10-0-1-227.ap-south-1.compute.inte
Answer private resource DNS name IPv4 (A)	Instance type t2.large
Auto-assigned IP address 35.154.220.142 (Public IP)	VPC ID vpc-d0302b2c2ed57258d (edureka/vpc)
IAM Role SSM_Role_EC2	Subnet ID subnet-09e1d3c2e6f3f1110 (PublicSubne

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

You can now check network connectivity with Reachability Analyzer.

**Networking details** Info

Public IPv4 address 35.154.220.142   open address	Private IPv4 addresses 10.0.1.227
Public IPv4 DNS ec2-35-154-220-142.ap-south-1.compute.amazonaws.com   open address	Private IP DNS name (IPv4 only) ip-10-0-1-227.ap-south-1.compute.inte
<b>Subnet ID</b> subnet-09e1d3c2e6f3f1110 (PublicSubnet)	IPv6 addresses -
Availability zone ap-south-1a	Carrier IP addresses (ephemeral) -

Figure 4.32: Backend instance configuration

**Instances (8)** Info

Find instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availabil
backend	i-0060db2121fd41607	Running	t2.large	2/2 checks passed	No alarms	ap-south
Edufront-env	i-0208b2ae35c6c84df	Running	t2.micro	2/2 checks passed	No alarms	ap-south

Figure 4.33: Backend instance configuration

Connect to backend instance using the RDP client through fleet manager remote desktop

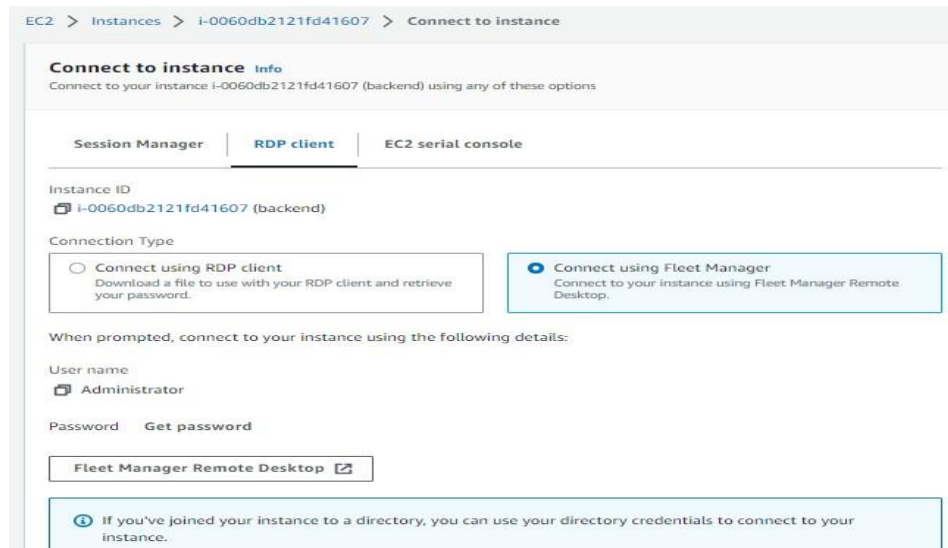


Figure 4.34: Connect using Fleet manager

Once by frontend and backend is ready, we have to launch our database (RDS) in private subnet

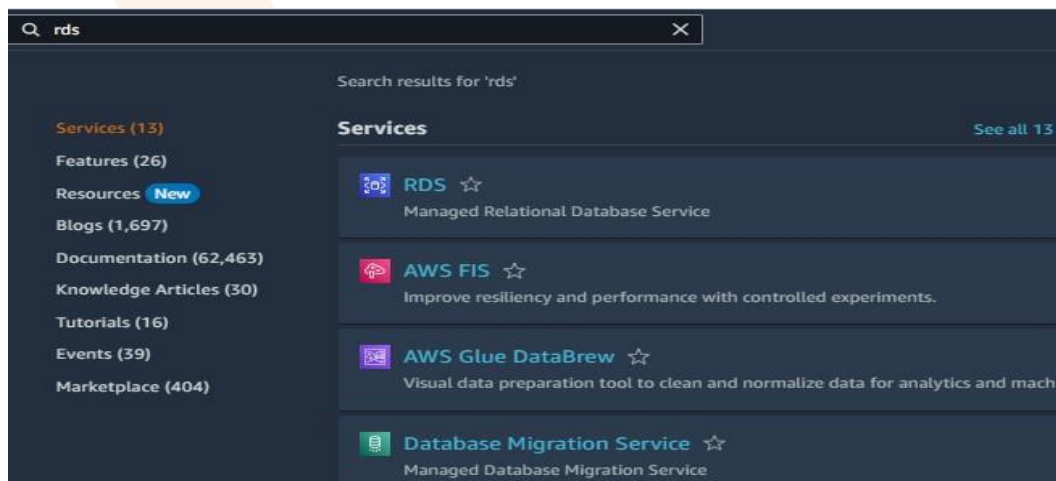
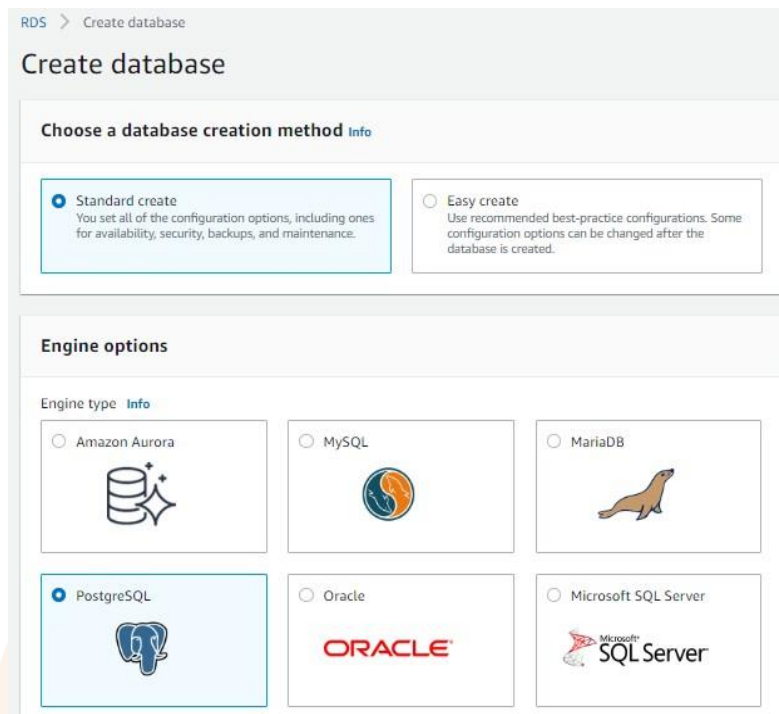


Figure 4.35: Search the RDS

Select the database engine and database creation method for our application.



The screenshot shows the AWS RDS 'Create database' console. At the top, it says 'RDS > Create database'. The main heading is 'Create database'. Below this, there's a section 'Choose a database creation method' with two options: 'Standard create' (selected) and 'Easy create'. The 'Standard create' option is highlighted with a blue border and a blue radio button. Below this, there's a section 'Engine options' with a sub-section 'Engine type'. There are six database engine options displayed in a grid: Amazon Aurora, MySQL, MariaDB, PostgreSQL (selected), Oracle, and Microsoft SQL Server. Each option has a radio button and a logo. The 'PostgreSQL' option is highlighted with a blue border and a blue radio button.

Figure 4.36: Create a database

While launching make sure that we have select our own created VPC for security of DB Server



### Settings

**DB instance identifier** [Info](#)  
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.  
  
The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

**▼ Credentials Settings**

**Master username** [Info](#)  
Type a login ID for the master user of your DB instance.  
  
1 to 16 alphanumeric characters. First character must be a letter.

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

☐ **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](#)

Figure 4.37: Configure the Database



**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.

**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

edu\_backend\_db\_sg

**i** Amazon RDS will add a new VPC security group `rds-ec2-1` to allow connectivity with your compute resource.

**Availability Zone** [Info](#)

ap-south-1a

**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)

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

Figure 4.38: Create a new Group Name

RDS will create a new sg for backend ec2 instance to allow connectivity with db instance we created.

Connect to backend instance using fleet manager by installing SSM agent through IAM role.

## Fleet Manager - Remote Desktop

You can connect to a maximum of 4 nodes in this view.

**All sessions** | I-0060db2121fd41607

Node ID	Node name	Session time remaining	
I-0060db2121fd41607	backend	-	Close

**Authentication type**  
The type of authentication to use when connecting to the node. [Learn more](#)

☐ **User credentials**  
Username and password.

☒ **Key pair**  
Connect as Administrator using EC2 key pair.

**Administrator account name**  
The default administrator account name might vary based on your locale.

Administrator

**Key pair associated with the instance**  
**windows**

**Key pair content**  
Select a method for uploading the key pair content.

☒ **Browse your local machine to select the key pair file.**  
The private key file content is automatically uploaded to your browser.

☐ **Paste key pair content**  
Copy and paste the key pair content into the field below.


 **Browse**

Figure 4.39: Set-up the Authentication Type



Figure 4.40: Windows Remote Desktop- Fleet Manager

Over the window instance install python for running our backend application

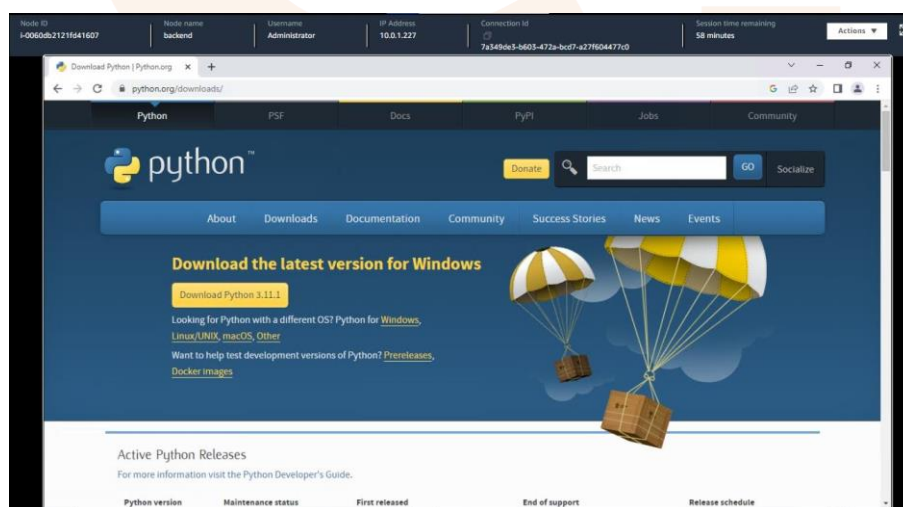
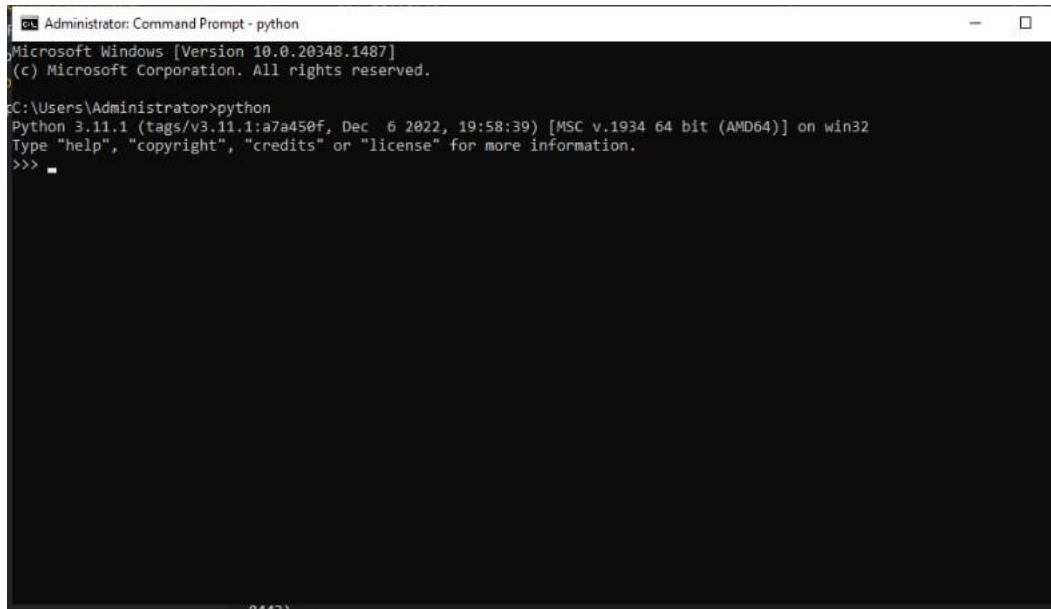


Figure 4.41: Download the Python

Now navigate to command prompt inside instance and type python to make sure python is running.



```
Administrator: Command Prompt - python
Microsoft Windows [Version 10.0.20348.1487]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>python
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

Figure 4.42: Command Prompt- python

Now navigate to <https://github.com/Aakashdeveloper> And open Python-aws-rds repo and clone the code

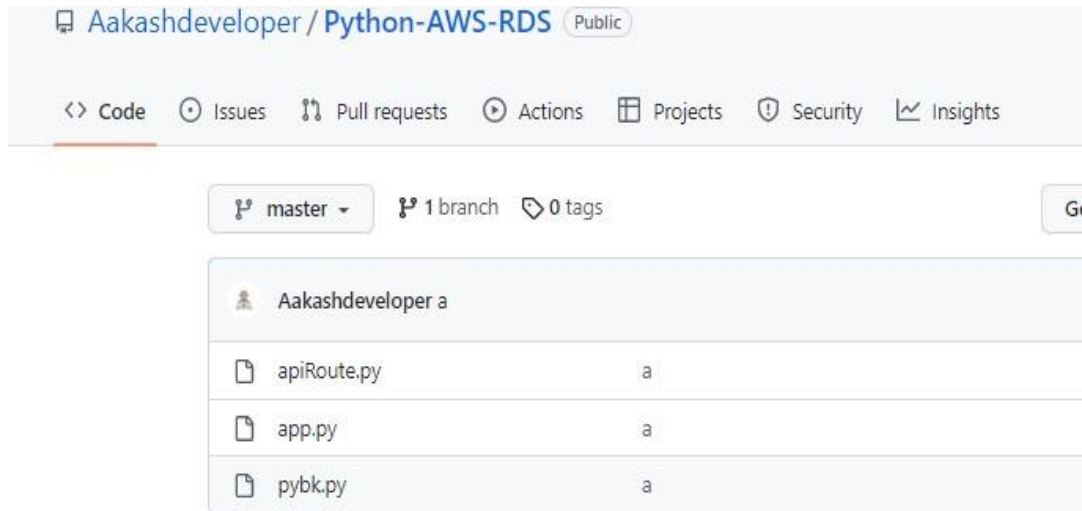


Figure 4.43: Command Prompt- python

Now as we have to connect to DB instance from the window server. For postgres we have to install pgadmin on the window server. <https://www.pgadmin.org/download/>

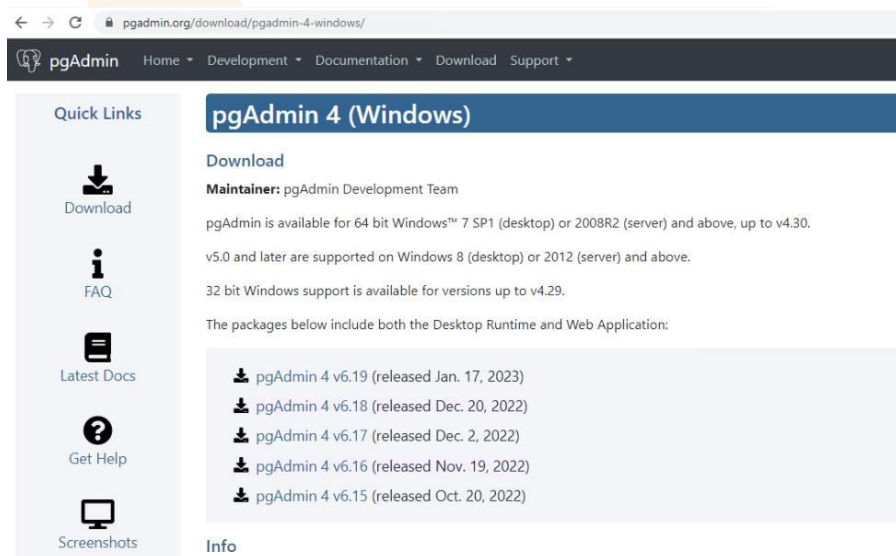


Figure 4.44: Download pg admin

Visual studio code is also required for code editing and checking python code.

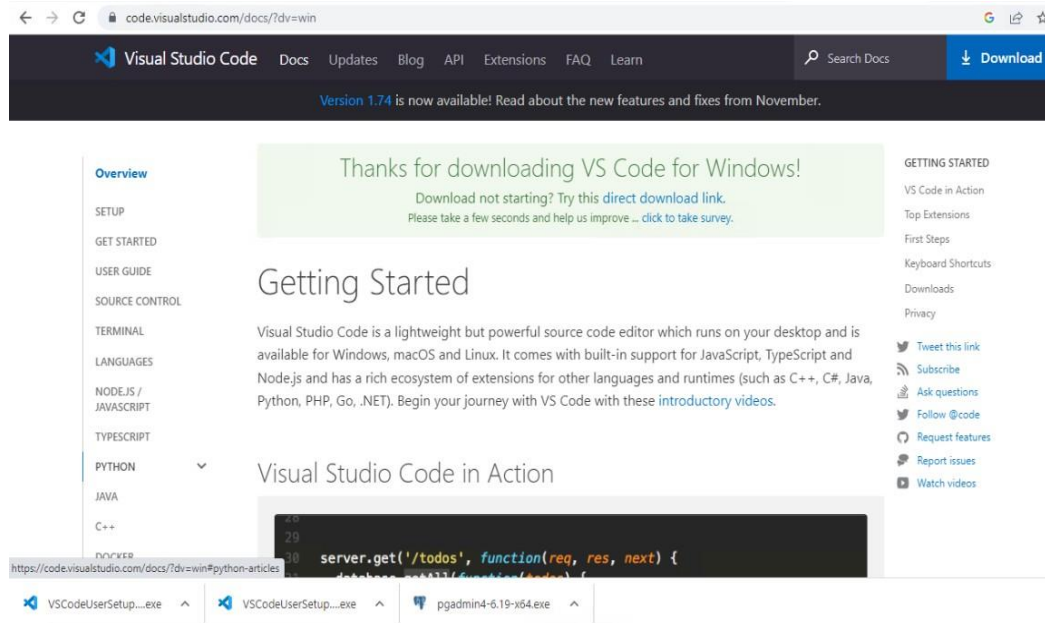


Figure 4.45: Download Visual Studio

On Line number 26 in apiRoute.py file we need to add credentials for our DB which we have launched change host, port ,password and dbname.



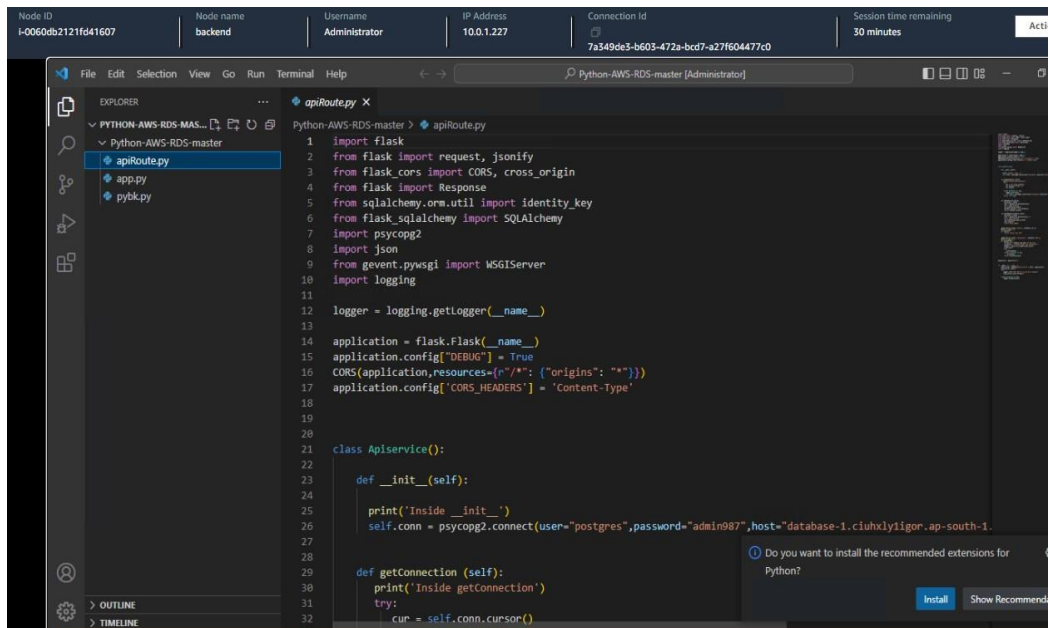


Figure 4.46: Visual Studio – Python AWS RDS Master

Add Select \* from restaurant in line 69

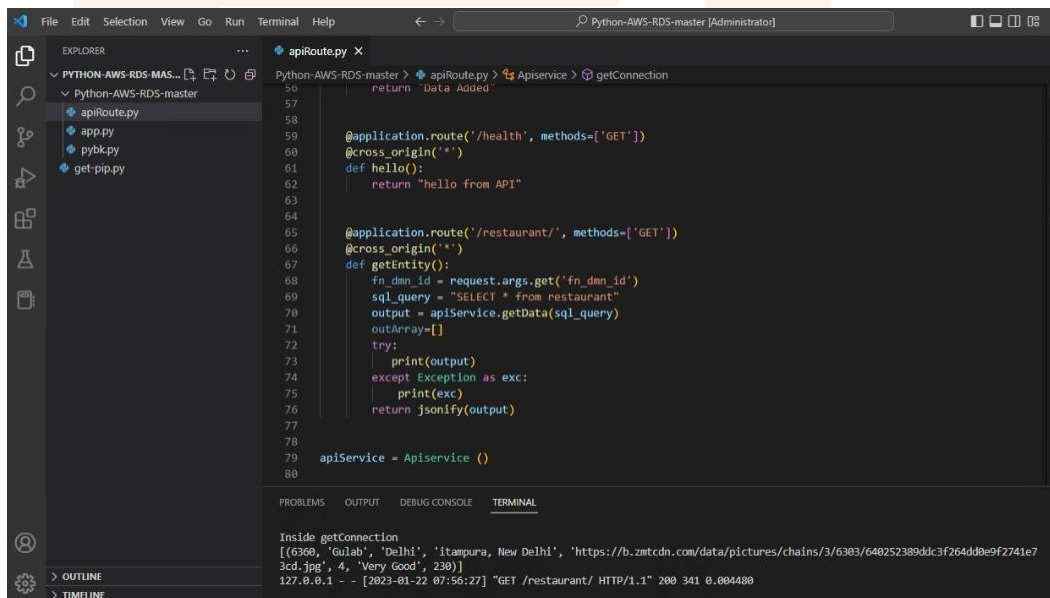


Figure 4.47: Check the terminal

We can get the credential for DB connection on the database page in connectivity section.

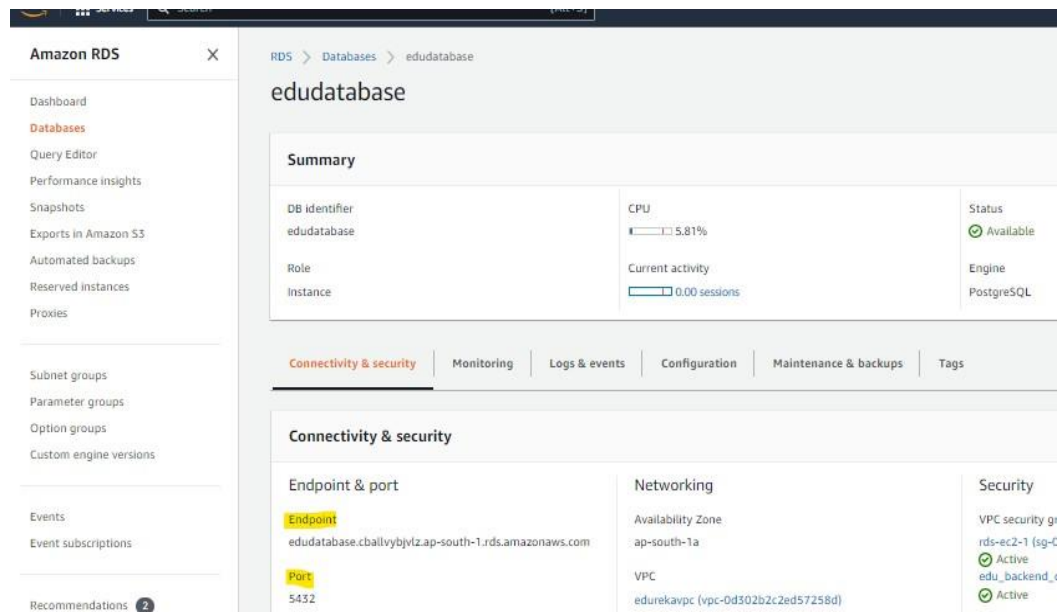


Figure 4.48: Note the RDS Endpoint and Port

Now on public Ec2 machine with pg admin login to postgres and create table and add column to the database.



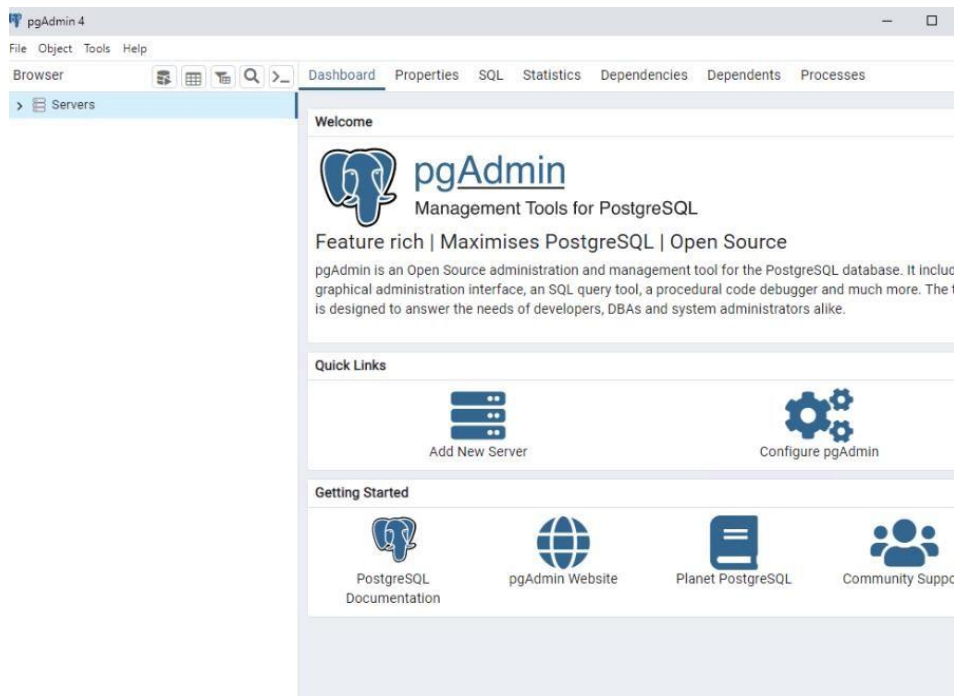


Figure 4.49: pgAdmin: Add New Server

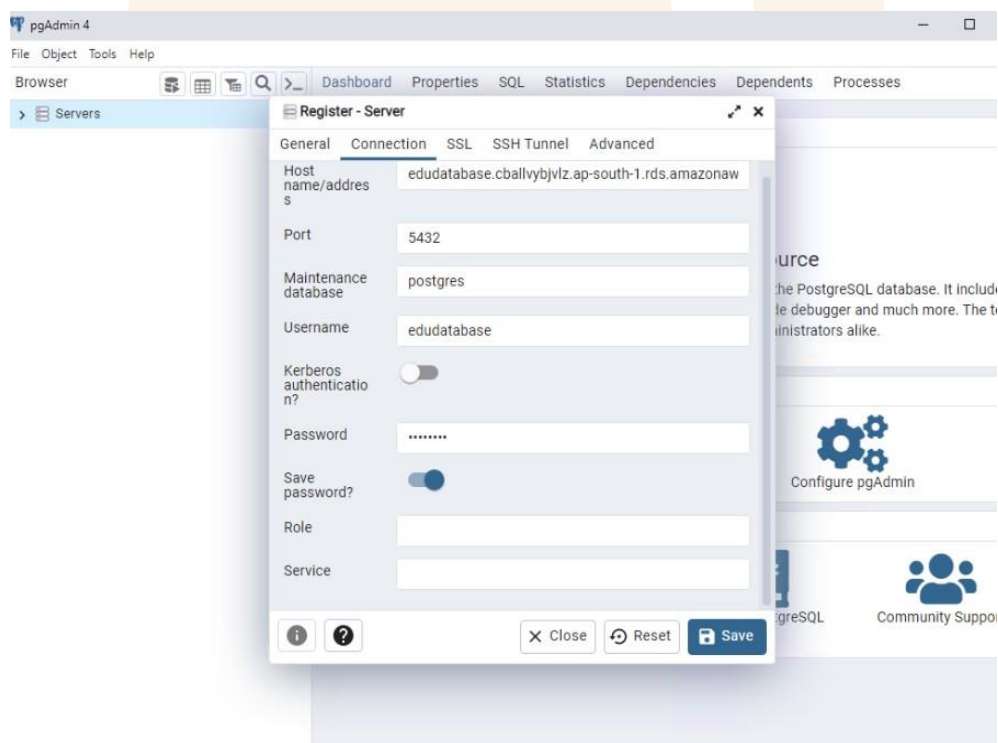


Figure 4.50: Register-Server Configure Connection

Add db credentials and details

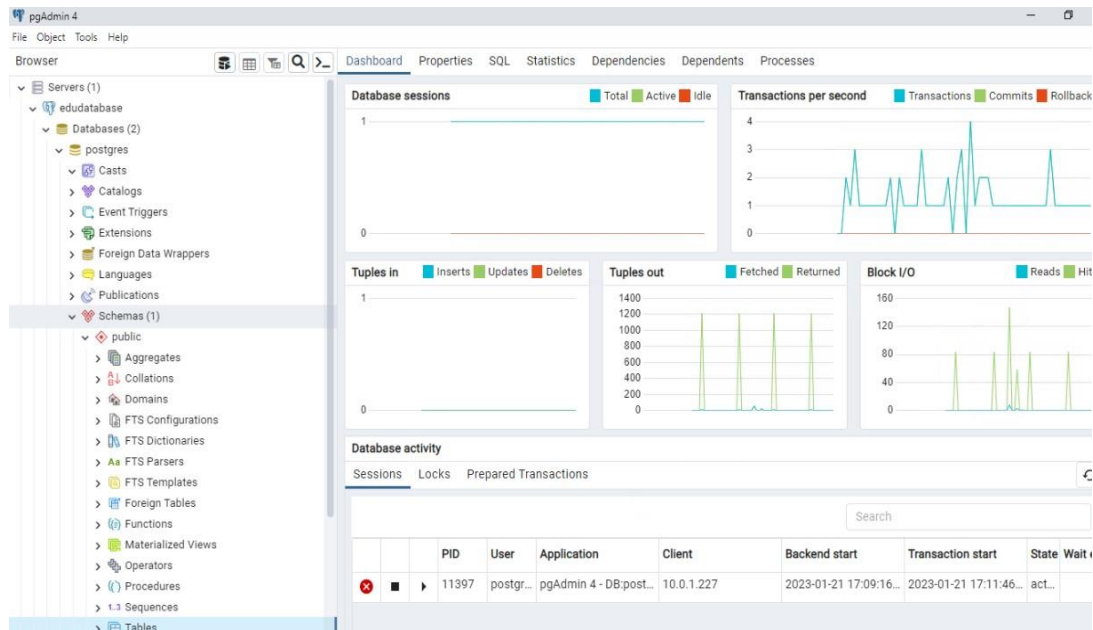


Figure 4.51: Select the Tables

Now we will create a table in the edudatabase

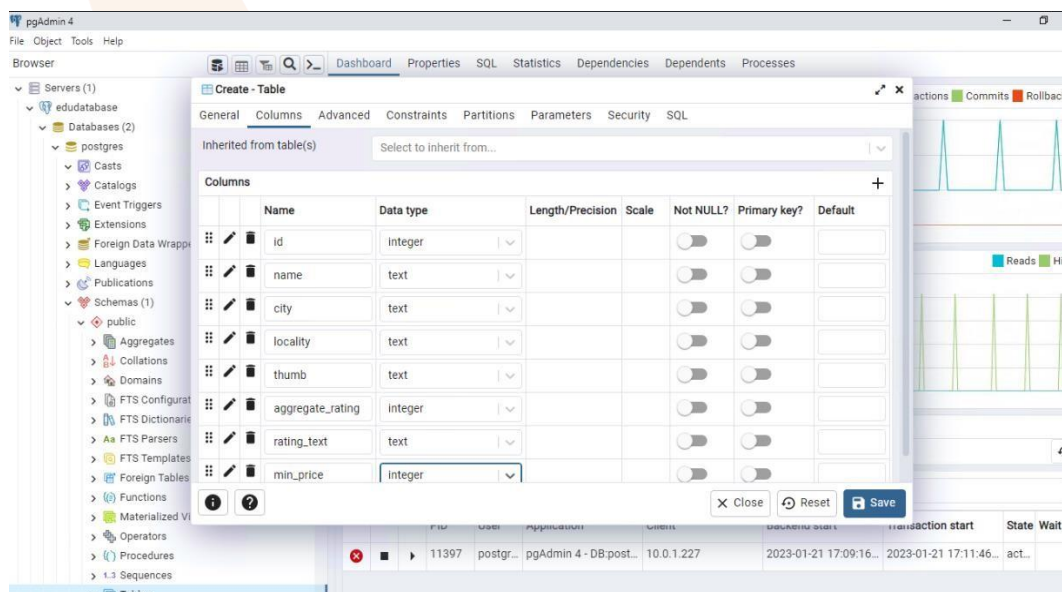


Figure 4.52: Create- Table

Insert the record into the table using pg admin console and verify the insertion with select query.

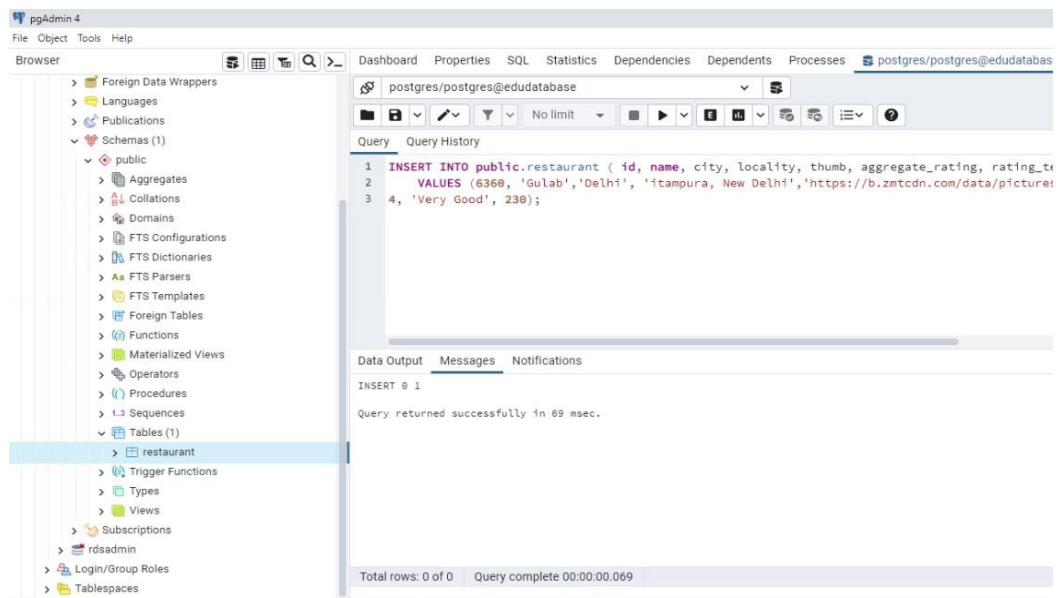


Figure 4.53: Run the Server

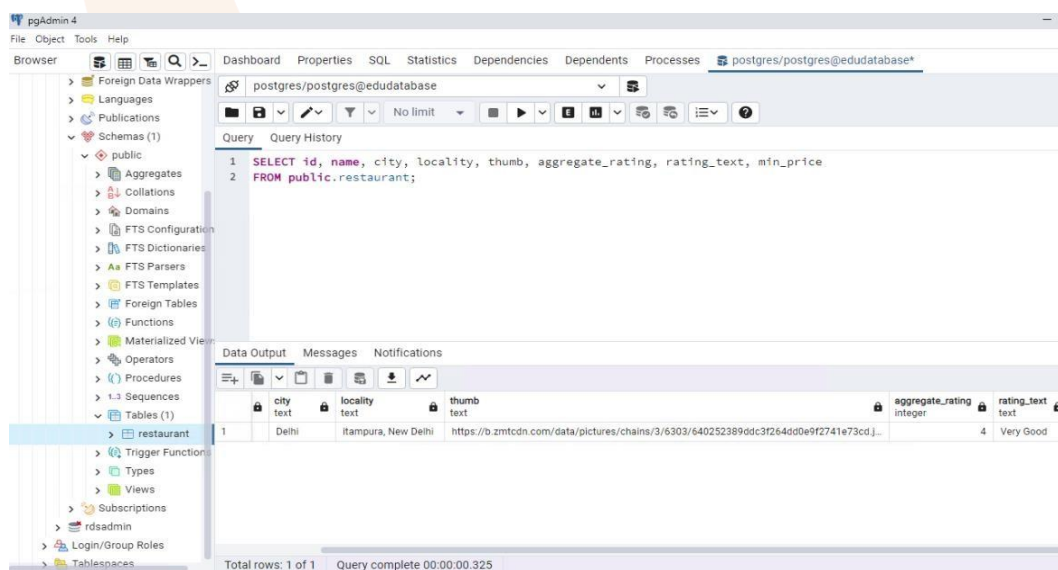


Figure 4.54: Run the query

Now we check our api code we can see the data inserted in private subnet into RDS is now accessible from the public Ec2.

Run this apiRoute.py code and check for **http://35.154.220.142:8443/restaurant/** in browser



Figure 4.55: Run the localhost

As our API is now ready we can connect backend to frontend and access data from the frontend

```

1  import React, {Component, Fragment} from 'react';
2  import Header from './Header';
3  import RestaurantsDisplay from './Restaurants';
4
5  const url = "http://35.154.220.142:8443/restaurant";
6
7  class Home extends Component{
8      constructor(){
9          super()
10     }
11     this.state={
12         restaurant:''
13     }
14
15     componentDidMount(){
16         fetch(url,{
17             method:'GET'
18         })
19         .then((res) => res.json())
20         .then((data) => {
21             this.setState({
22                 restaurant:data
23             })
24         })
25     })
26 }
27
28 render(){
29     return(
30         <Fragment>
31             <Header/>
32             <RestaurantsDisplay datalist={this.state.restaurant}/>
33         </Fragment>
34     )
35 }

```

Figure 4.56: Change the const url in the code

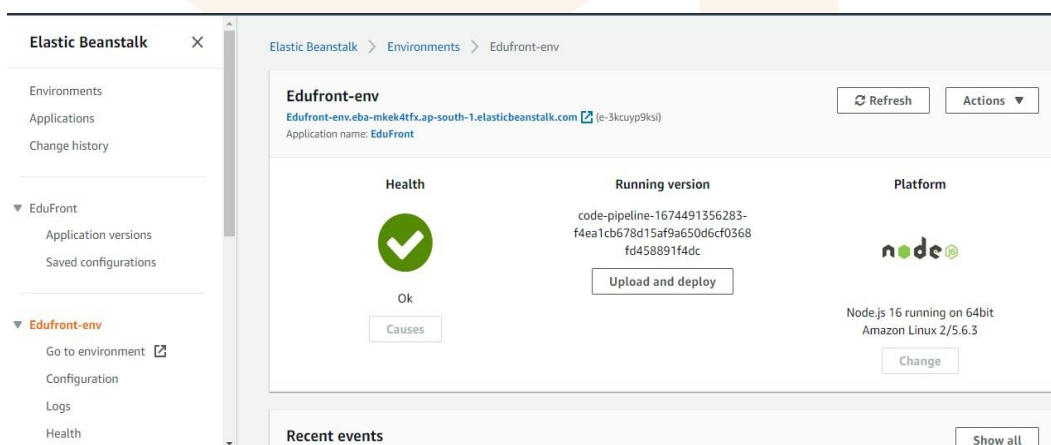


Figure 4.57: Click on the Edufront-env link



We can see the final output of the application on the screen.

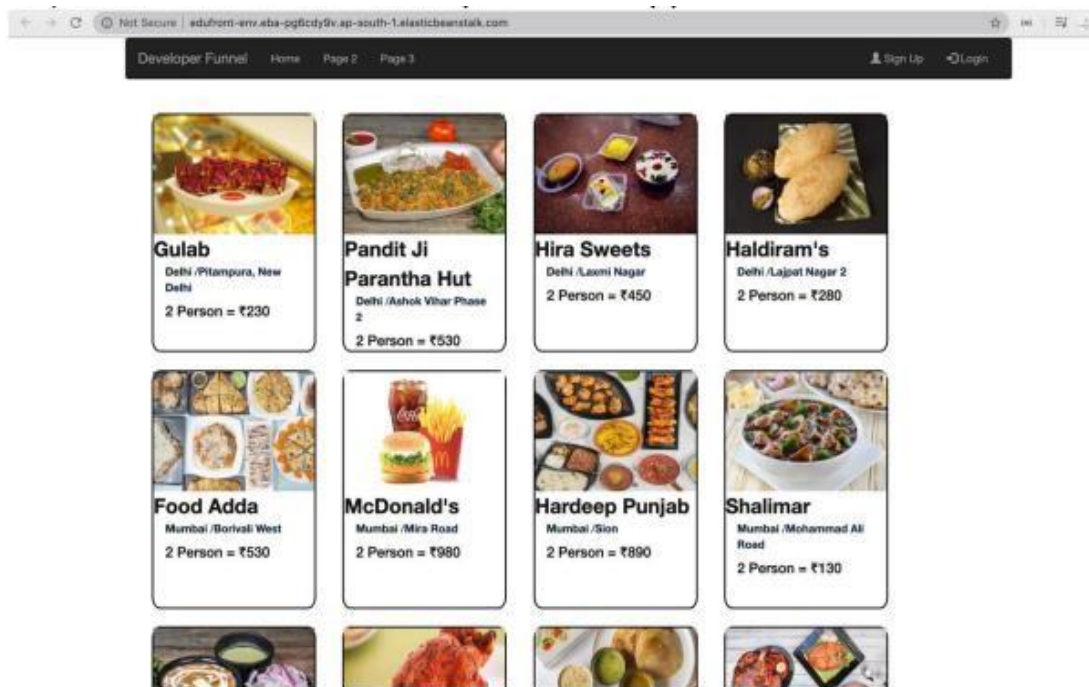


Figure 4.58: Final Output