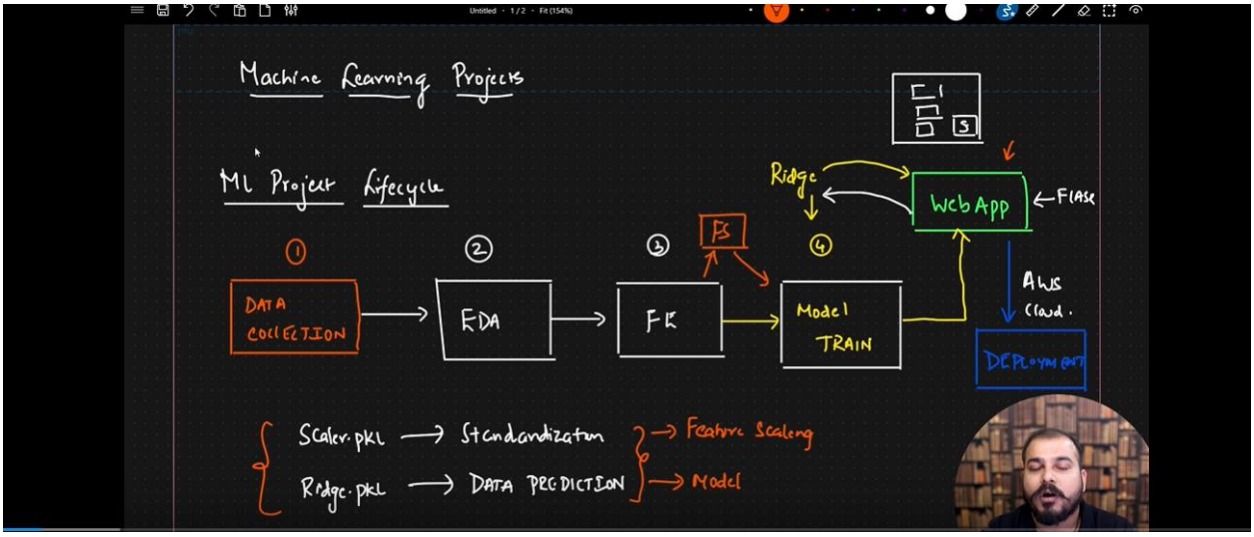
**List of steps involved to deploy the ML project:**

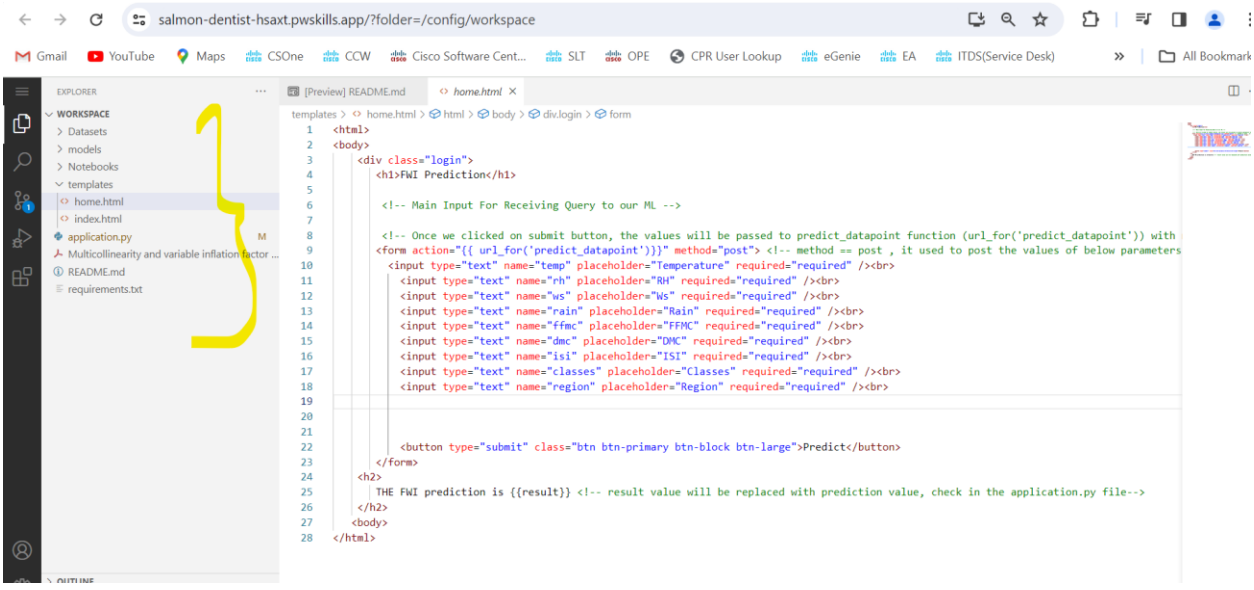



Till now we have completed the data collection, EDA, FE and model training. Now we need to develop the flask web application to deploy in the cloud environment.

<You can find the code in the below Git hub repository>

https://github.com/karrinethaji/ML_Intermediate_16_Linear-regression-endtoend-project-pwskills

Go to VS Code and develop the below code:

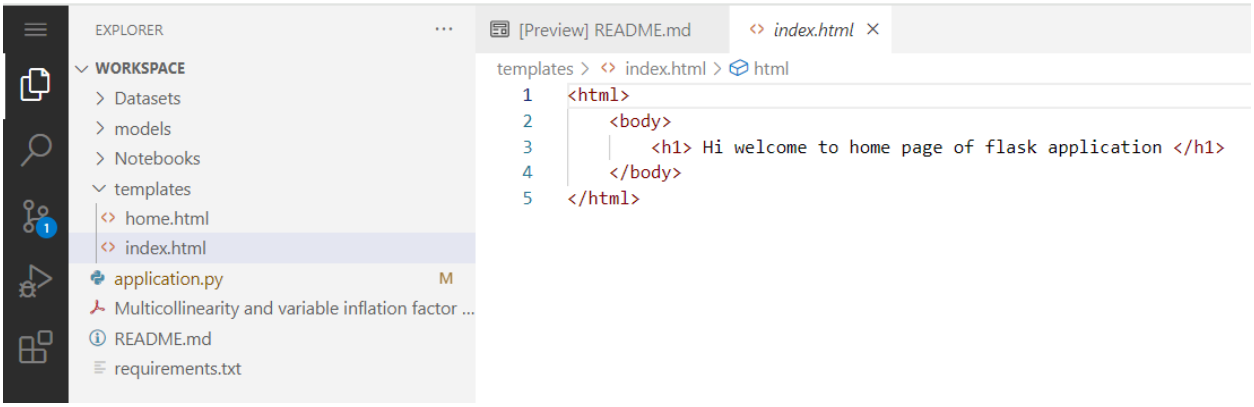



Application.py → flask application

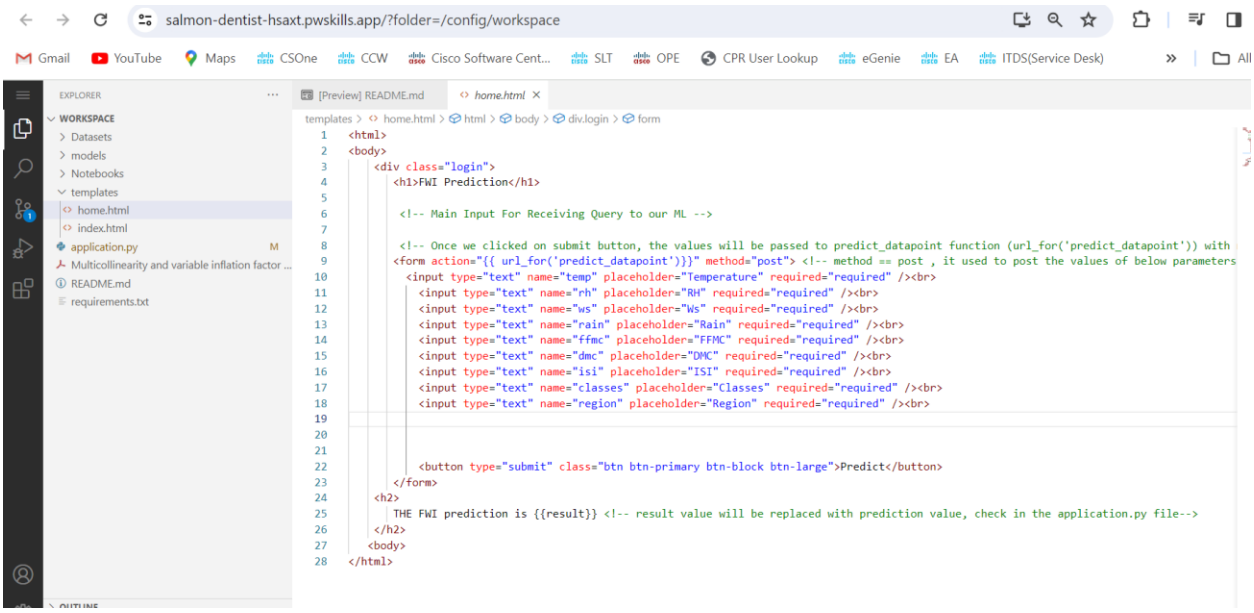
Go through the code and understand the things.

**Templates:**

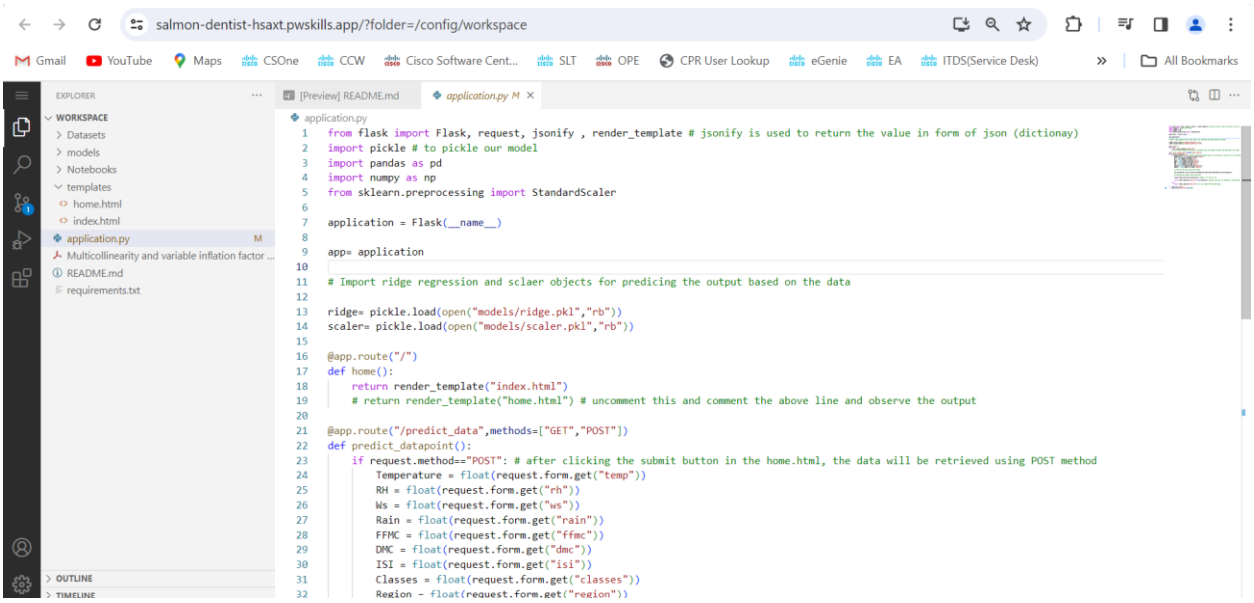
**Index.html:**

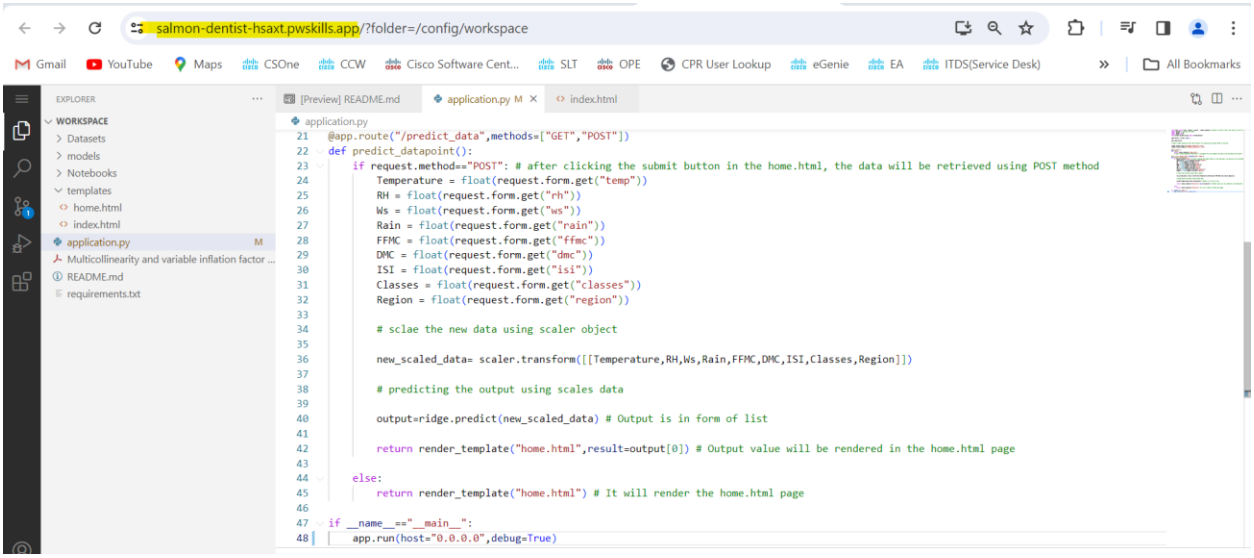



**Home.html:**




**Application.py: → flask application**

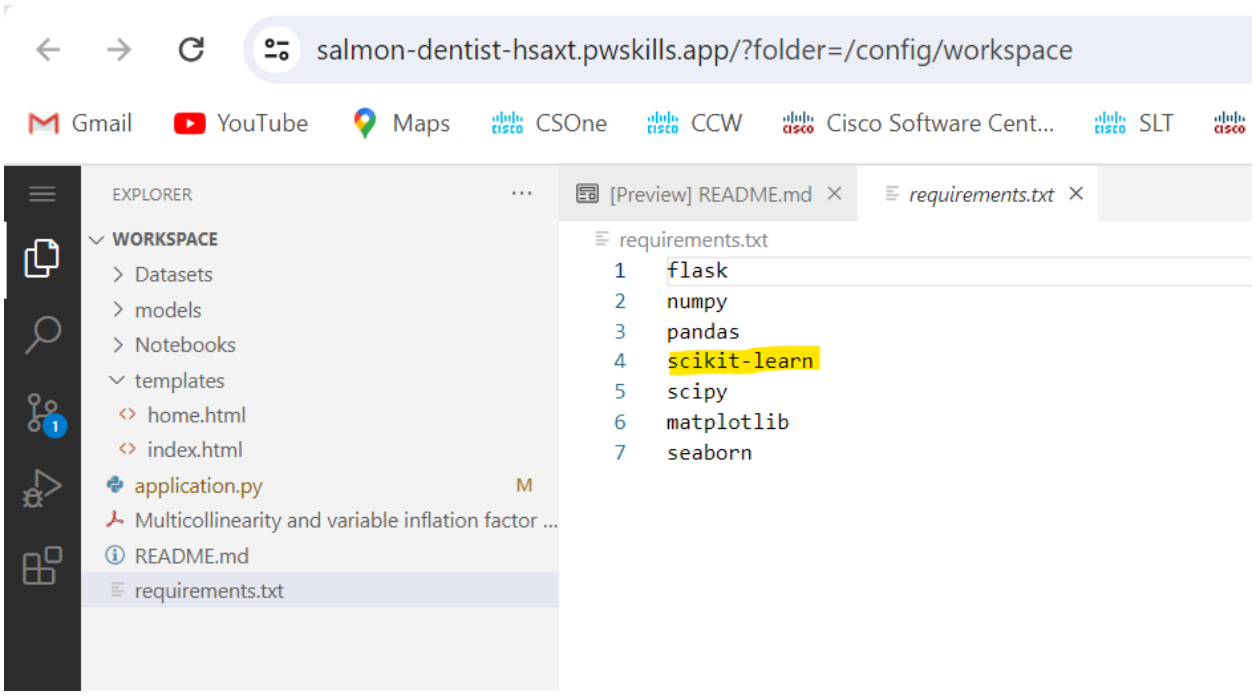


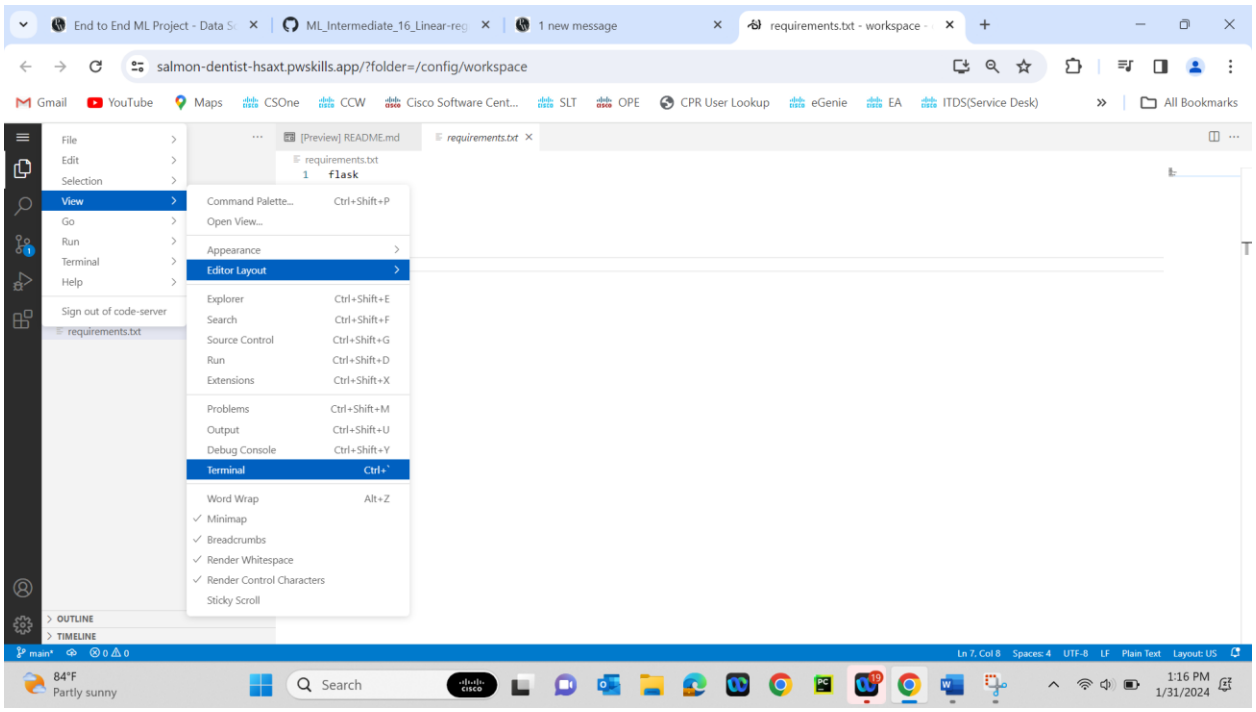

**** port=5000 , we need to specify the same port number in URL # 5000 is the default URL**

**Flow of execution of flask application:**

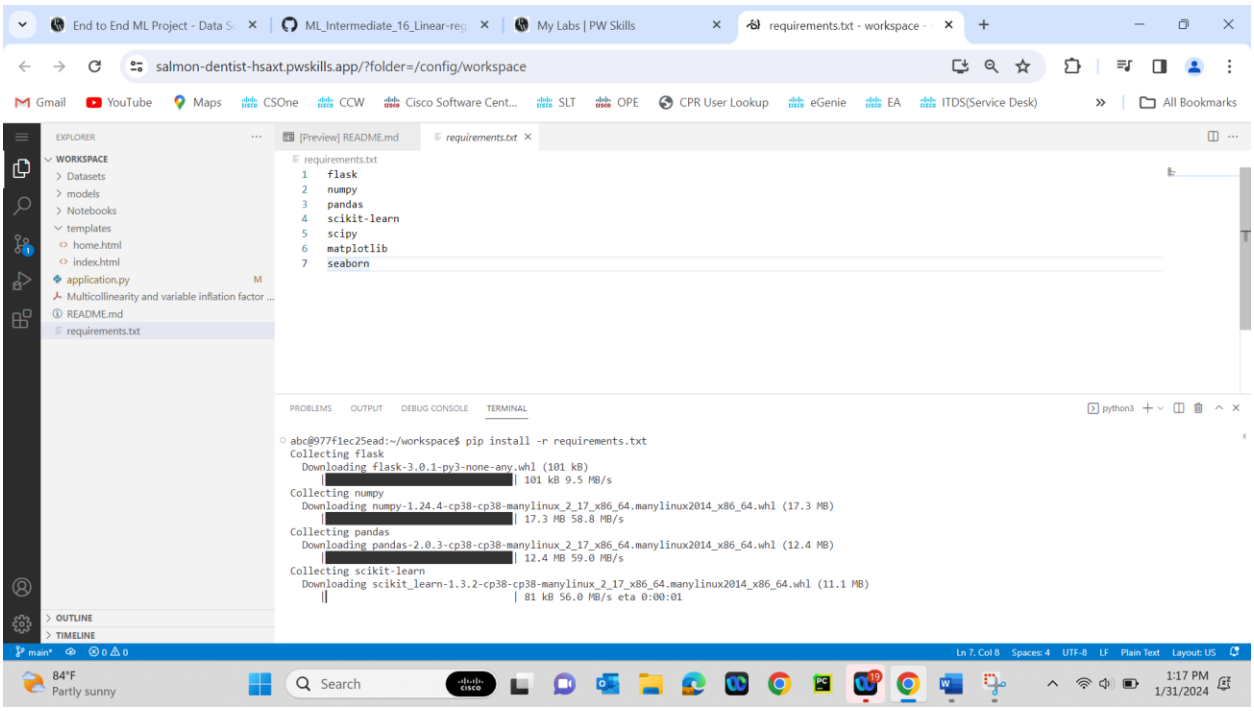
- ➔ Imported the necessary packages by using requirements.txt file.
- ➔ We can import multiple packages using the command pip install -r requirements.txt




➔ Enable the terminal




➔ In VSCode terminal we can run the command “pip install -r requirements.txt” to install the packages which are showing in the above picture.




➔ In the flask application, we need to initialize the WSGI server ( application = Flask(__name__))

➔ Import the pickle files of the ridge regression model and standardscaler object

➔ We have written below code for the default URL, Once we accessed the default URL, the index.html template will execute.

```
@app.route("/")
def home():
    return render_template("index.html")
    # return render_template("home.html") # uncomment this and comment the above
line and observe the output
```

➔ We have written below code for the predict_data URL, Once we accessed the predict_data URL, the below code will execute.

```
@app.route("/predict_data",methods=["GET","POST"])
def predict_datapoint():
    if request.method=="POST": # after clicking the submit button in the
home.html, the data will be retrieved using POST method
        Temperature = float(request.form.get("temp"))
        RH = float(request.form.get("rh"))
        Ws = float(request.form.get("ws"))
        Rain = float(request.form.get("rain"))
        FFMC = float(request.form.get("ffmc"))
        DMC = float(request.form.get("dmc"))
```

```
        ISI = float(request.form.get("isi"))
        Classes = float(request.form.get("classes"))
        Region = float(request.form.get("region"))

        # sclae the new data using scaler object

        new_scaled_data=
scaler.transform([[Temperature,RH,Ws,Rain,FFMC,DMC,ISI,Classes,Region]])

        # predicting the output using scales data

        output=ridge.predict(new_scaled_data) # Output is in form of list

        return render_template("home.html",result=output[0]) # Output value will
be rendered in the home.html page

    else:
        return render_template("home.html") # It will render the home.html page
```

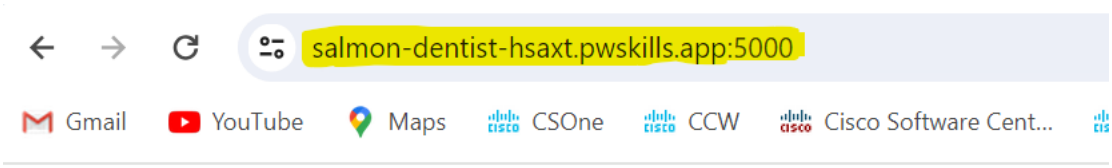
home.html content:

```
<!-- Main Input For Receiving Query to our ML -->

<!-- Once we clicked on submit button, the values will be passed to predict_datapoint function (url
<form action="{{ url_for('predict_datapoint')}}" method="post"> <!-- method == post , it used to pos
  <input type="text" name="temp" placeholder="Temperature" required="required" /><br>
    <input type="text" name="rh" placeholder="RH" required="required" /><br>
    <input type="text" name="ws" placeholder="Ws" required="required" /><br>
    <input type="text" name="rain" placeholder="Rain" required="required" /><br>
    <input type="text" name="ffmc" placeholder="FFMC" required="required" /><br>
    <input type="text" name="dmc" placeholder="DMC" required="required" /><br>
    <input type="text" name="isi" placeholder="ISI" required="required" /><br>
    <input type="text" name="classes" placeholder="Classes" required="required" /><br>
    <input type="text" name="region" placeholder="Region" required="required" /><br>
```

➔ In the home.html content, we have provided the method=”POST”.

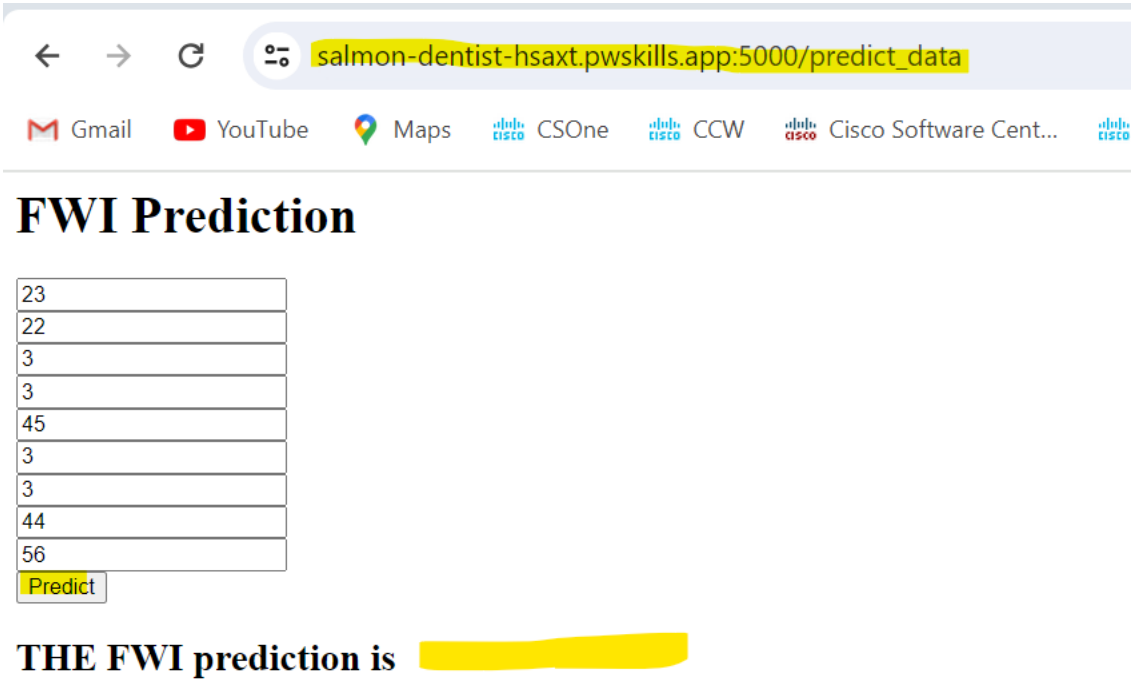
Once we access the predict_data URL , It will move to else block initially (because initially the method won’t be POST) and will execute the home.html with form (asking for the values to enter).

https://salmon-dentist-hsaxt.pwskills.app:5000/ → It is default URL, It will show the index.html page




# Hi welcome to home page of flask application

https://salmon-dentist-hsaxt.pwskills.app:5000/predict_data --> It is predict_data URL, It will execute the html page which is showing in else block (home.html)




Once we entered the values of all the parameters, we need to click on Predict.

Once we click on submit button, as we specified the **method="post"** in the form (home.html), the if block will execute.

In If block, we retrieved the values which we entered in the form using the below code for each parameter:

```
Variable_name = float(request.form.get("name_of_parameter_in_form"))
```

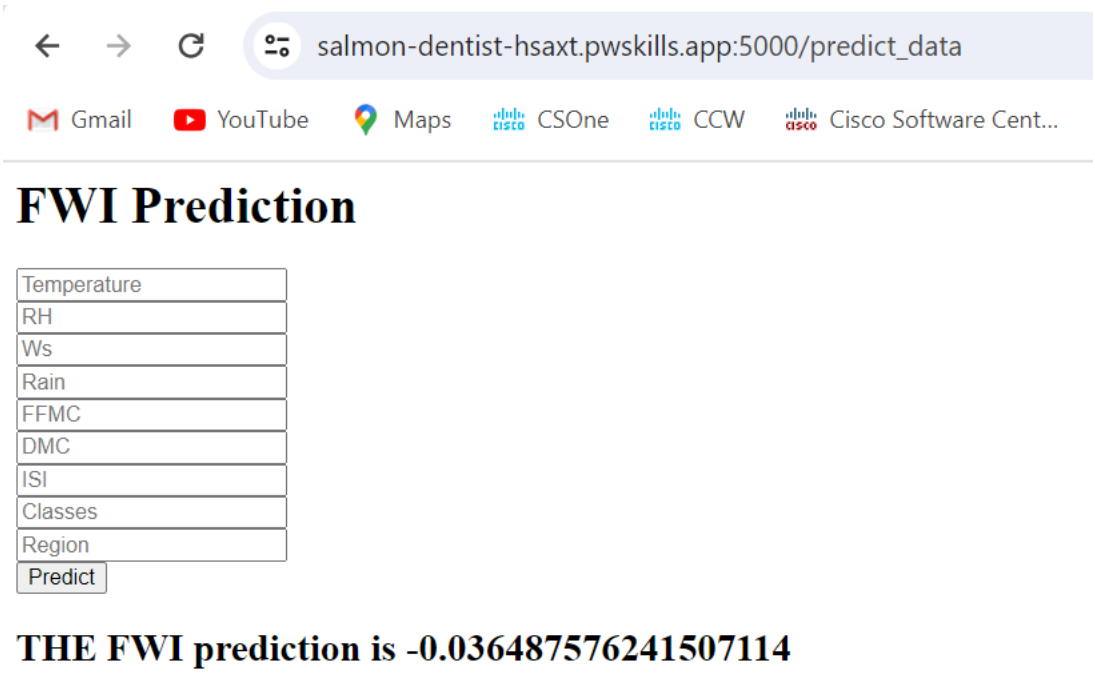
- ➔ We scaled the data using the standard scaler pickle object and we predicted the output using the ridge pickle object

➔ Then we rendered the result in the home.html page.

```
return render_template("home.html",result=output[0])
```

➔ The above code will render the output in home.html page at {{result}}.

```
            <button type="submit" class="btn btn-primary btn-block btn-
large">Predict</button>
        </form>
    <h2>
        THE FWI prediction is {{result}} <!-- result value will be replaced with
prediction value, check in the application.py file-->
```




This is the explanation for flask code

**Let's see how we can push the VSCode to github: (already covered in the earlier projects like webscrapping refer once)**

➔ **Create a public repository in git hub and use the below code to push the code into github**

**If you want to commit all the files from the VSCode editor then use 2nd command as "git add ."**

**Provide your github URL in the 5th step I,e git remote add origin <git hub url>**

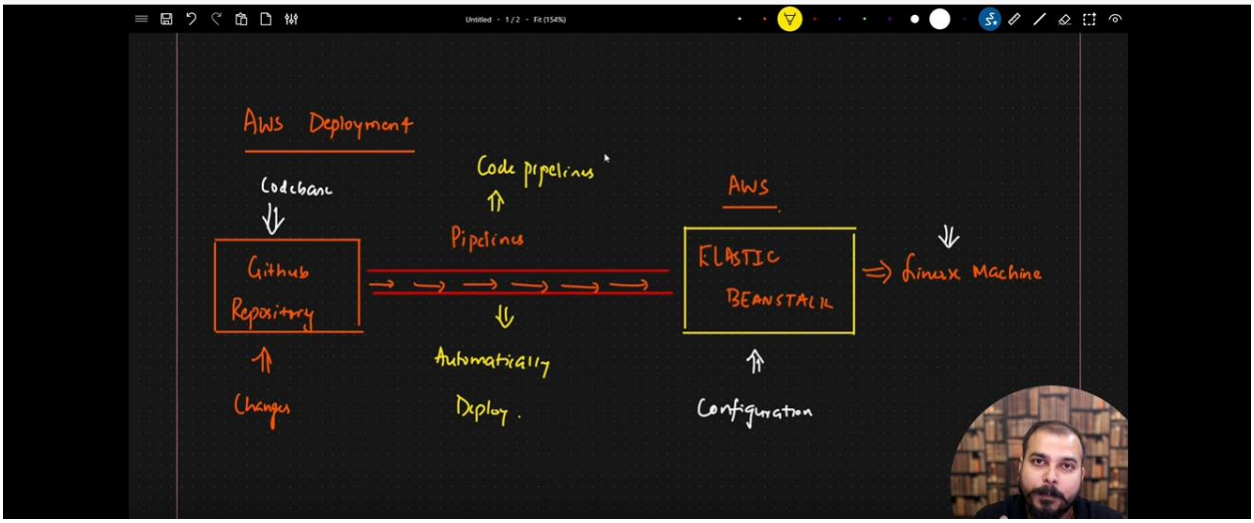
**NOTE:** If the code is not pushed into github after using the command "git push **-u** origin main" then use "git push **-f** origin main" instead.

**I have deployed the code into the below repository:**

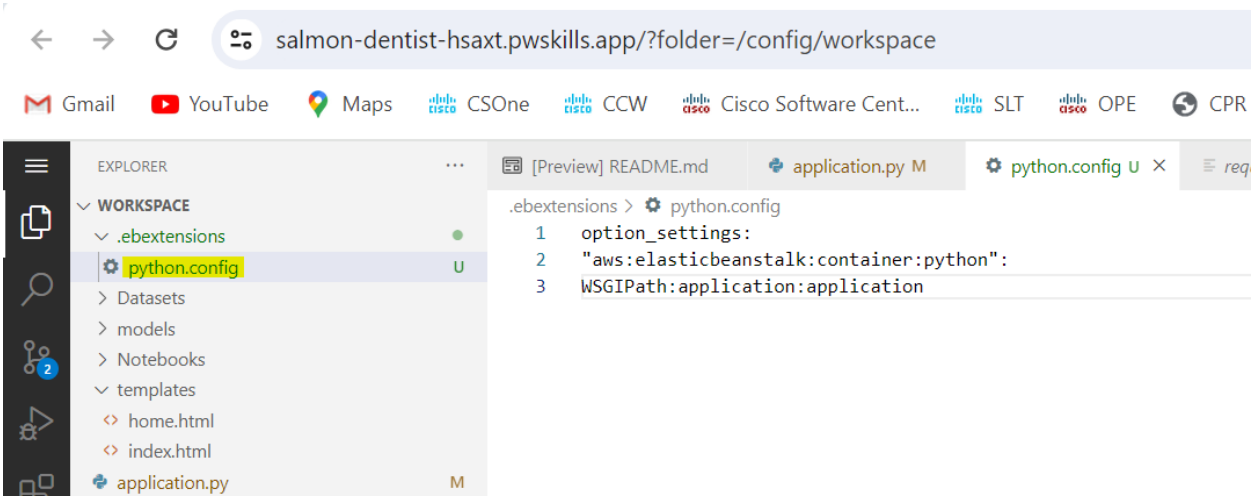
**https://github.com/karrinethaji/ML_Intermediate_16_Linear-regression-endtoend-project-pwskills/tree/main**

**Deploying the code in AWS environment:**

- ➔ We will use the “Elastic beanstalk” service to deploy our code
- ➔ Elastic beanstalk is a linux based machine
- ➔ We need to configure things (need to add python.config file to our repository as the elastic beanstalk is linux based machine) in the “elastic beanstalk” to deploy the code (we use codepipelines to get the code from github)
- ➔ Pipelines are the medium to push the code from github to elastic beanstalk




We need to make some changes (adding a **.ebextensions** folder and python.config file to our existing repository before deploying the code in AWS)




**Content of **python.config** file

option_settings:

"aws:elasticbeanstalk:container:python":

WSGIPath: application:application

WSGIPath:<flask-application-name>:<app_name_in_code>

In our example, the flask application name was application.py so we provided name as application.

<flask-application-name> → It is the flask application name I,e **application**.py

<app_name_in_code> → In the flask application, we written **application**= Flask(__name__) so, here app_name_in_code will be replaced with **application**

**Process to add the files (old+new) to the existing repository from VSCode:** (Once we committed the files, old files will remain available in the repository and new files will be appended)

**Commands used to recommit old+new files:**

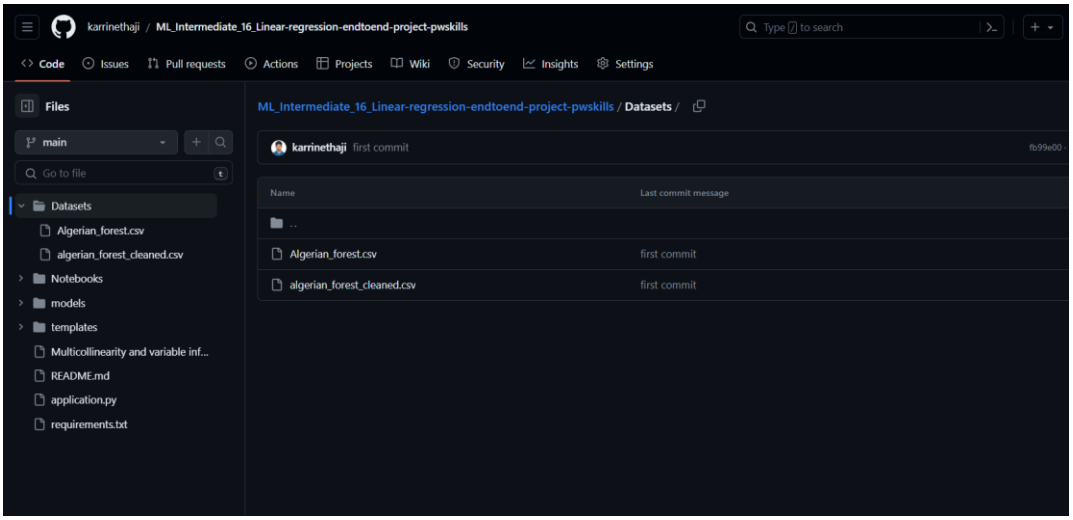
- ➔ **git init**
- ➔ **git add .**
- ➔ **git commit -m "<any message>" I,e git commit -m "elastic beanstalk configuration"**
  **If required use the commands to enter username and email address of github**
- ➔ **git push -u origin main** or git push -f origin main

**Commands used:**

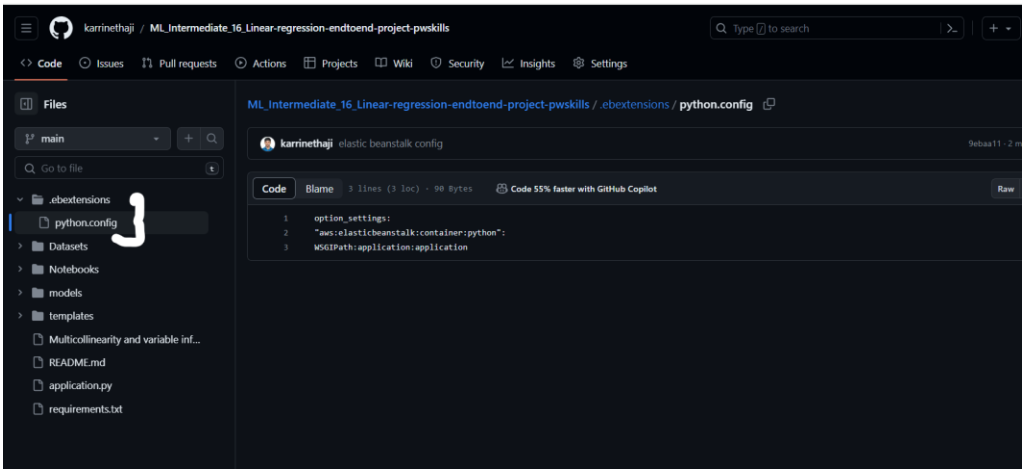
```
abc@977f1ec25ead:~/workspace$ git init
Reinitialized existing Git repository in /config/workspace/.git/
abc@977f1ec25ead:~/workspace$ git add .
abc@977f1ec25ead:~/workspace$ git commit -m "elastic beanstalk config"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'abc@977f1ec25ead.(none)')
abc@977f1ec25ead:~/workspace$ git config --global user.email "karrinethaji001@gmail.com"
abc@977f1ec25ead:~/workspace$ git config --global user.name "karrinethaji"
abc@977f1ec25ead:~/workspace$ git commit -m "elastic beanstalk config"
[main 9ebaa11] elastic beanstalk config
 2 files changed, 4 insertions(+), 1 deletion(-)
 create mode 100644 .ebextensions/python.config
```

```
abc@977f1ec25ead:~/workspace$ git push -u origin main
To https://github.com/karrinethaji/ML_Intermediate_16_Linear-regression-endtoend-project-fromgithub.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/karrinethaji/ML_Intermediate_16_Linear-regression-endtoend-project-fromgithub.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
abc@977f1ec25ead:~/workspace$ git push -f origin main
Enumerating objects: 52, done.
Counting objects: 100% (52/52), done.
Delta compression using up to 64 threads
Compressing objects: 100% (50/50), done.
Writing objects: 100% (52/52), 2.31 MiB | 4.68 MiB/s, done.
Total 52 (delta 14), reused 0 (delta 0)
remote: Resolving deltas: 100% (14/14), done.
remote: This repository moved. Please use the new location:
remote:   https://github.com/karrinethaji/ML_Intermediate_16_Linear-regression-endtoend-project-pwskills.git
To https://github.com/karrinethaji/ML_Intermediate_16_Linear-regression-endtoend-project-fromgithub.git
 + 1d788f6...9ebaa11 main -> main (forced update)
```

**Before pushing code:**

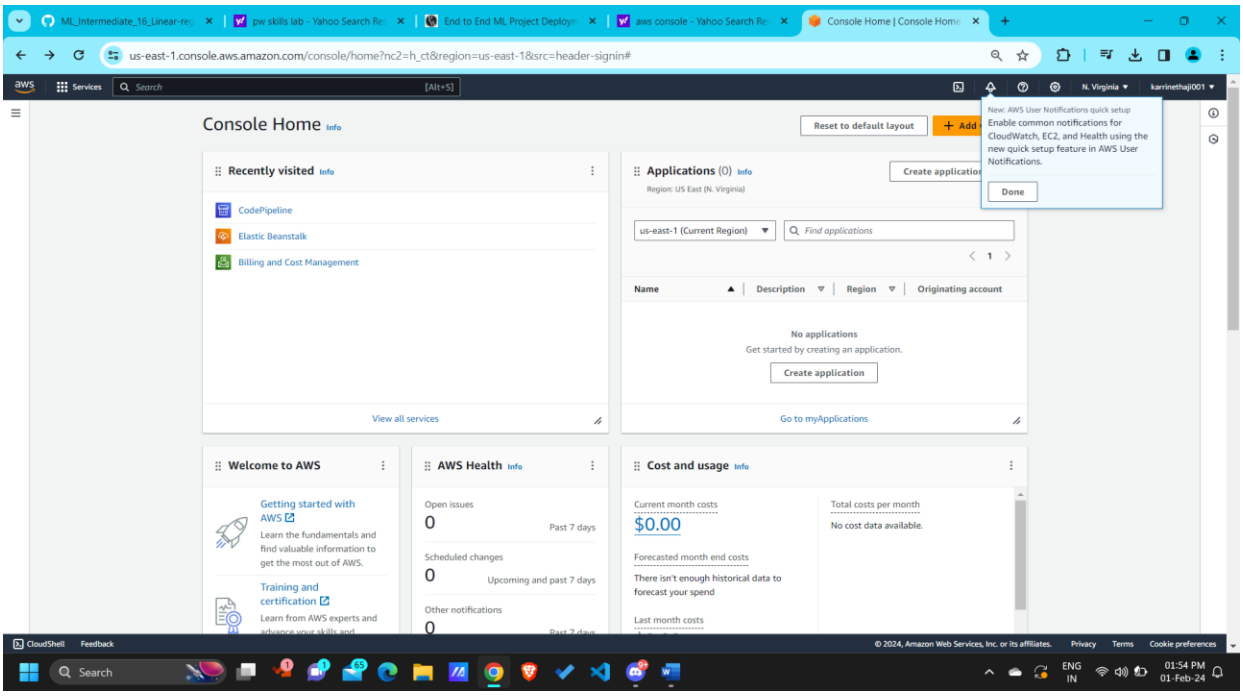



**After pushing code:**

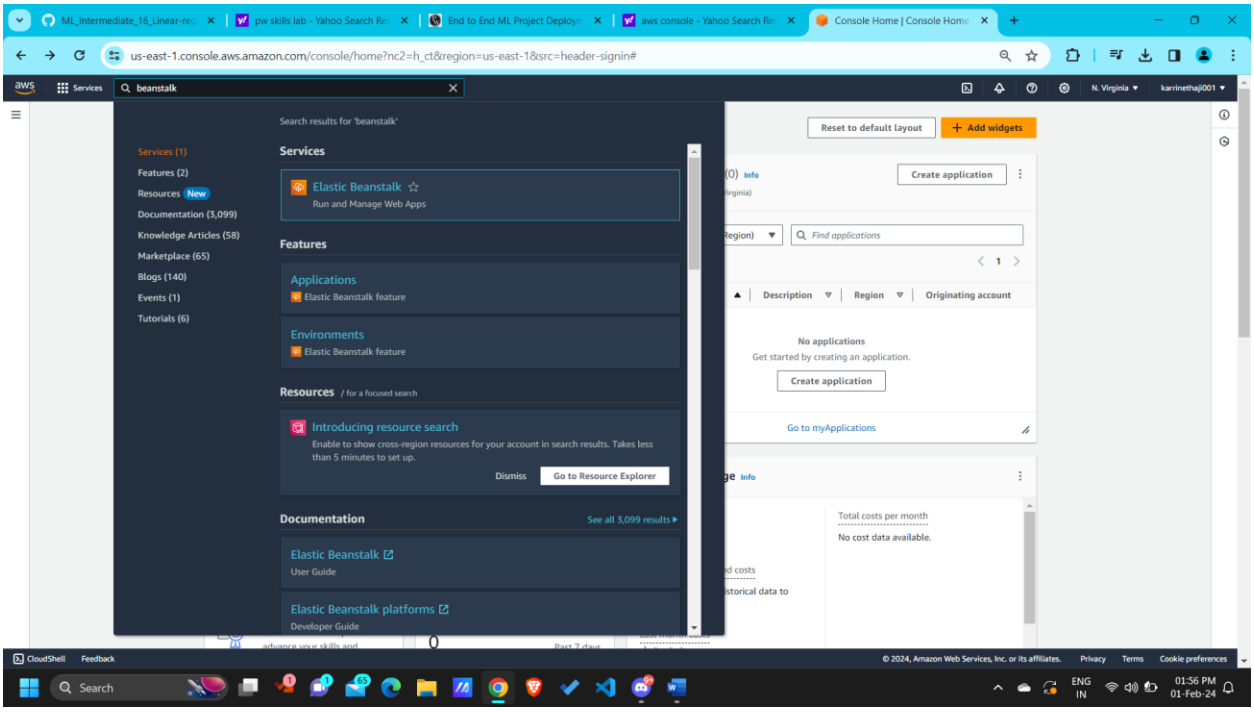



**Go to AWS Console:** https://us-east-1.console.aws.amazon.com/console/home?nc2=h_ct&region=us-east-1&src=header-signin#

**Log in email:** ka**001**@gmail.com



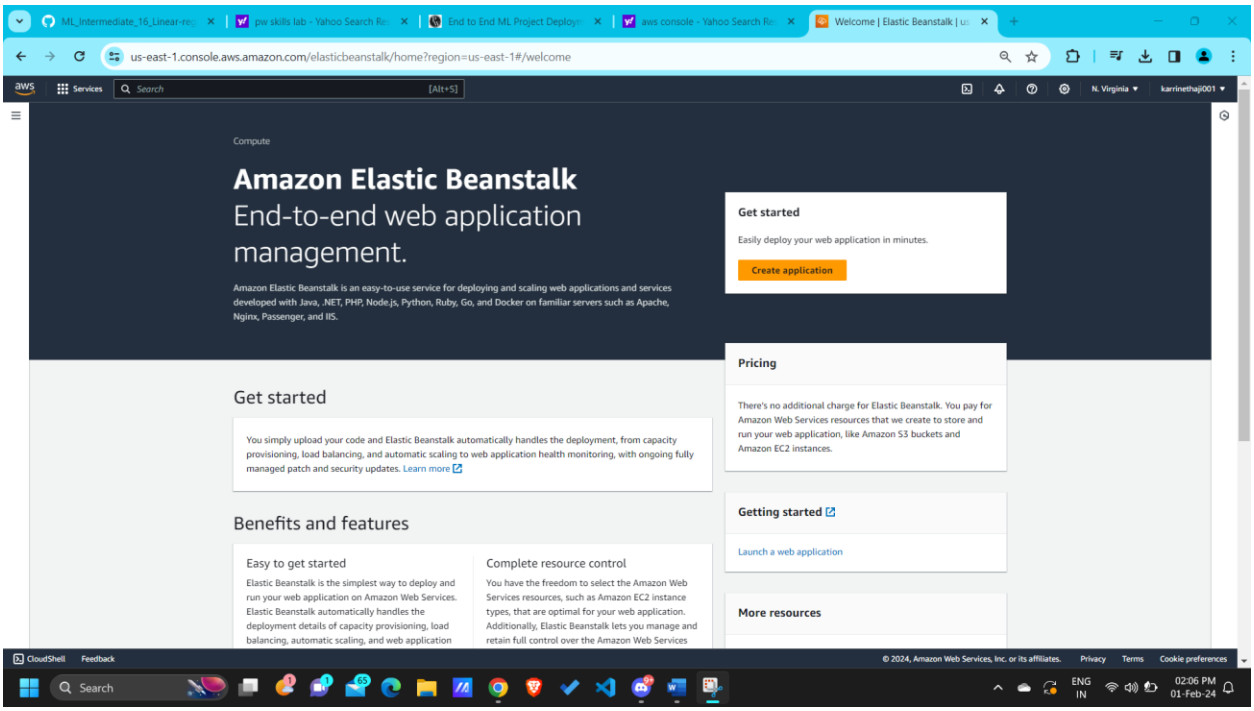

Search as elastic beanstalk and select the **Elastic Beanstalk** service



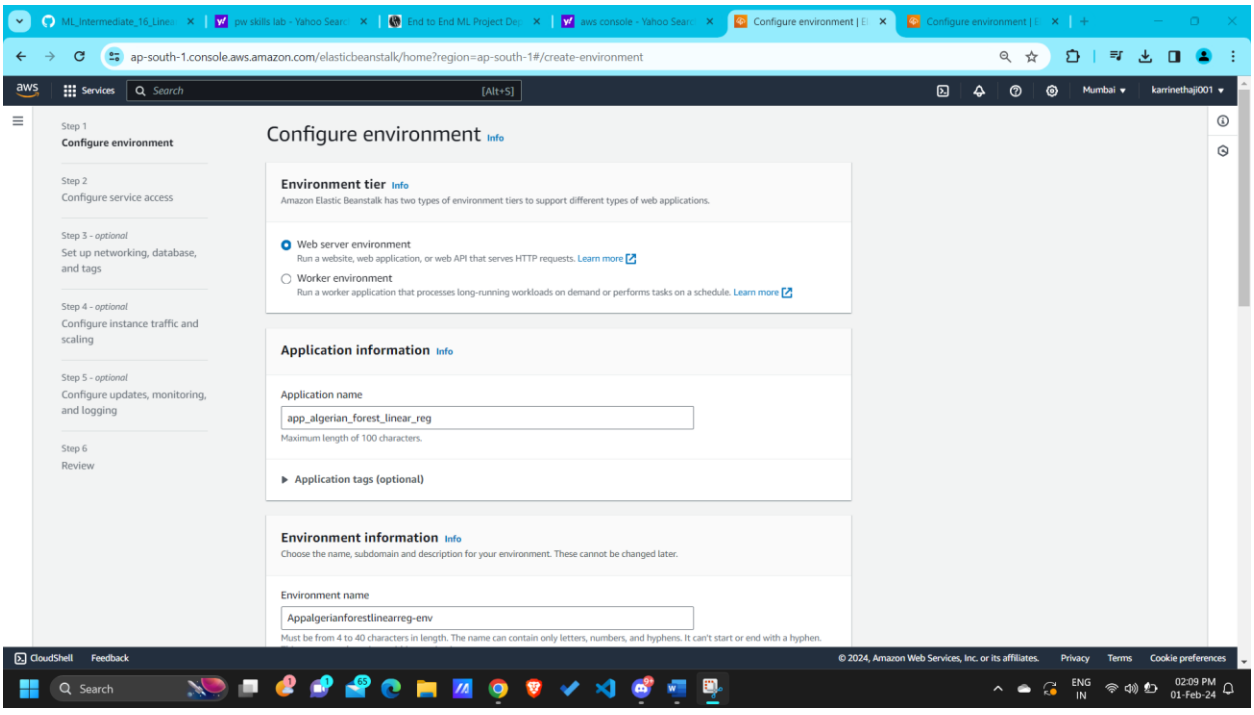

➔ Select the "**Elastic Beanstalk**" service and you will get a UI interface as below

**Documentation for elastic beanstalk: (**Important)**

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environments-create-wizard.html



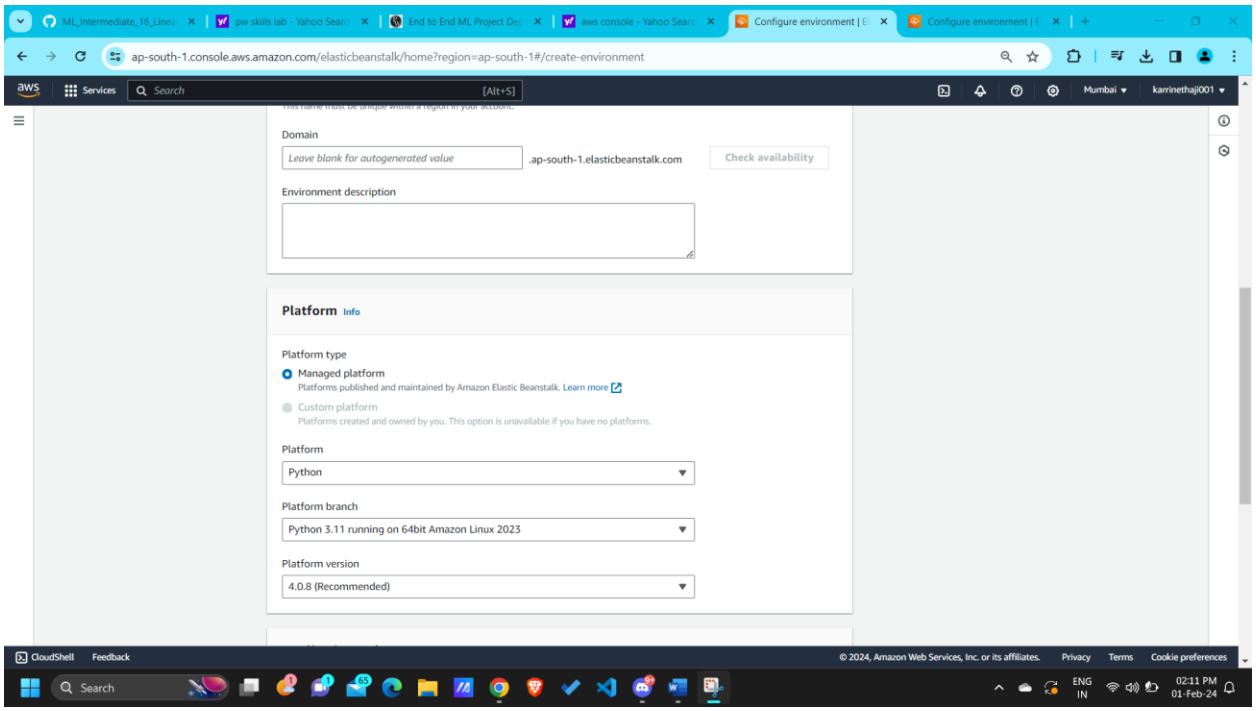

➔ Click on "**Create application**"

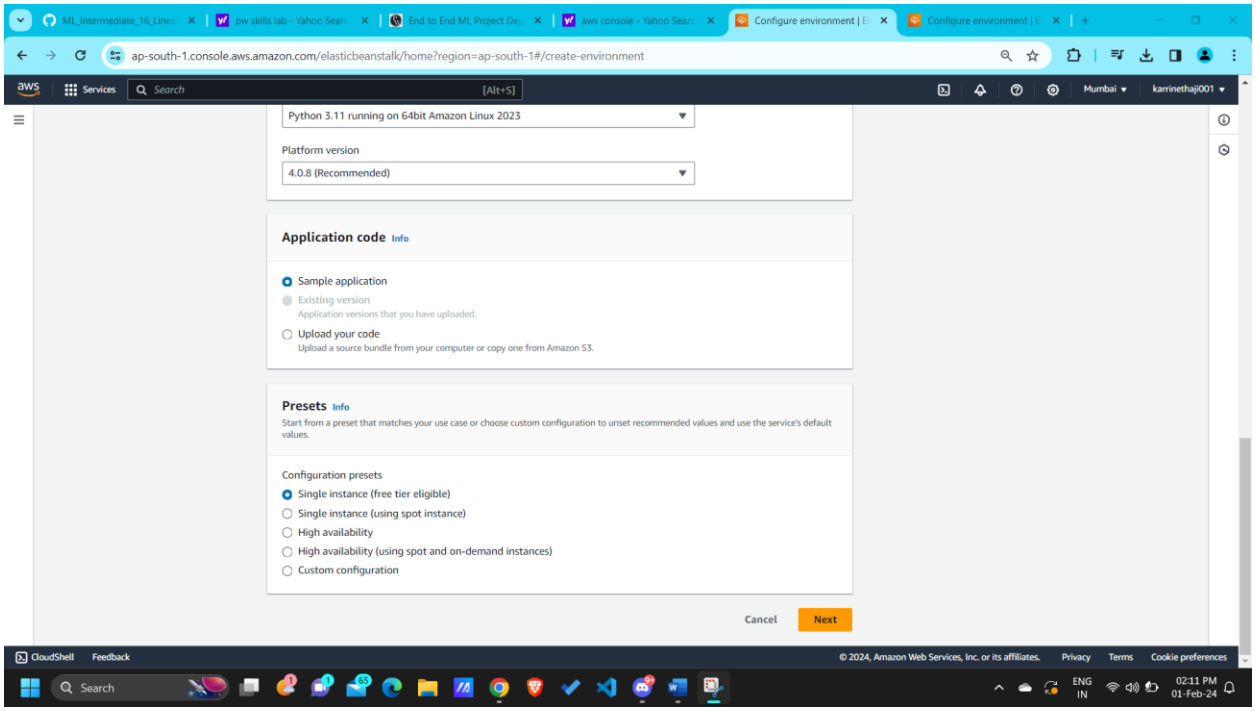



➔ Enter the application name and the environment name will be auto populated according to the application name.

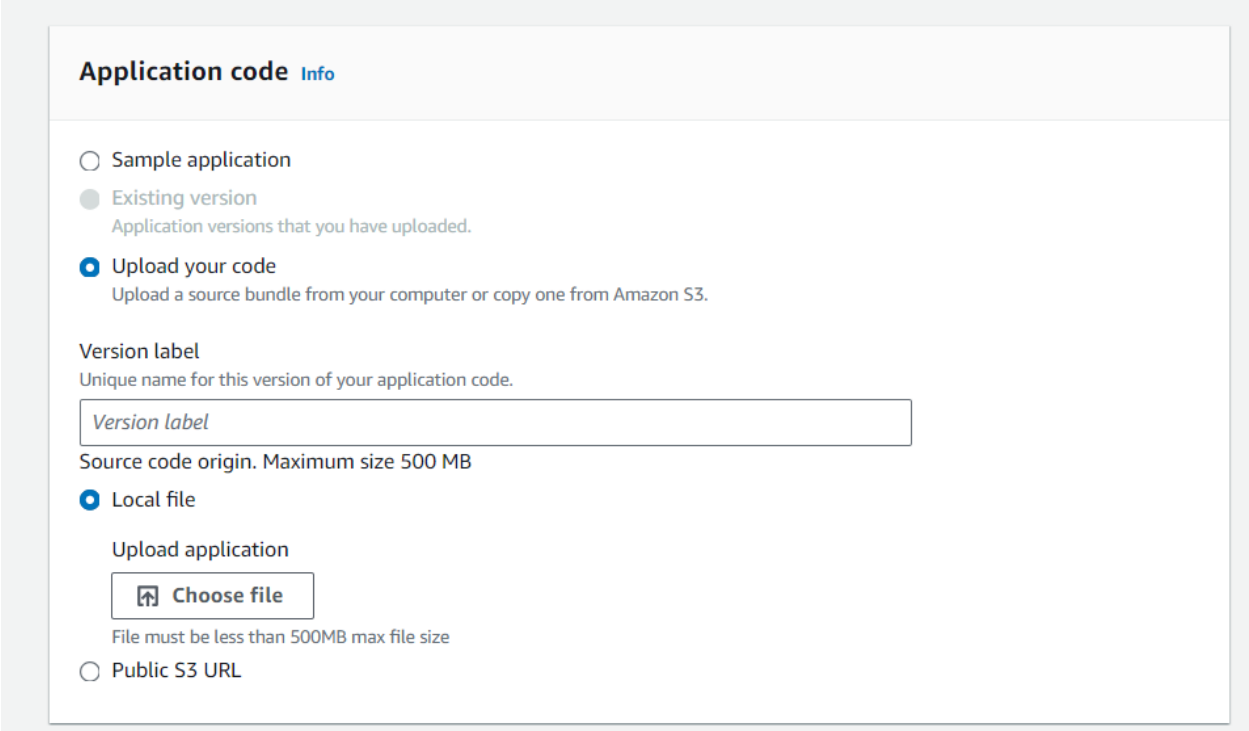
**Application name:** app_algerian_forest_linear_reg

**Environment name:** Appalgerianforestlinearreg-env

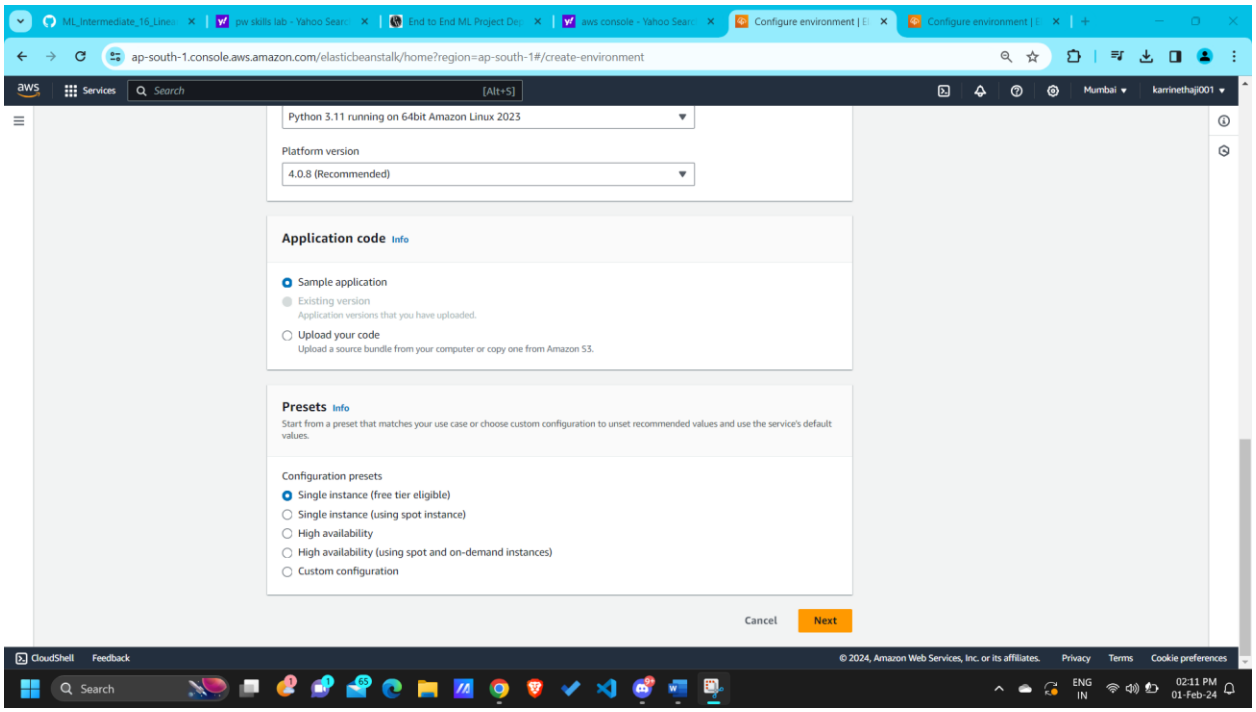


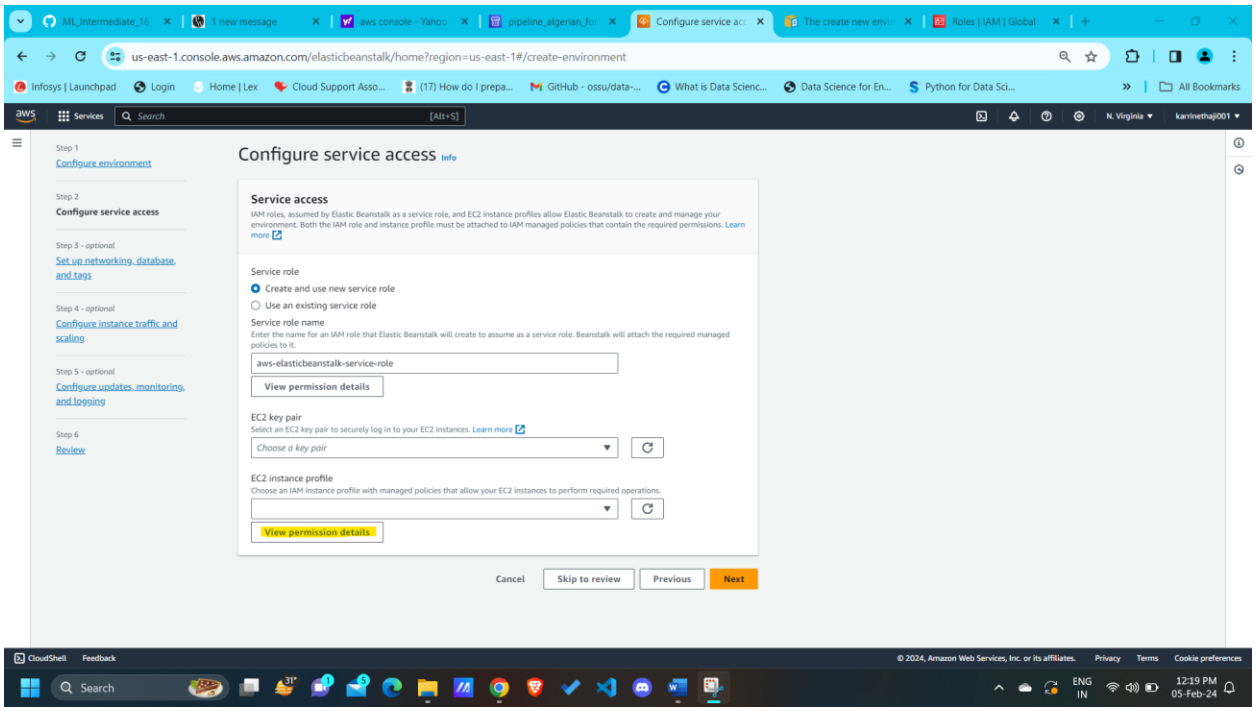

➔ In application code , we can select the upload your code option as well if you have zip file in your local PC




If we don’t have zip file then we can create the sample application and then later we can use code pipeline to push our code to elastic beanstalk.




➔ Click on “**Next**”



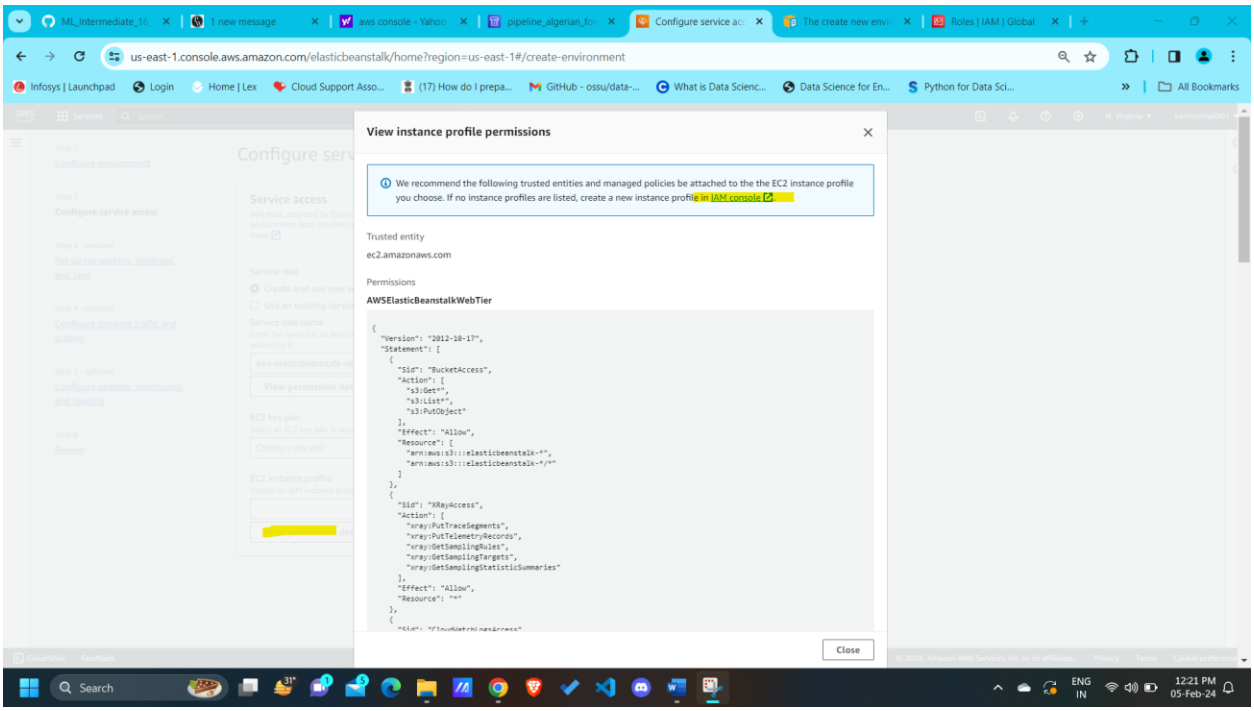

➔ If you are using the elastic beanstalk for the first time, then Create a **IAM role** using the below procedure:

**Create IAM Role for EC2 instance profile:**

**To create a an IAM Role for EC2 instance profile selection**

1. Choose **View permission details.** This displays under the EC2 instance profile dropdown list.

   A modal window titled View instance profile permissions displays. This window lists the managed profiles that you'll need to attach to the new EC2 instance profile that you create. It also provides a link to launch the IAM console.

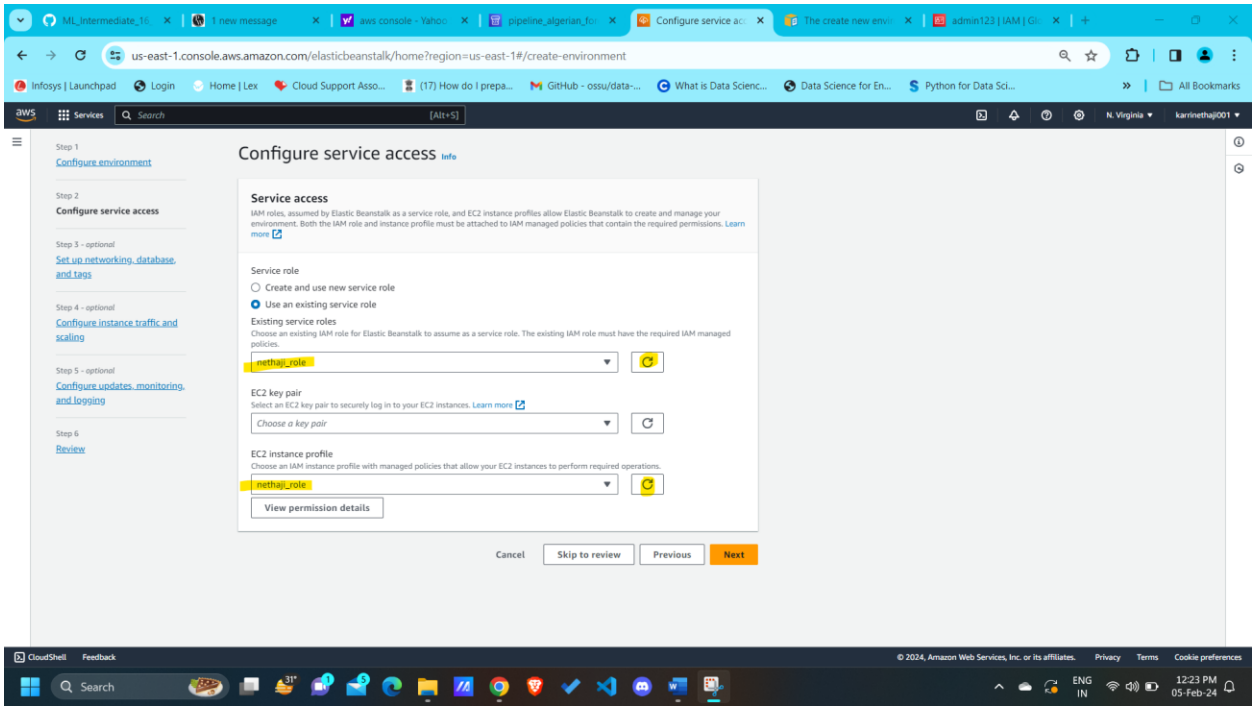



2. Choose the **IAM console** link displayed at the top of the window.
3. In the IAM console navigation pane, choose **Roles**.
4. Choose **Create role**.
5. Under Trusted entity type, choose **AWS service**.
6. Under **Use case**, choose **EC2**.
7. Choose **Next**.
8. Attach the appropriate managed policies. Scroll in the View instance profile permissions modal window to see the managed policies. The policies are also listed here:
    - `AWSElasticBeanstalkWebTier`
    - `AWSElasticBeanstalkWorkerTier`
    - `AWSElasticBeanstalkMulticontainerDocker`

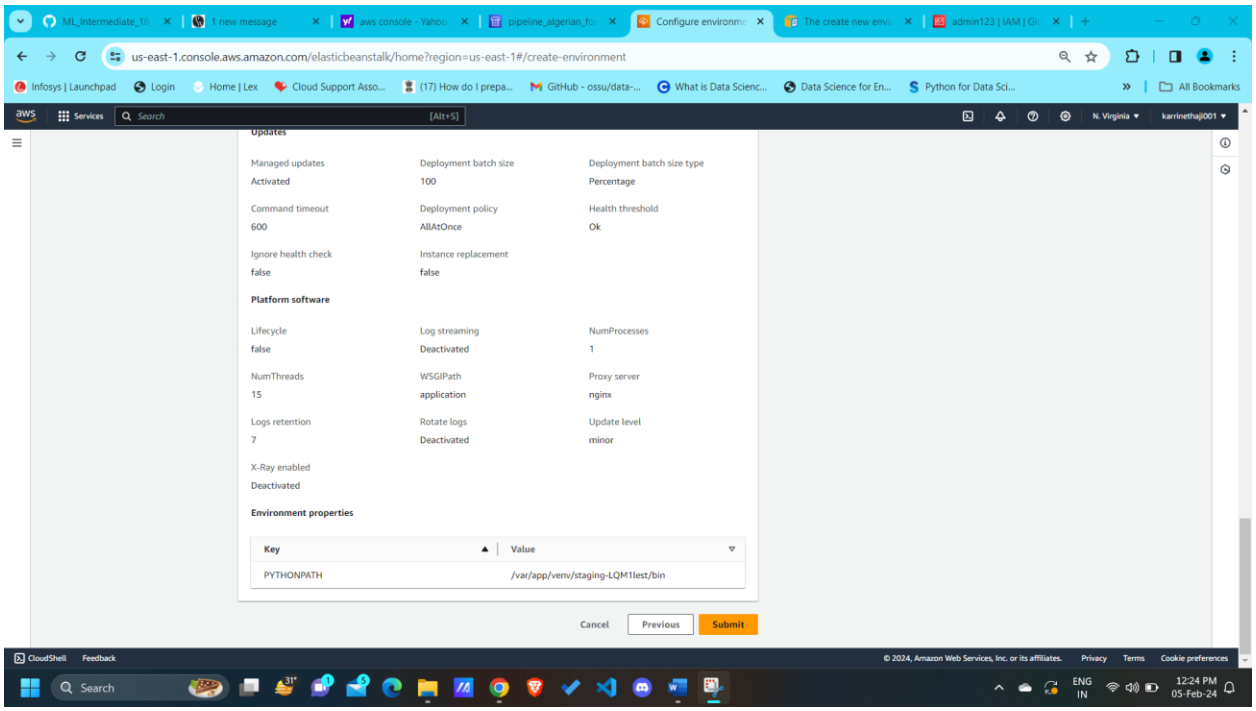
    `(or)`

    Select “AdministratorAccess” role instead of top 3 accesses.
9. Choose **Next**.
10. Enter a name for the role.
11. (Optional) Add tags to the role.
12. Choose **Create role**.
13. Return to the Elastic Beanstalk console window that is open.
14. Close the modal window View instance profile permissions.

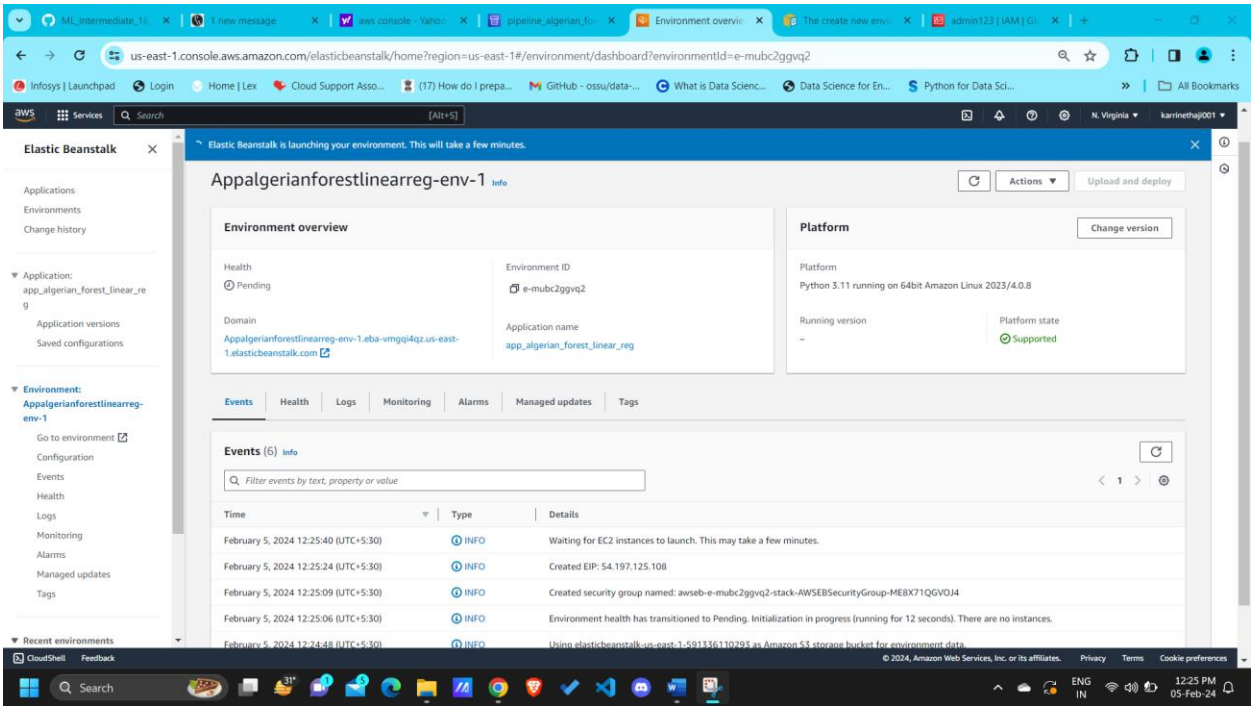
➔ I have created a role (**nethaji_role**) using the above procedure

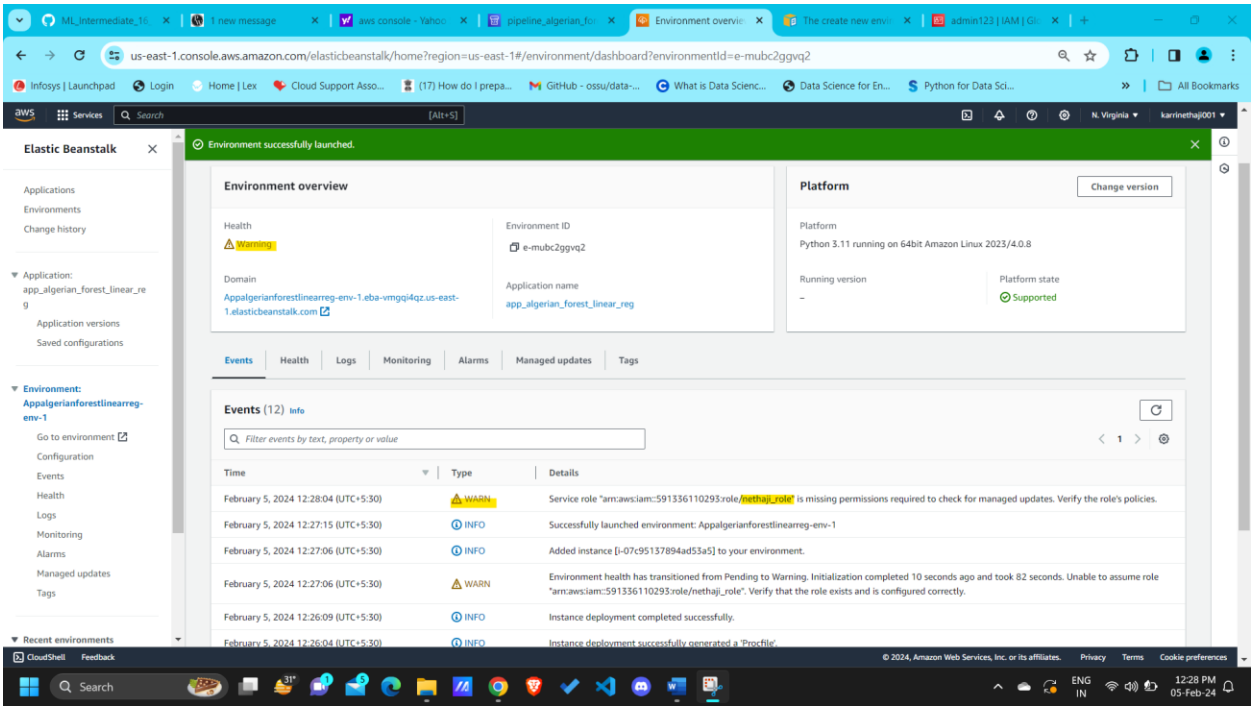



➔ Click on "**Skip to review**"



➔ Review the details and click on “**Submit**”, It will take some time to Create the environment. Wait for few minutes and create the codepipeline or create a duplicate tab and proceed to create a Codepipeline using the below procedure:

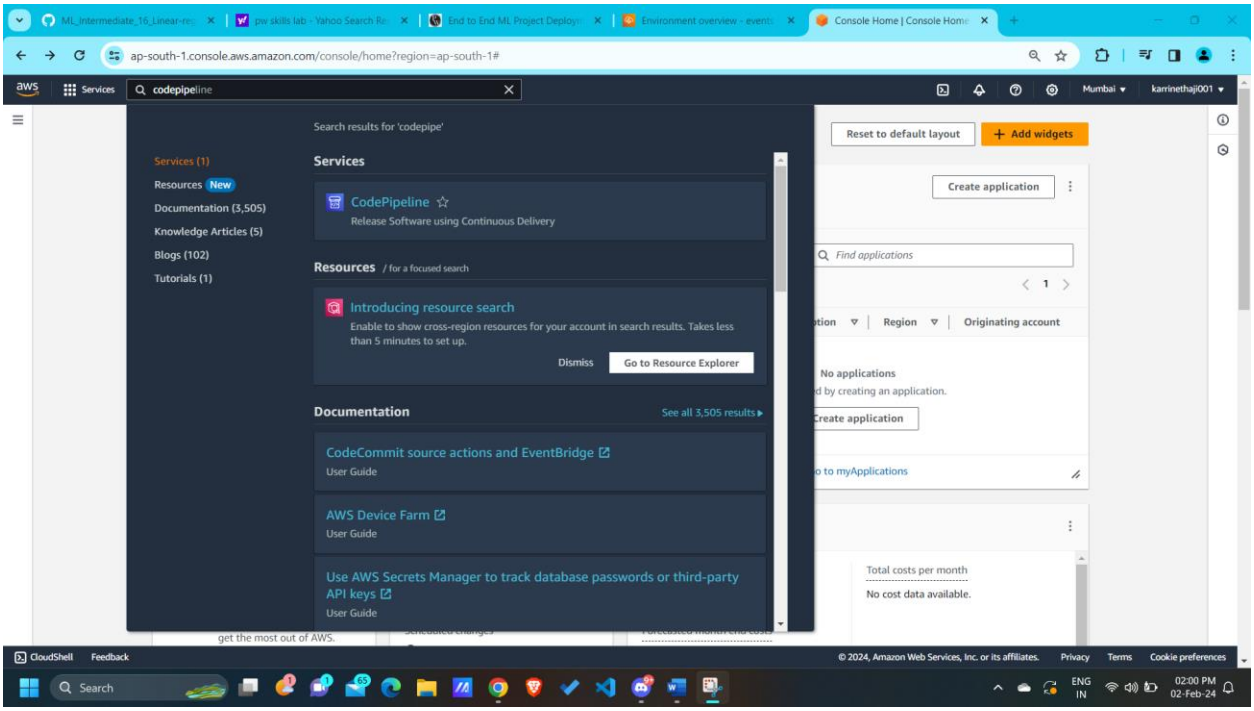


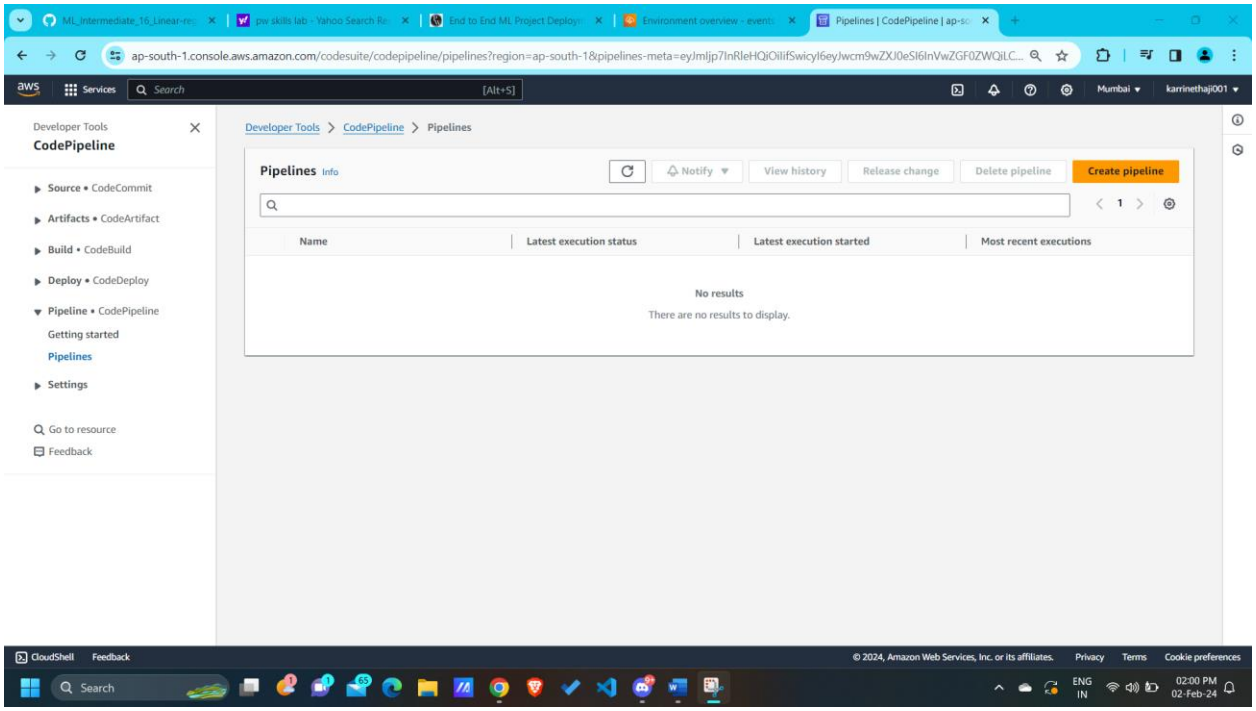

** we can ignore the warnings and proceed to go with creating the code pipeline.

**Let’s build the code pipeline using the below procedure:**

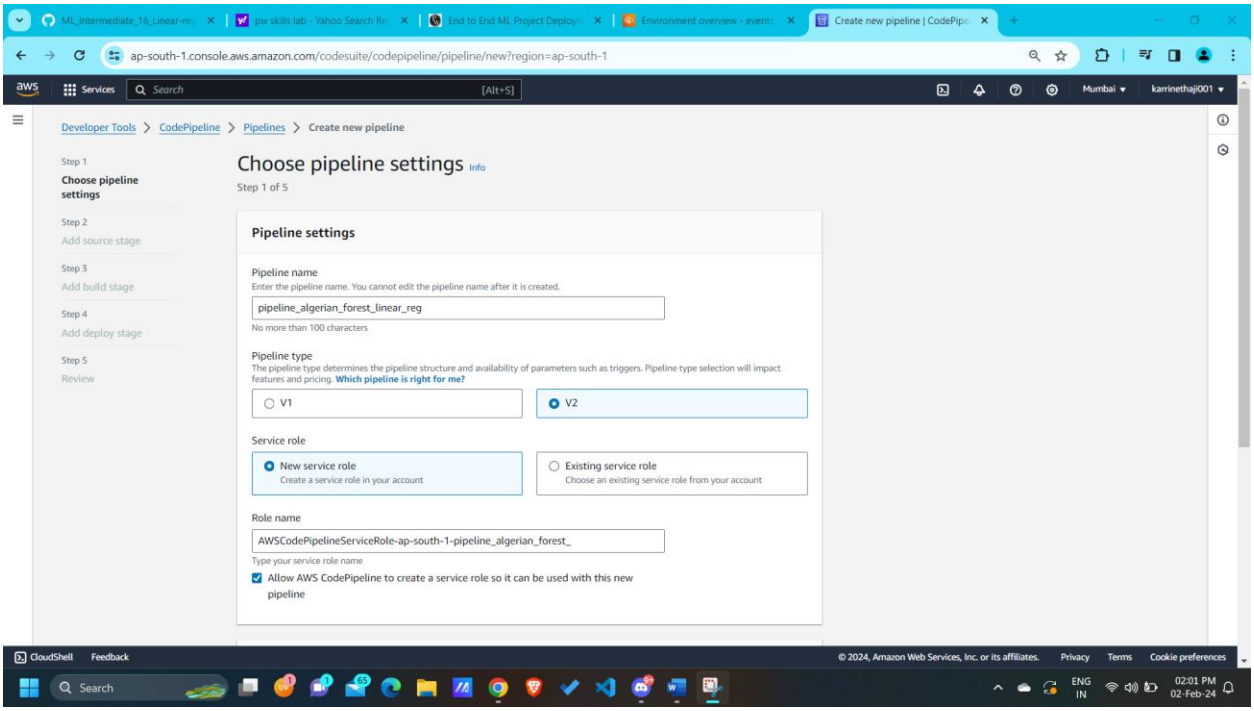
➔ Search as **CodePipeline**

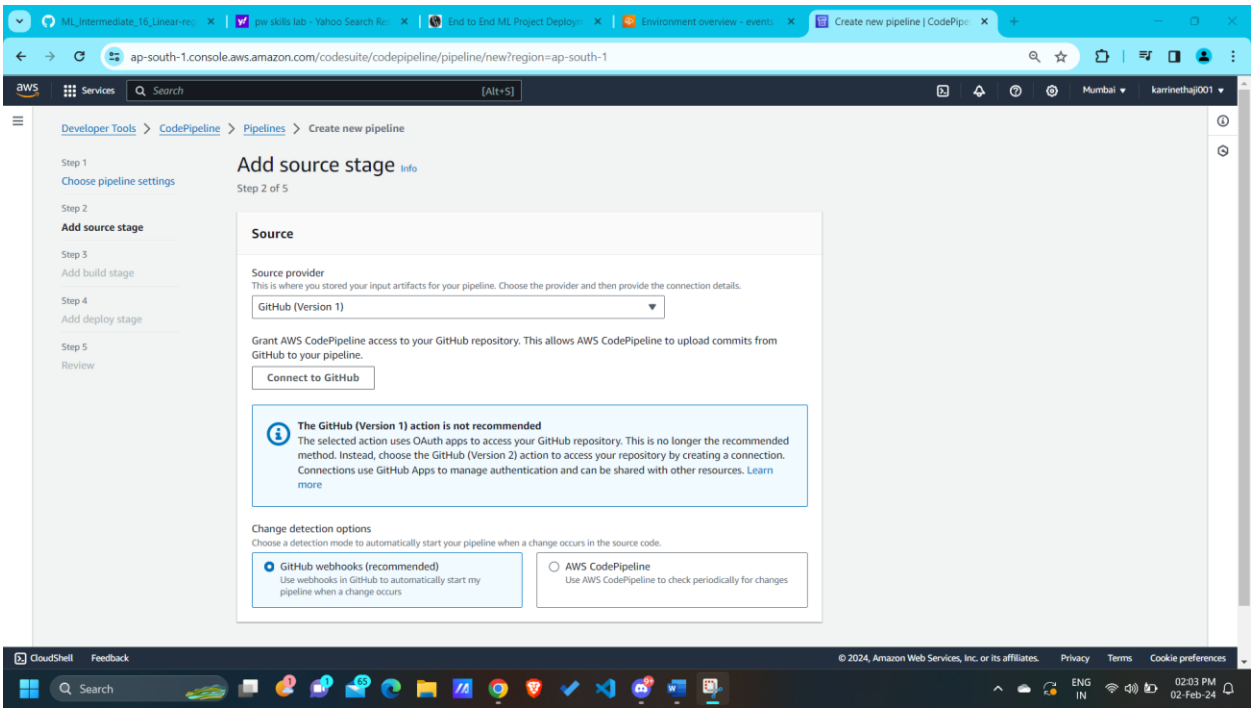



➔ Click on “**CodePipeline**”

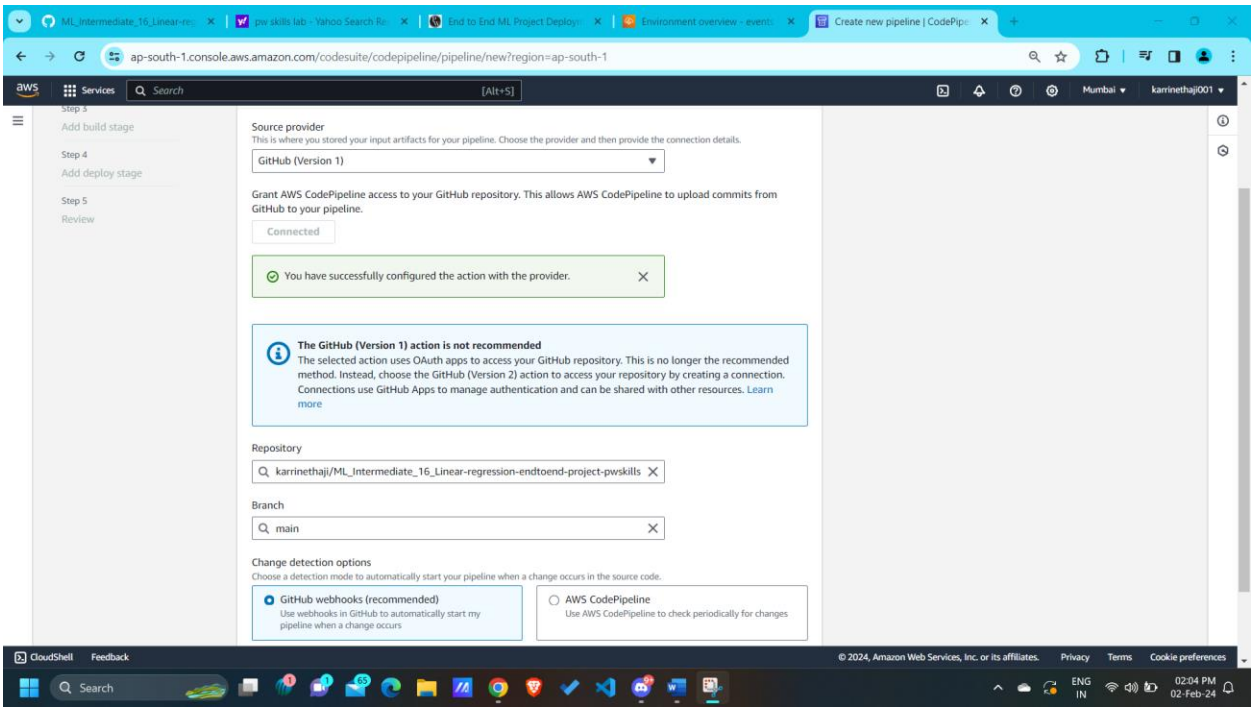


➔ Click on “**Create pipeline**”

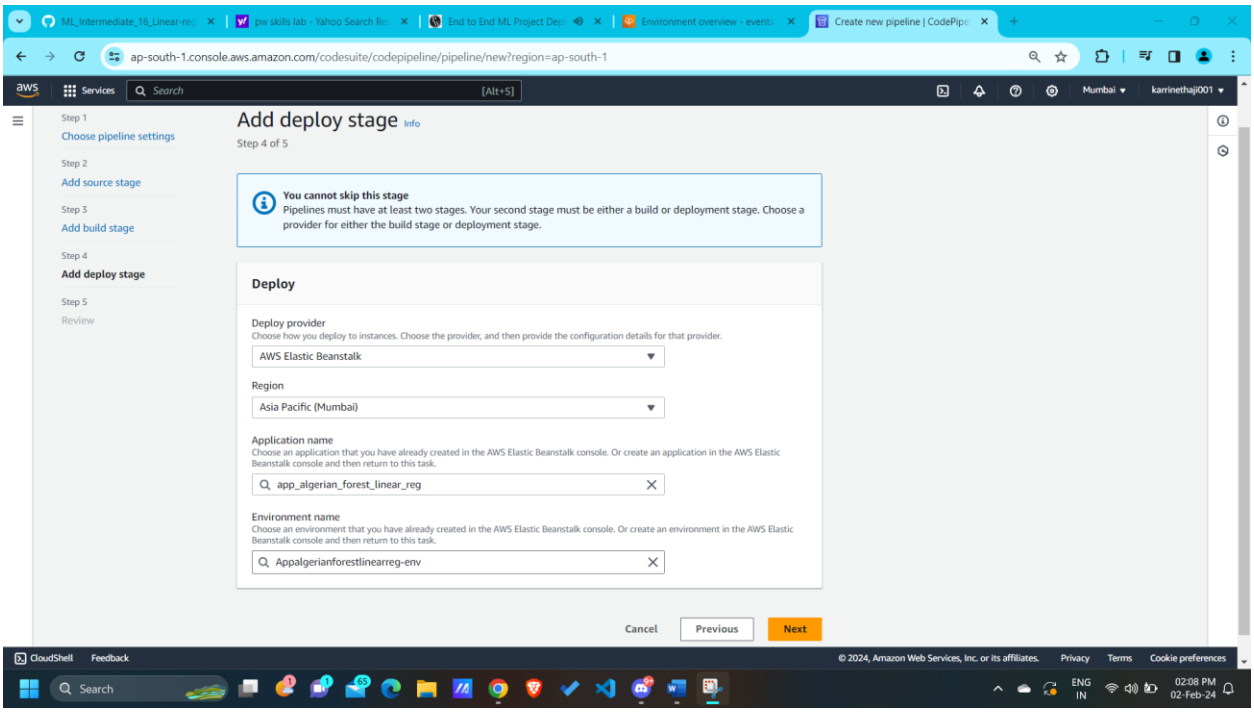


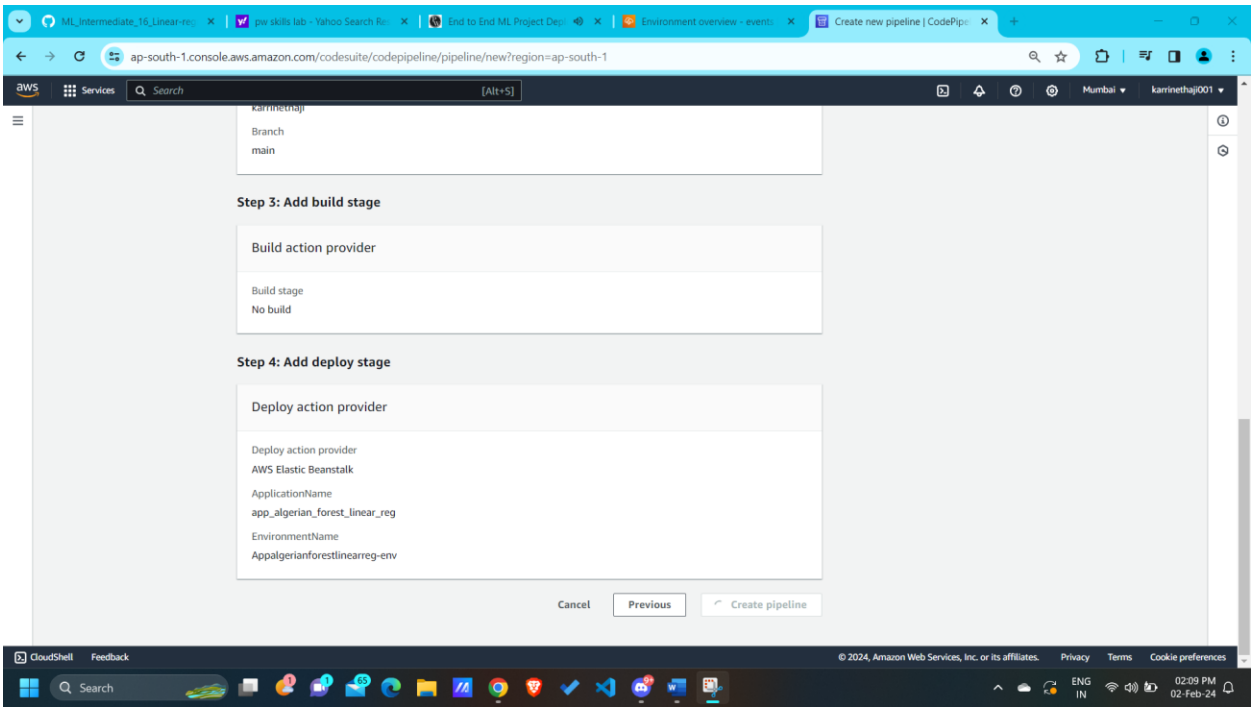

➔ Connect to GitHub and select the repository, branch




➔ **Skip the stage 3**: Build stage

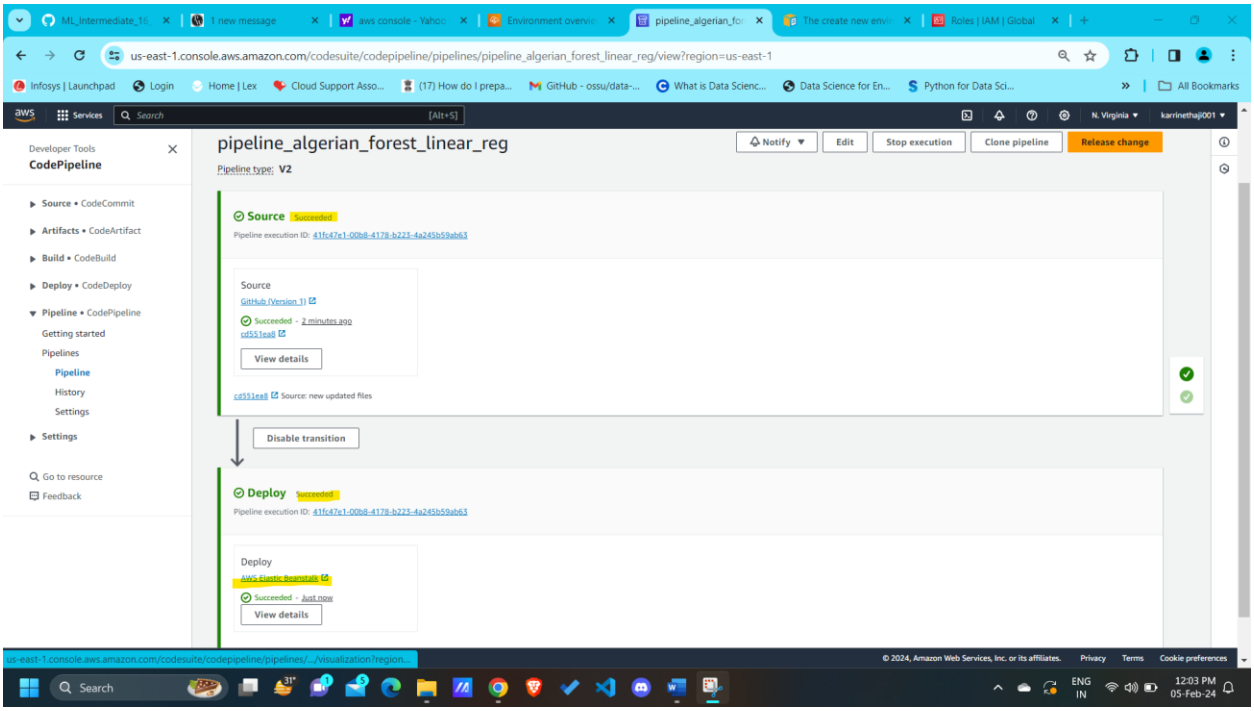



➔ In stage 4 select the deploy provider as "**AWS Beanstalk**" and select the application_name and environment name.

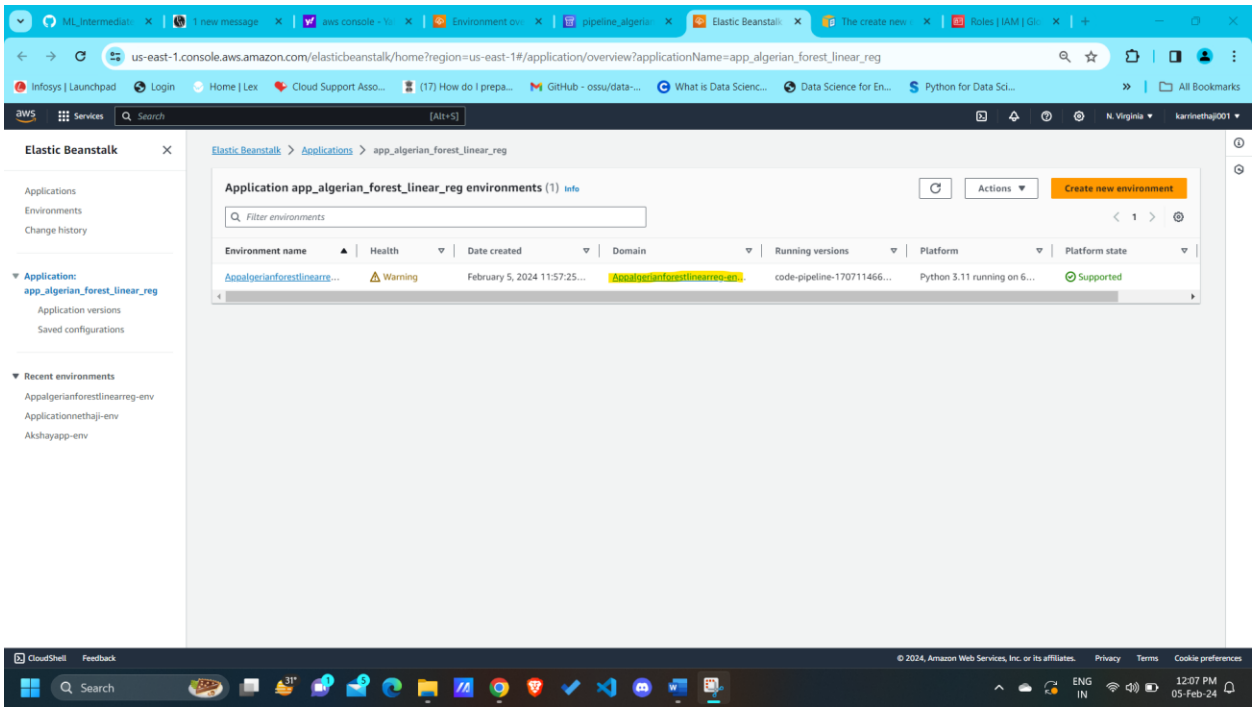



➔ In the review step, Once the elastic beanstalk environment (Appalgerianforestlinearreg-env) deployed successfully, then click on "**Create pipeline**"

➔ Wait for few minutes, it will code pipeline will push the code to elastic beanstalk.

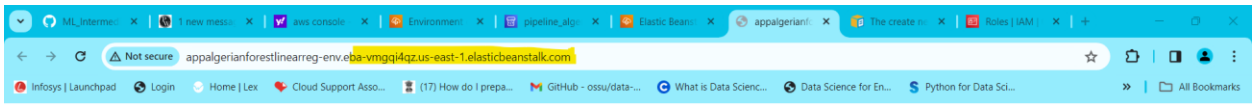



➔ Click on "**AWS Elastic Beanstalk**"

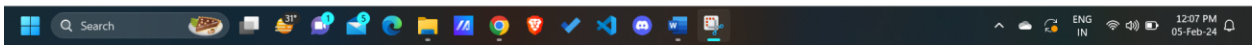



➔ Click on “Domain” (highlighted)

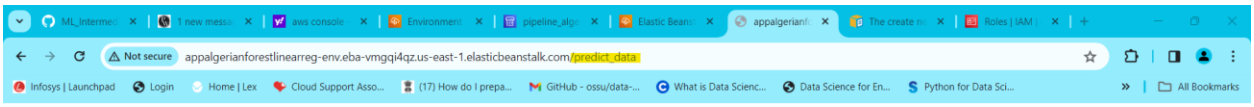
➔ Our output will shown in the aws cloud environment.




**Hi welcome to home page of flask application**



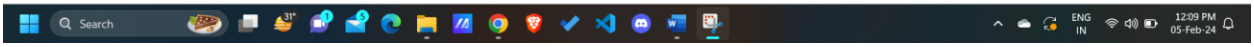
➔ Enter the predict_data in the URL to access predict_data URL.

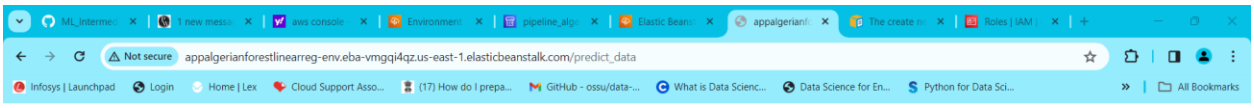



# FWI Prediction

Temperature
RH
Ws
Rain
FFMC
DMC
ISI
Classes
Region
Predict

**THE FWI prediction is**

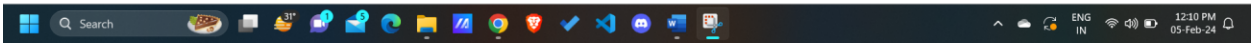



# FWI Prediction

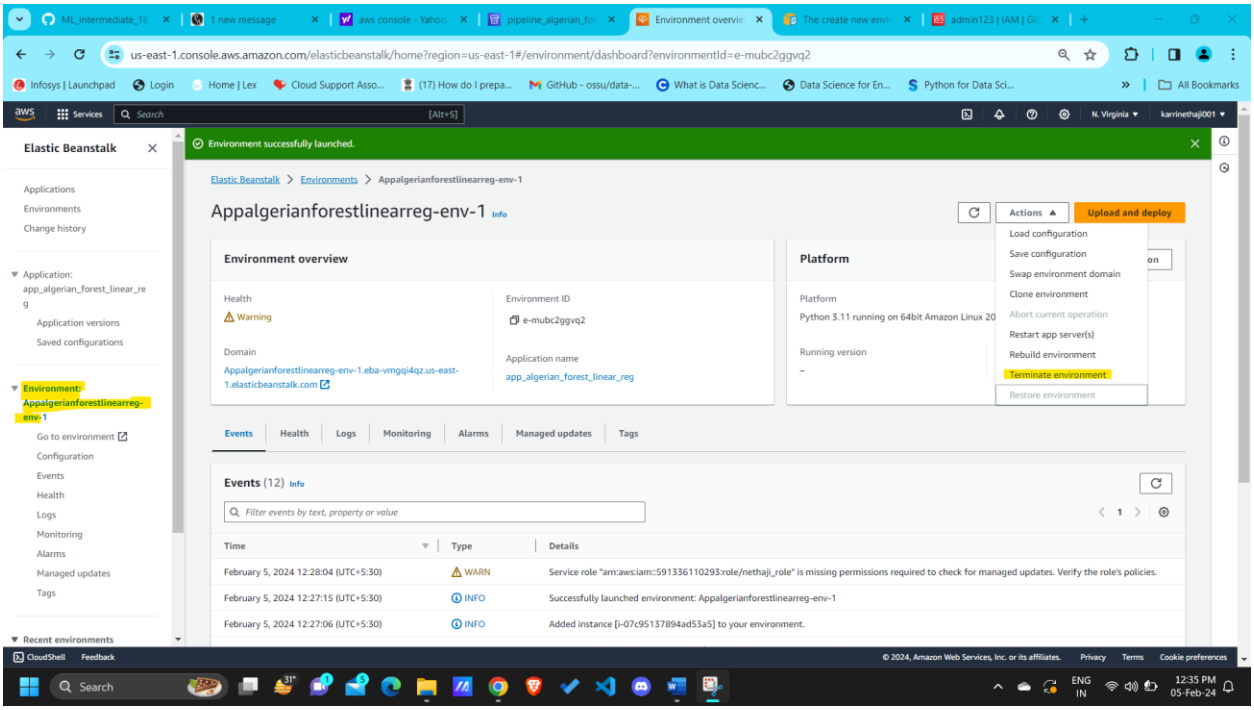
Temperature
RH
Ws
Rain
FFMC
DMC
ISI
Classes
Region
Predict

**THE FWI prediction is 8.422212854315845**

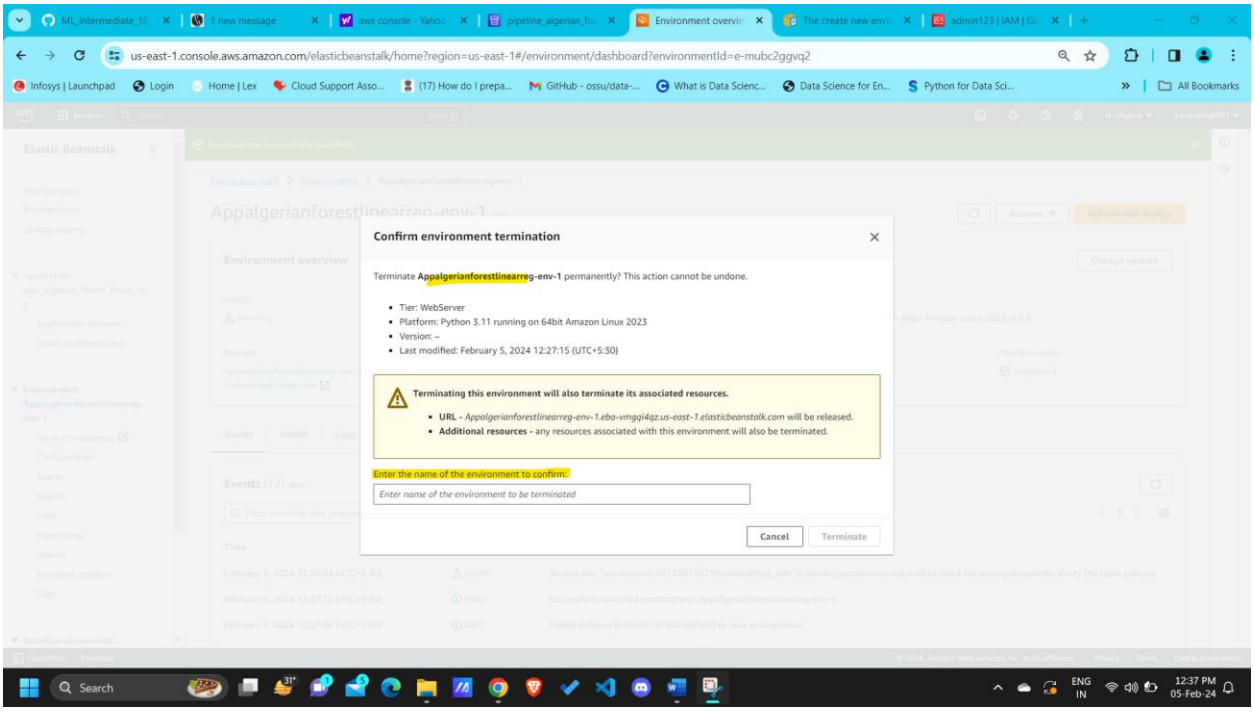


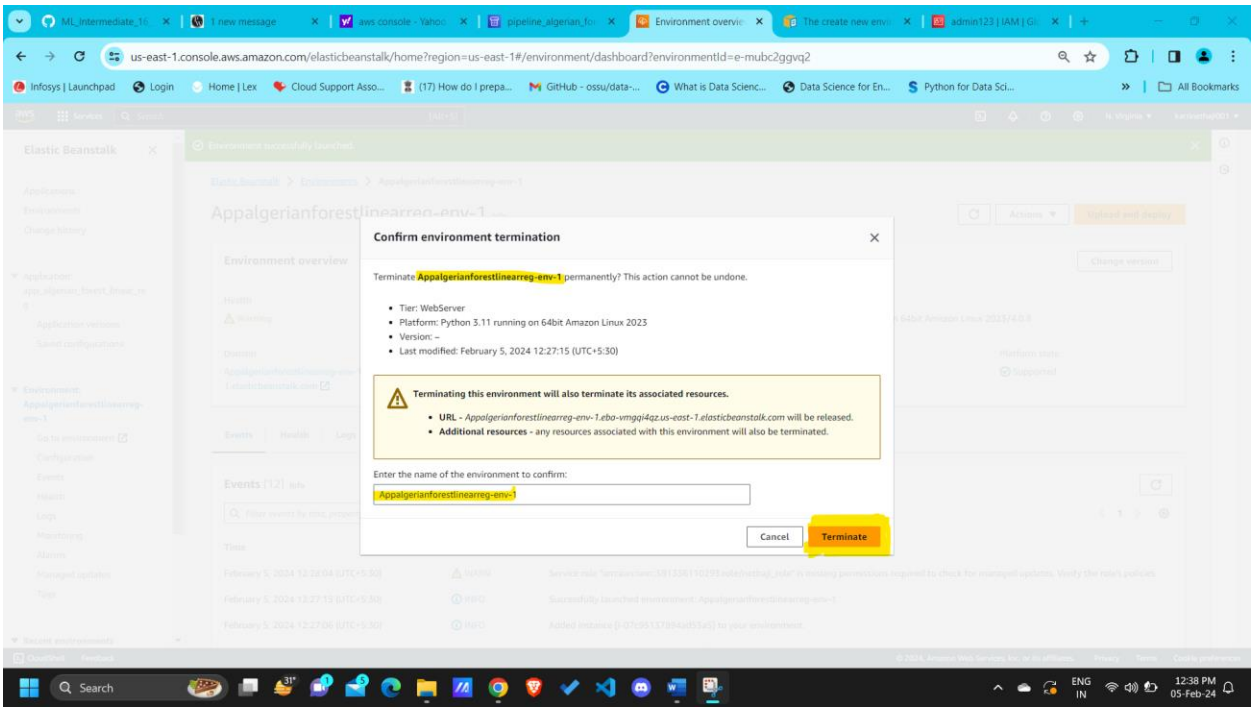
After deploying the project, **terminate the environment** using the below procedure:

**Terminating the environment:**




➔ Click on "**Terminate environment**"






➔ Click on "**Terminate**"

< ---------- **Process to deploy the code in AWS elastic beanstalk** ---------->