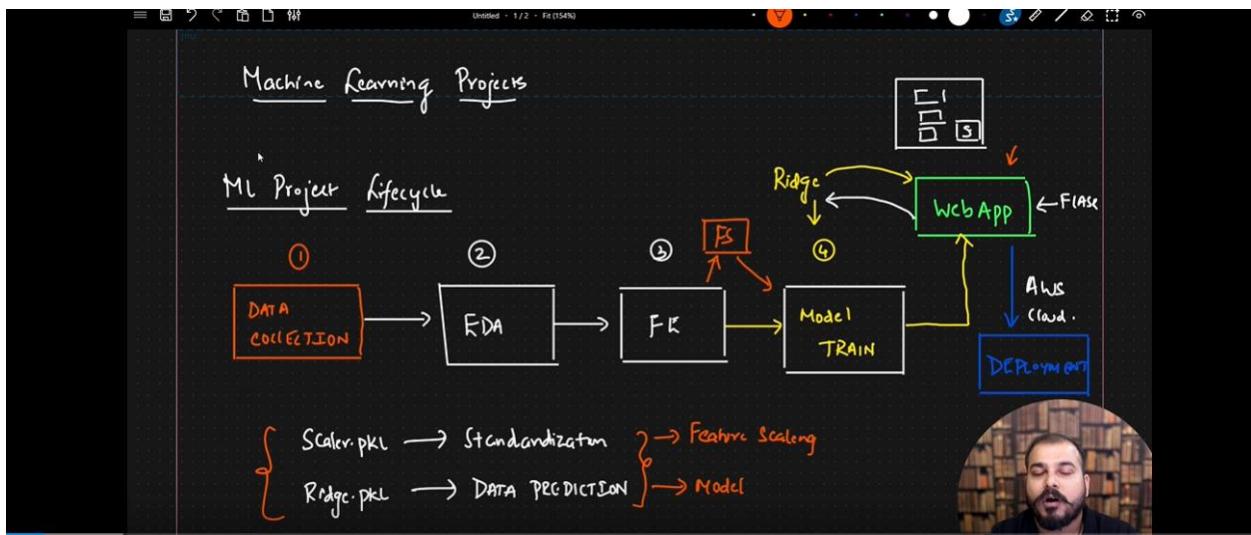


List of steps involved to deploy the ML project:



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/karinethaji/ML_Intermediate_16_Linear-regression-endtoend-project-pwskills

Go to VS Code and develop the below code:

```

<html>
<body>
  <div class="login">
    <h1>FWI Prediction</h1>
    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict_datapoint') }}" method="post"> <!-- method == post , it used to post the values of below parameters
      <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="DTC" 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>
    </form>
    <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
  </div>
  <h2> THE FWI prediction is {{result}} <!-- result value will be replaced with prediction value, check in the application.py file-->
</h2>
</body>
</html>

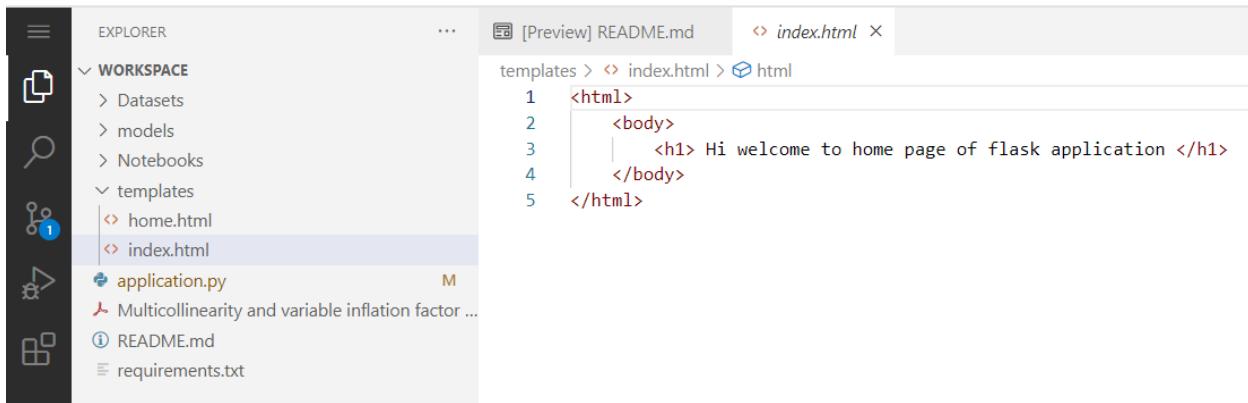
```

Application.py → flask application

Go through the code and understand the things.

Templates:

Index.html:



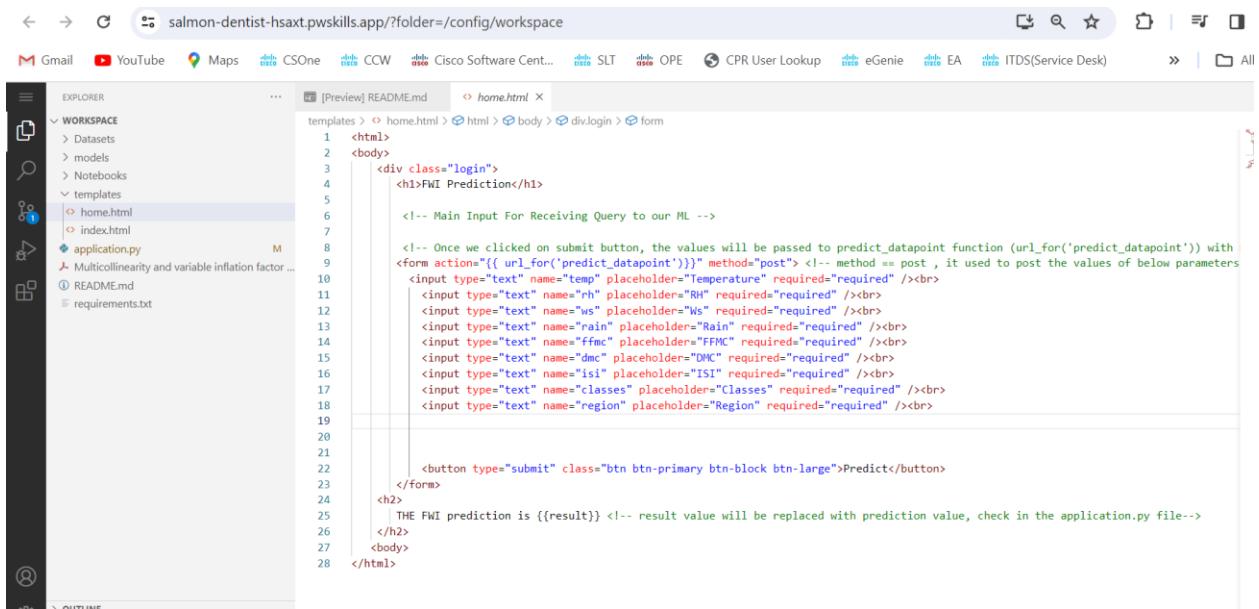
The screenshot shows a Jupyter Notebook interface with the following file structure in the left sidebar:

- WORKSPACE
 - Datasets
 - models
 - Notebooks
 - templates
 - home.html
 - index.html
 - application.py
 - Multicollinearity and variable inflation factor ...
 - README.md
 - requirements.txt

The right pane displays the content of `index.html`:

```
1 <html>
2   <body>
3     |   <h1> Hi welcome to home page of flask application </h1>
4   </body>
5 </html>
```

Home.html:



The screenshot shows a Jupyter Notebook interface with the following file structure in the left sidebar:

- WORKSPACE
 - Datasets
 - models
 - Notebooks
 - templates
 - home.html
 - index.html
 - application.py
 - Multicollinearity and variable inflation factor ...
 - README.md
 - requirements.txt

The right pane displays the content of `home.html`:

```
1 <html>
2   <body>
3     |   <div class="login">
4       |     <h1>FWI Prediction</h1>
5
6     |     <!-- Main Input For Receiving Query to our ML -->
7
8     |     <!-- Once we clicked on submit button, the values will be passed to predict_datapoint function (url_for('predict_datapoint')) with
9     |     <form actions="{{ url_for('predict_datapoint') }}" method="post"> <!-- method == post , it used to post the values of below parameters
10    |     <input type="text" name="temp" placeholder="Temperature" required="required" /><br>
11      |     <input type="text" name="rh" placeholder="Rh" required="required" /><br>
12      |     <input type="text" name="ws" placeholder="Ws" required="required" /><br>
13      |     <input type="text" name="rain" placeholder="Rain" required="required" /><br>
14      |     <input type="text" name="fmc" placeholder="FFMC" required="required" /><br>
15      |     <input type="text" name="dmc" placeholder="DMC" required="required" /><br>
16      |     <input type="text" name="isi" placeholder="ISI" required="required" /><br>
17      |     <input type="text" name="classes" placeholder="Classes" required="required" /><br>
18      |     <input type="text" name="region" placeholder="Region" required="required" /><br>
19
20
21
22     |     <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
23   |   </form>
24   | </div>
25   | THE FWI prediction is {{result}} <!-- result value will be replaced with prediction value, check in the application.py file-->
26   | </h2>
27   | </body>
28 </html>
```

Application.py: → flask application

salmon-dentist-hsaxt.pwskills.app/?folder=/config/workspace

```

EXPLORER [Preview] README.md application.py X
WORKSPACE
> Datasets
> models
> Notebooks
templates
  o home.html
  o index.html
application.py M
  Multicollinearity and variable inflation factor...
  README.md
  requirements.txt

application.py
1  from flask import Flask, request, jsonify , render_template # jsonify is used to return the value in form of json (dictionay)
2  import pickle # to pickle our model
3  import pandas as pd
4  import numpy as np
5  from sklearn.preprocessing import StandardScaler
6
7  application = Flask(__name__)
8
9  app= application
10
11 # Import ridge regression and sclae objects for predicing the output based on the data
12
13 ridge= pickle.load(open("models/ridge.pkl","rb"))
14 scaler= pickle.load(open("models/scaler.pkl","rb"))
15
16 @app.route("/")
17 def home():
18     return render_template("index.html")
19     # return render_template("home.html") # uncomment this and comment the above line and observe the output
20
21 @app.route("/predict_data",methods=["GET","POST"])
22 def predict_datapoint():
23     if request.method=="POST": # after clicking the submit button in the home.html, the data will be retrieved using POST method
24         Temperature = float(request.form.get("temp"))
25         RH = float(request.form.get("RH"))
26         Ws = float(request.form.get("Ws"))
27         Rain = float(request.form.get("rain"))
28         FFMC = float(request.form.get("ffmc"))
29         DMC = float(request.form.get("dmc"))
30         ISI = float(request.form.get("isi"))
31         Classes = float(request.form.get("classes"))
32         Region = float(request.form.get("region"))

# sciae 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
if __name__ == "__main__":
    app.run(host="0.0.0.0",debug=True)

```

salmon-dentist-hsaxt.pwskills.app/?folder=/config/workspace

```

EXPLORER [Preview] README.md application.py M index.html
WORKSPACE
> Datasets
> models
> Notebooks
templates
  o home.html
  o index.html
application.py M
  Multicollinearity and variable inflation factor...
  README.md
  requirements.txt

application.py
21 @app.route("/predict_data",methods=["GET","POST"])
22 def predict_datapoint():
23     if request.method=="POST": # after clicking the submit button in the home.html, the data will be retrieved using POST method
24         Temperature = float(request.form.get("temp"))
25         RH = float(request.form.get("RH"))
26         Ws = float(request.form.get("Ws"))
27         Rain = float(request.form.get("rain"))
28         FFMC = float(request.form.get("ffmc"))
29         DMC = float(request.form.get("dmc"))
30         ISI = float(request.form.get("isi"))
31         Classes = float(request.form.get("classes"))
32         Region = float(request.form.get("region"))

# sciae 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

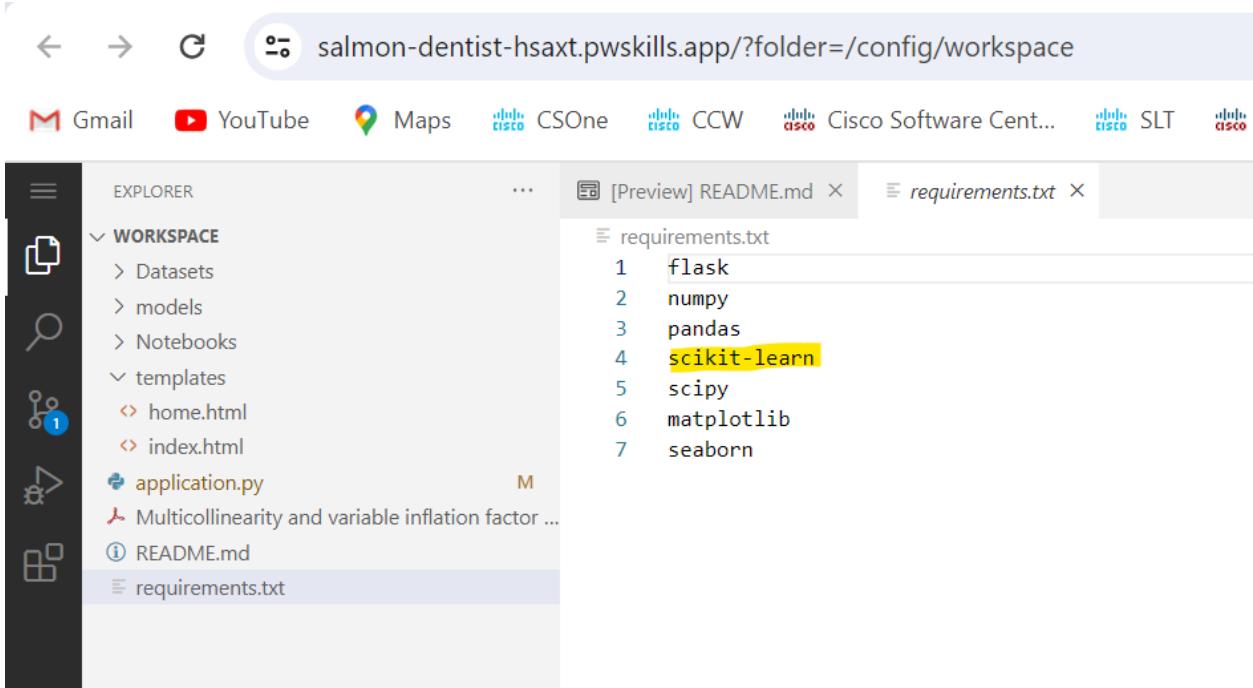
return render_template("home.html",result=output[0]) # Output value will be rendered in the home.html page
else:
    return render_template("home.html") # It will render the home.html page
if __name__ == "__main__":
    app.run(host="0.0.0.0",debug=True)

```

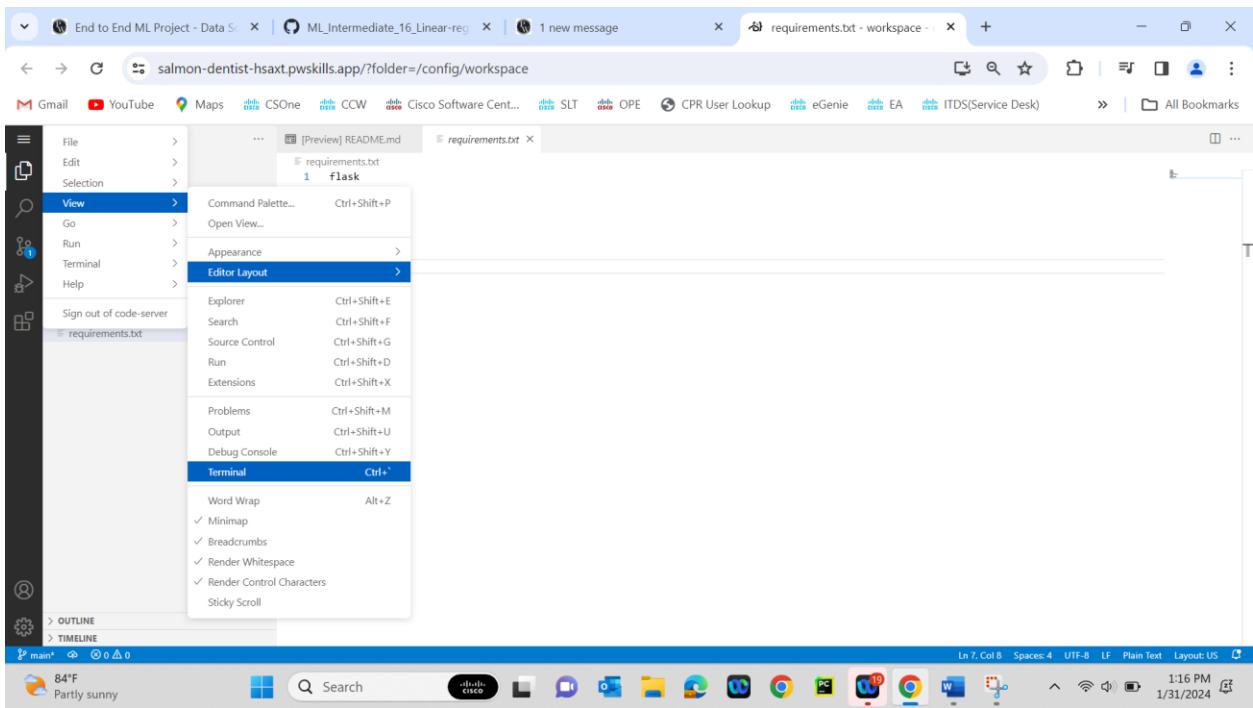
**** port=5000 , we need to specify the same port number in URL # 5000 is the default URL**

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.

The screenshot shows a Jupyter Notebook interface with the following details:

- EXPLORER:** Shows the workspace structure with files like README.md, requirements.txt, application.py, and various HTML templates.
- TERMINAL:** Shows the command `abc@977f1ec25ead:~/workspace\$ pip install -r requirements.txt` being run, followed by the output of package installations (flask, numpy, pandas, scikit-learn).
- STATUS BAR:** Shows the date (1/31/2024), time (1:17 PM), and file path (1.main*).

- 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"))

# scale 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:

```

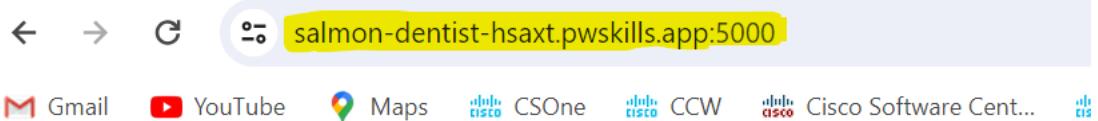
6     |     | <!-- Main Input For Receiving Query to our ML -->
7
8     |     | <!-- Once we clicked on submit button, the values will be passed to predict_datapoint function (url
9     |     | <form action="{{ url_for('predict_datapoint') }}" method="post"> <!-- method == post , it used to pos
0     |     |     <input type="text" name="temp" placeholder="Temperature" required="required" /><br>
.1    |     |     <input type="text" name="rh" placeholder="RH" required="required" /><br>
.2    |     |     <input type="text" name="ws" placeholder="Ws" required="required" /><br>
.3    |     |     <input type="text" name="rain" placeholder="Rain" required="required" /><br>
.4    |     |     <input type="text" name="ffmc" placeholder="FFMC" required="required" /><br>
.5    |     |     <input type="text" name="dmc" placeholder="DMC" required="required" /><br>
.6    |     |     <input type="text" name="isi" placeholder="ISI" required="required" /><br>
.7    |     |     <input type="text" name="classes" placeholder="Classes" required="required" /><br>
.8    |     |     <input type="text" name="region" placeholder="Region" required="required" /><br>

```

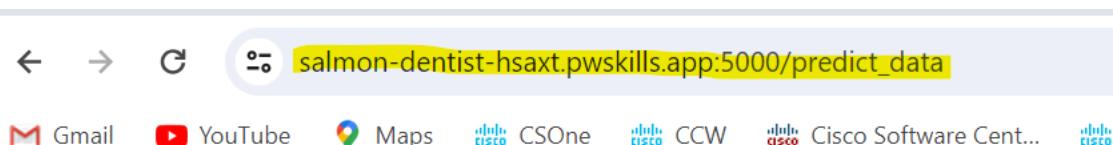
→ In the home.html content, we have provided the method="POST".

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



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)



FWI Prediction

23
22
3
3
45
3
3
44
56
Predict

THE FWI prediction is [REDACTED]

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"))
```

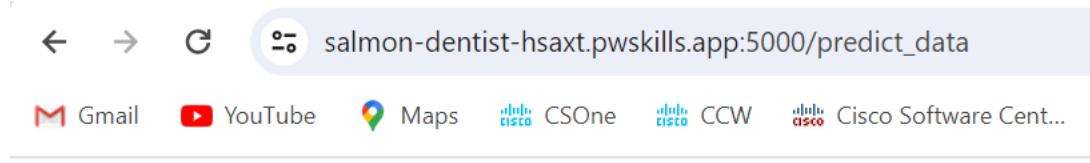
- We scaled the data using the standard scaler pickle object and we predicted the output using the ridge pickle object

- 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-->
```



FWI Prediction

Temperature
RH
Ws
Rain
FFMC
DMC
ISI
Classes
Region
Predict

THE FWI prediction is -0.036487576241507114

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

...or create a new repository on the command line

```
echo "# testforestfires" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/krishnaik06/testforestfires.git
git push -u origin main
```

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>

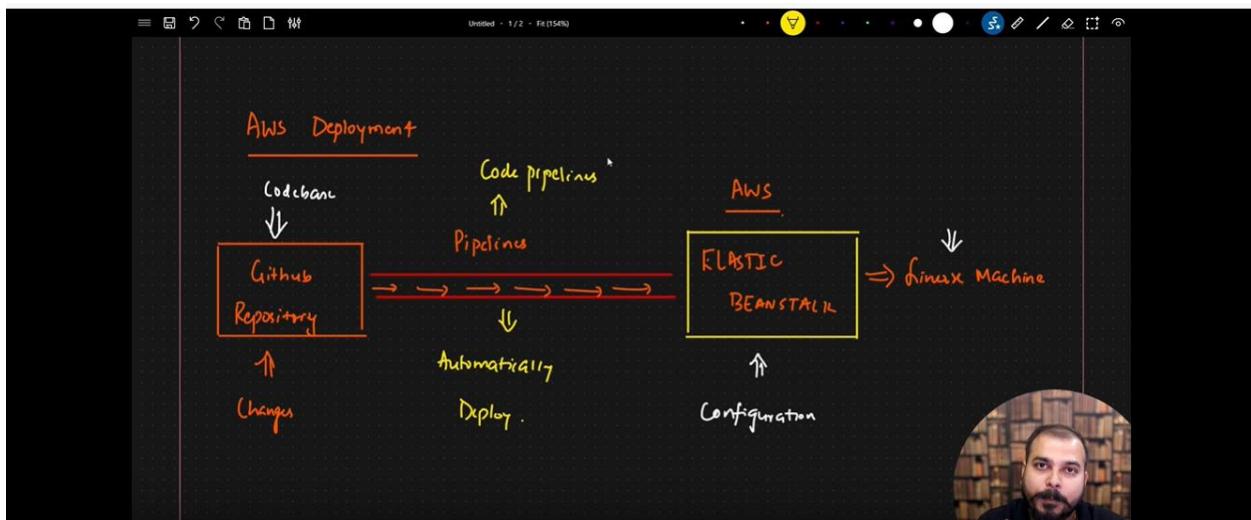
NOTE: If the code is not pushed into github after using the command “git push -u origin main” then use “git push -f origin main” instead.

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)

```
option_settings:
  "aws:elasticbeanstalk:container:python":
    WSGIPath: application:application
```

**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 .
④ 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 9eba11] 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...9eba11 main -> main (forced update)
```

Before pushing code:

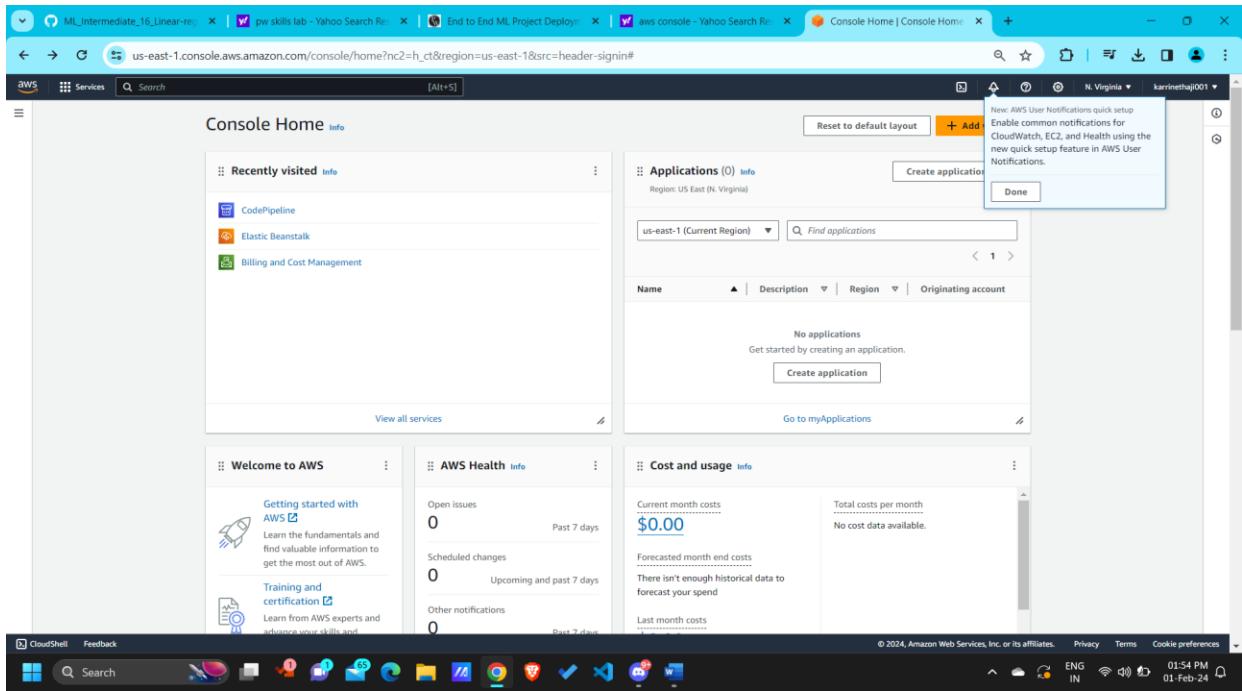
The screenshot shows a GitHub repository interface. On the left, the file tree displays a main folder with subfolders 'Datasets', 'Notebooks', 'models', and 'templates'. Inside 'Datasets' are files 'Algerian_forest.csv' and 'algerian_forest_cleaned.csv'. Other files like 'README.md', 'application.py', and 'requirements.txt' are located in the root and 'templates' folder. On the right, the 'Datasets' folder is selected, showing a commit history for 'Algerian_forest.csv' and 'algerian_forest_cleaned.csv' with the message 'first commit'.

After pushing code:

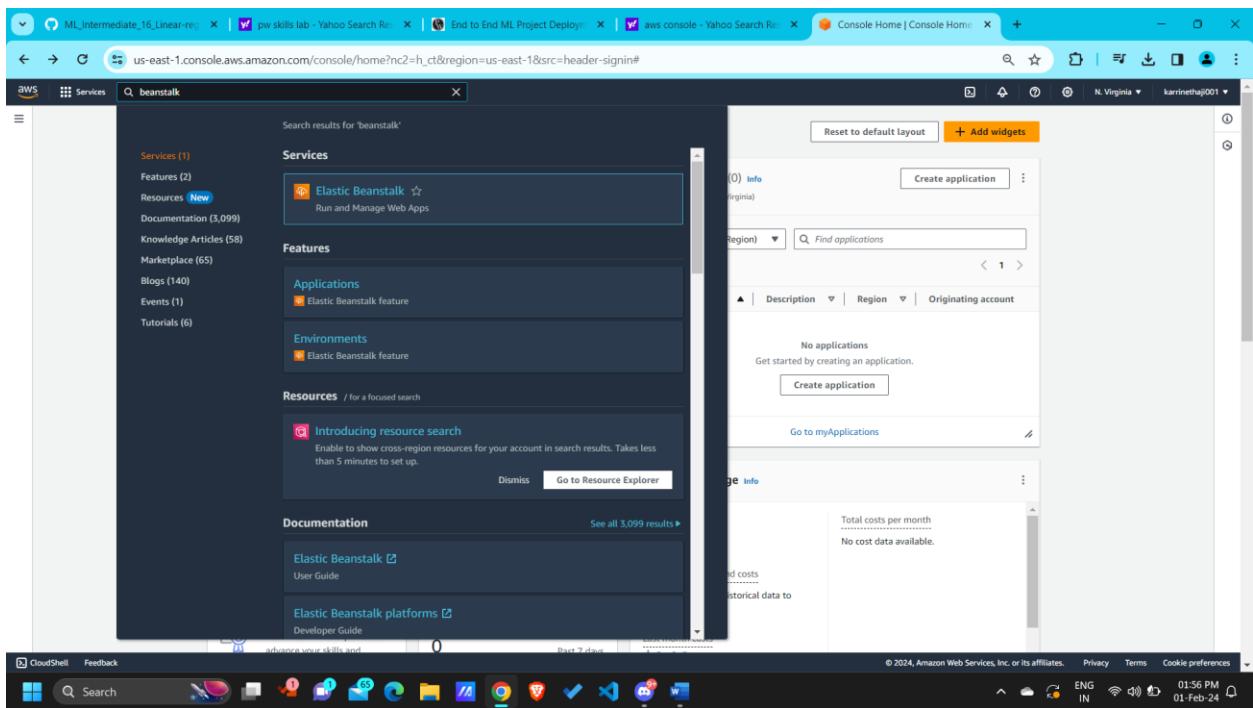
The screenshot shows the same GitHub repository after pushing code. The file tree now includes a new folder 'ebextensions' containing a file 'python.config'. This file contains configuration for AWS Elastic Beanstalk, specifically setting the container to 'python' and the WSGI path to 'application:application'. The commit history for this file shows a commit from 'karinethaji' with the message 'elastic beanstalk config'.

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

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



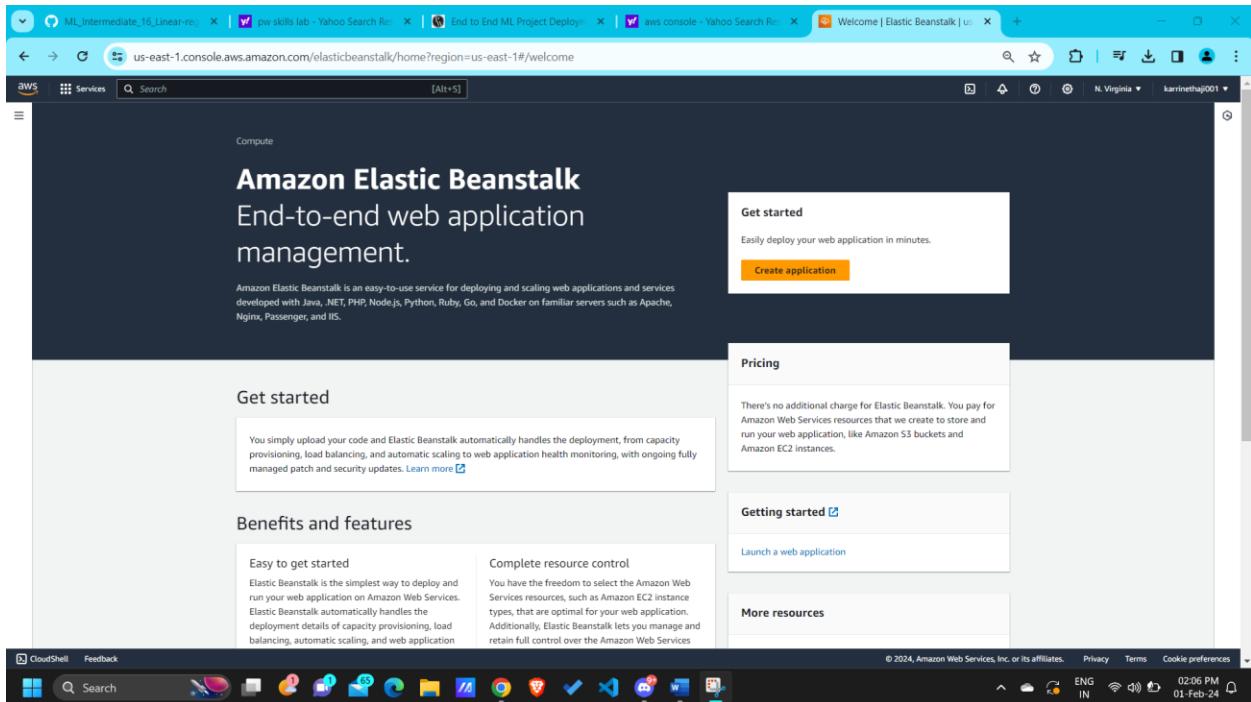
Search as elastic beanstalk and select the **Elastic Beanstalk** service



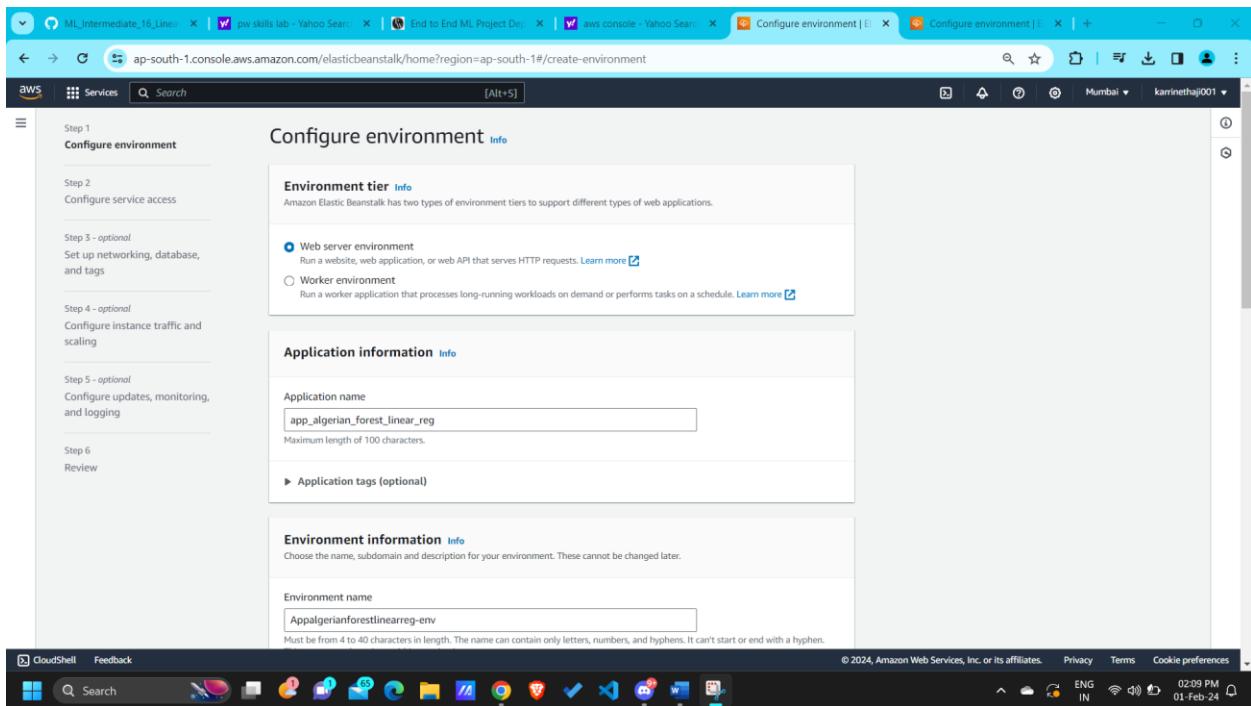
→ Select the “**Elastic Beanstalk**” service and you will get a UI interface as below

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

ap-south-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-south-1#/create-environment

Domain
Leave blank for autogenerated value .ap-south-1.elasticbeanstalk.com

Environment description

Platform Info

Platform type
 Managed platform Platforms published and maintained by Amazon Elastic Beanstalk. Learn more [\[?\]](#)
 Custom platform Platforms created and owned by you. This option is unavailable if you have no platforms.

Platform

Platform branch

Platform version

Application code Info

Sample application
 Existing version Application versions that you have uploaded.
 Upload your code Upload a source bundle from your computer or copy one from Amazon S3.

Presets Info

Start from a preset that matches your use case or choose custom configuration to unset recommended values and use the service's default values.

Configuration presets
 Single instance (free tier eligible)
 Single instance (using spot instance)
 High availability
 High availability (using spot and on-demand instances)
 Custom configuration

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

Application code [Info](#)

Sample application

Existing version
Application versions that you have uploaded.

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

Version label
Unique name for this version of your application code.

Source code origin. Maximum size 500 MB

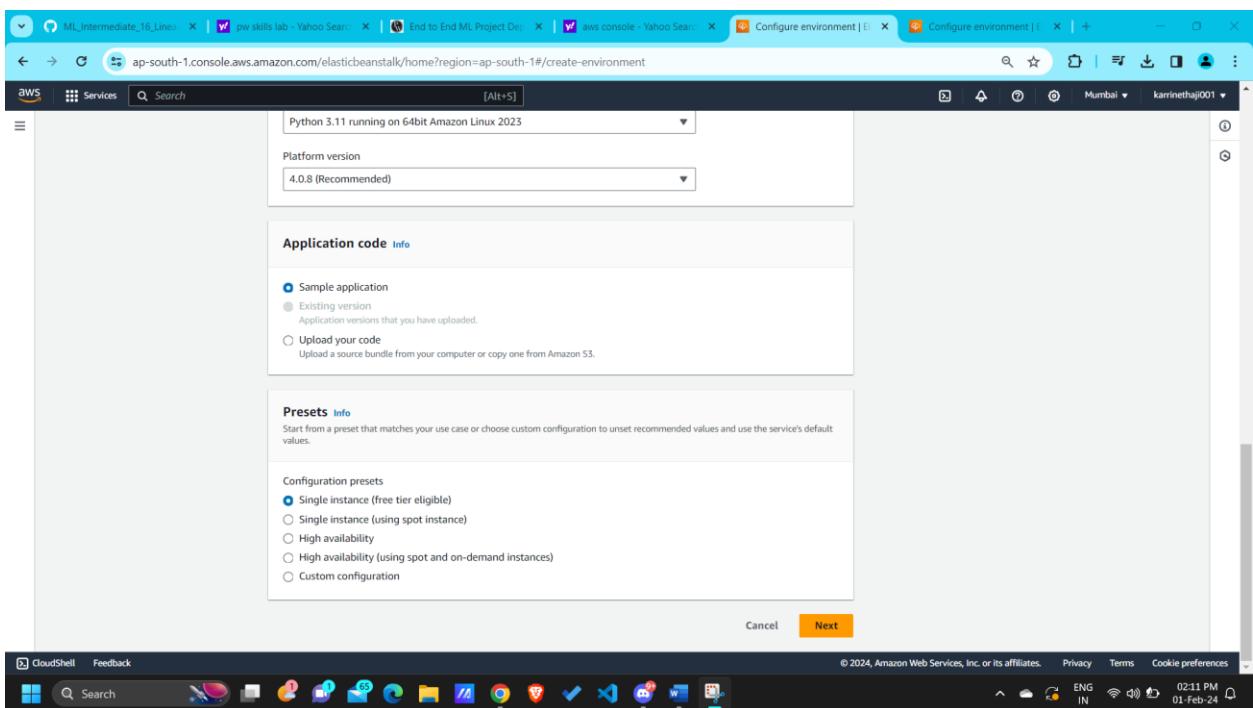
Local file

Upload application

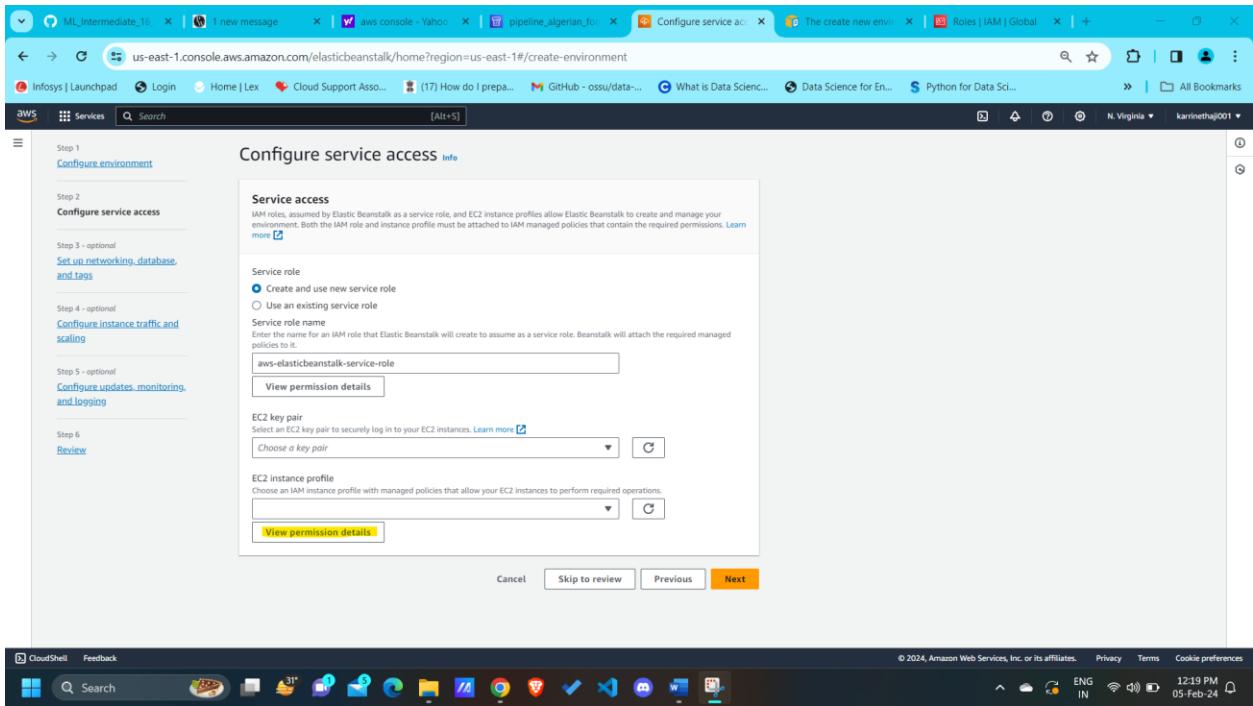
File must be less than 500MB max file size

Public S3 URL

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”



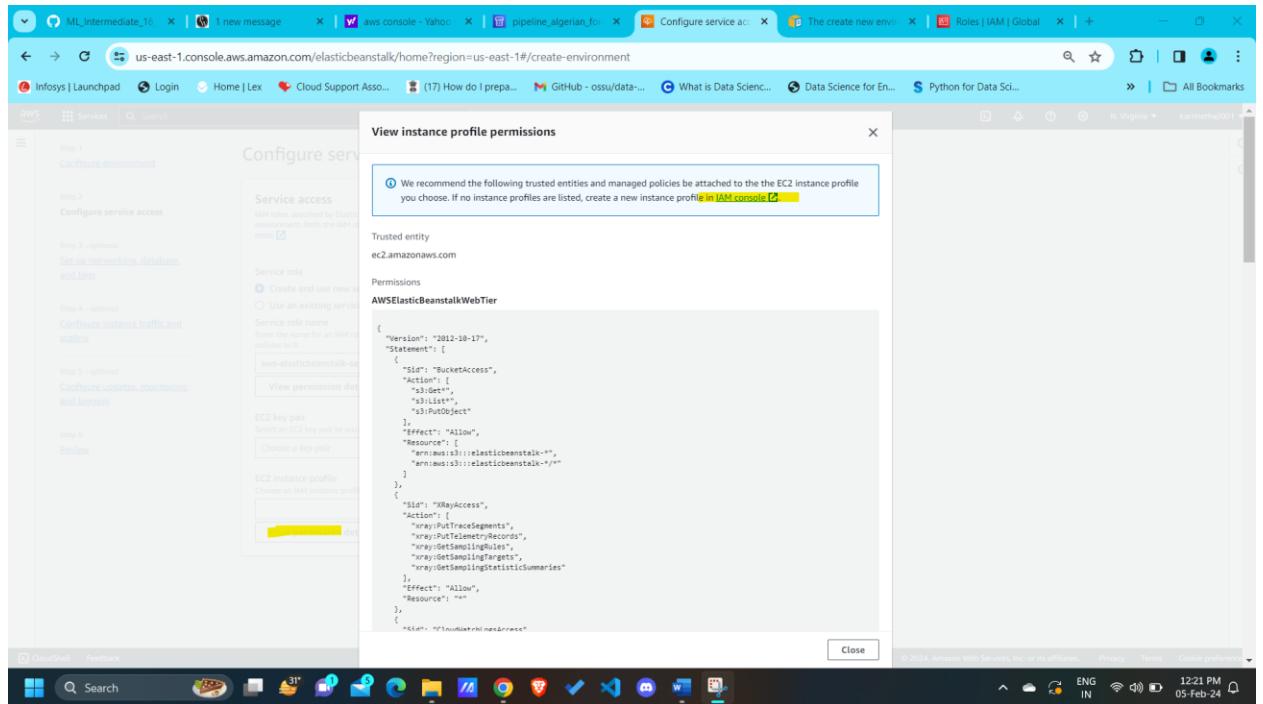
- If you are using the elastic beanstalk for the first time, then Create a **IAM role** using the below procedure:

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

Configure service access

Service access

IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. [Learn more](#)

Service role

Create and use new service role
 Use an existing service role

Existing service roles

Choose an existing IAM role for Elastic Beanstalk to assume as a service role. The existing IAM role must have the required IAM managed policies.

nethay_role

EC2 key pair

Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

Choose a key pair

EC2 instance profile

Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

nethay_role

[View permission details](#)

Cancel Skip to review Previous Next

→ Click on “Skip to review”

Updates

Managed updates	Deployment batch size	Deployment batch size type
Activated	100	Percentage
Command timeout	Deployment policy	Health threshold
600	AllAtOnce	Ok
Ignore health check	Instance replacement	
false	false	

Platform software

Lifecycle	Log streaming	NumProcesses
false	Deactivated	1
NumThreads	WSGIPath	Proxy server
15	application	nginx
Logs retention	Rotate logs	Update level
7	Deactivated	minor

X-Ray enabled

Deactivated

Environment properties

Key	Value
PYTHONPATH	/var/app/venv/staging-LQM1test/bin

Cancel Previous Submit

- 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:

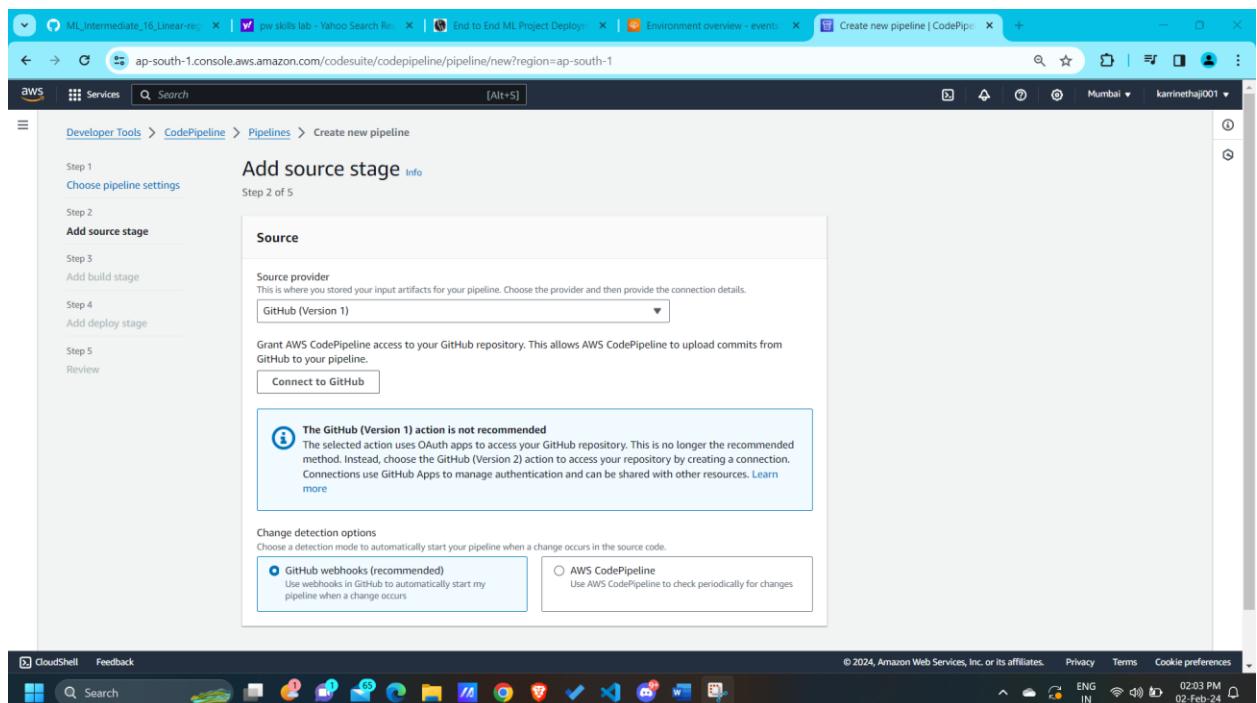
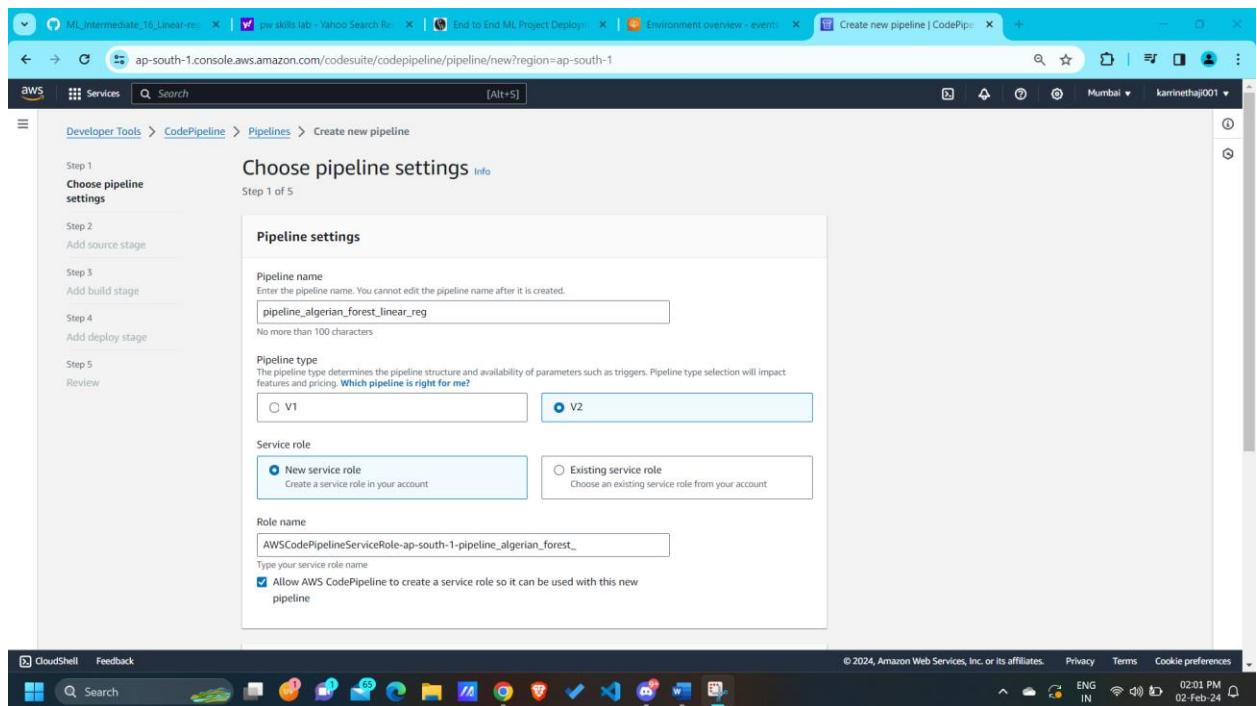
→ Search as **CodePipeline**

The screenshot shows the AWS CloudWatch Metrics search interface. The search bar at the top contains the query "codepipeline". The results pane displays several items under the "Services" category, including "CodePipeline" which is highlighted with a star icon. Other items listed include "Documentation", "Blogs", and "Tutorials". To the right of the search results, there is a sidebar with options like "Create application", "Find applications", and "Create application". The bottom of the screen shows the Windows taskbar with various pinned icons.

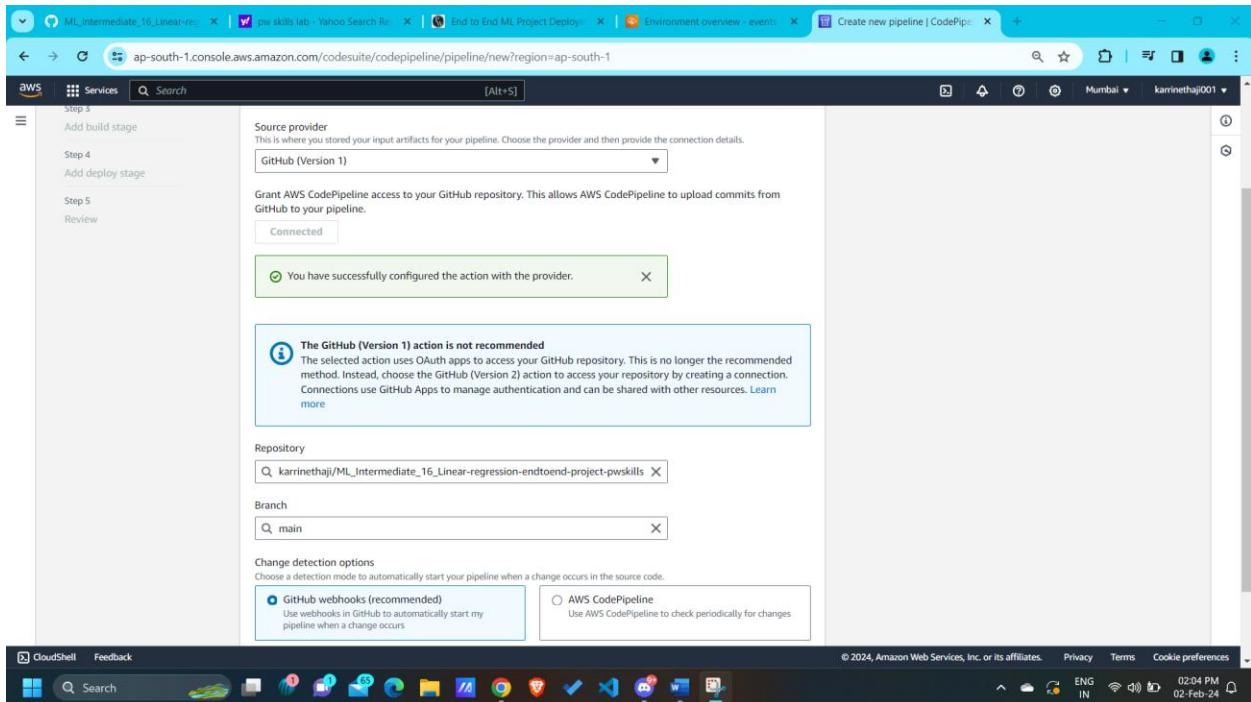
→ Click on "**CodePipeline**"

The screenshot shows the AWS CodePipeline service dashboard. The left sidebar has a "Developer Tools" section with "CodePipeline" selected, which is highlighted in blue. Under "CodePipeline", there are links for "Source", "Artifacts", "Build", "Deploy", "Pipeline", and "Settings". The main content area is titled "Pipelines" and shows a table with one row. The table columns are "Name", "Latest execution status", "Latest execution started", and "Most recent executions". The table body contains the message "No results" and "There are no results to display." At the top of the main content area, there are buttons for "Create pipeline", "Notify", "View history", "Release change", and "Delete pipeline". The bottom of the screen shows the Windows taskbar with various pinned icons.

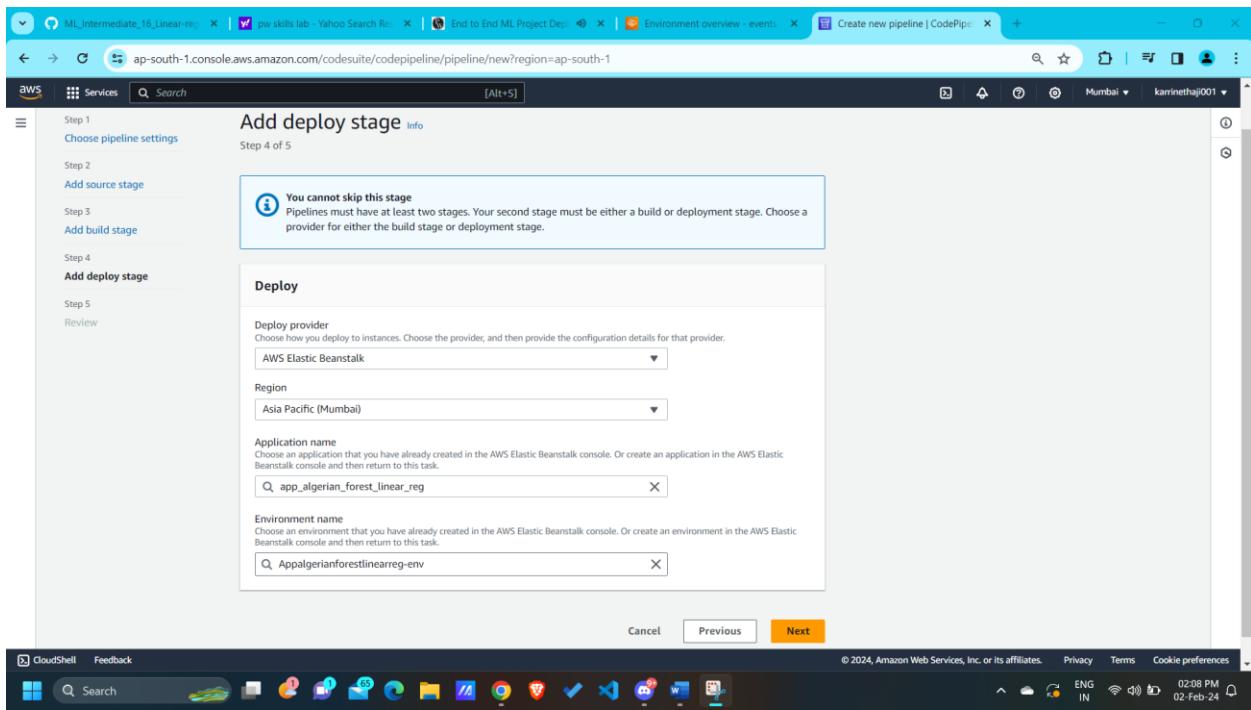
→ Click on “Create pipeline”



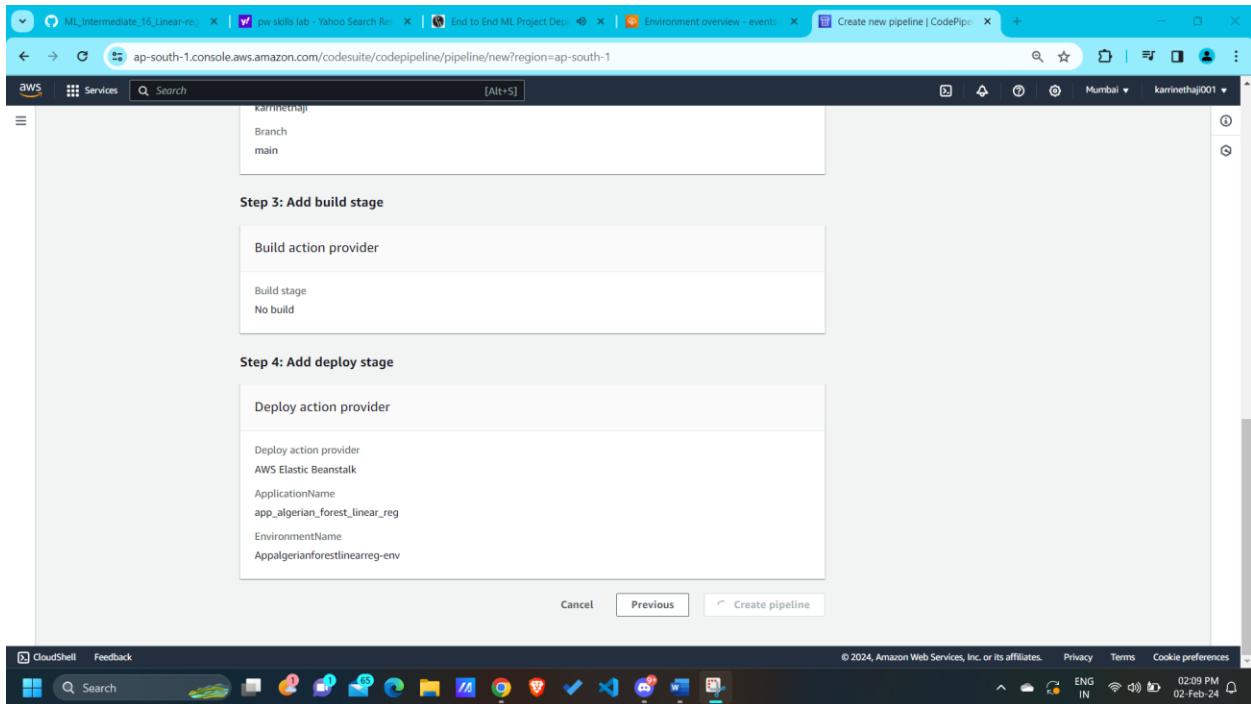
→ Connect to GitHub and select the repository, branch



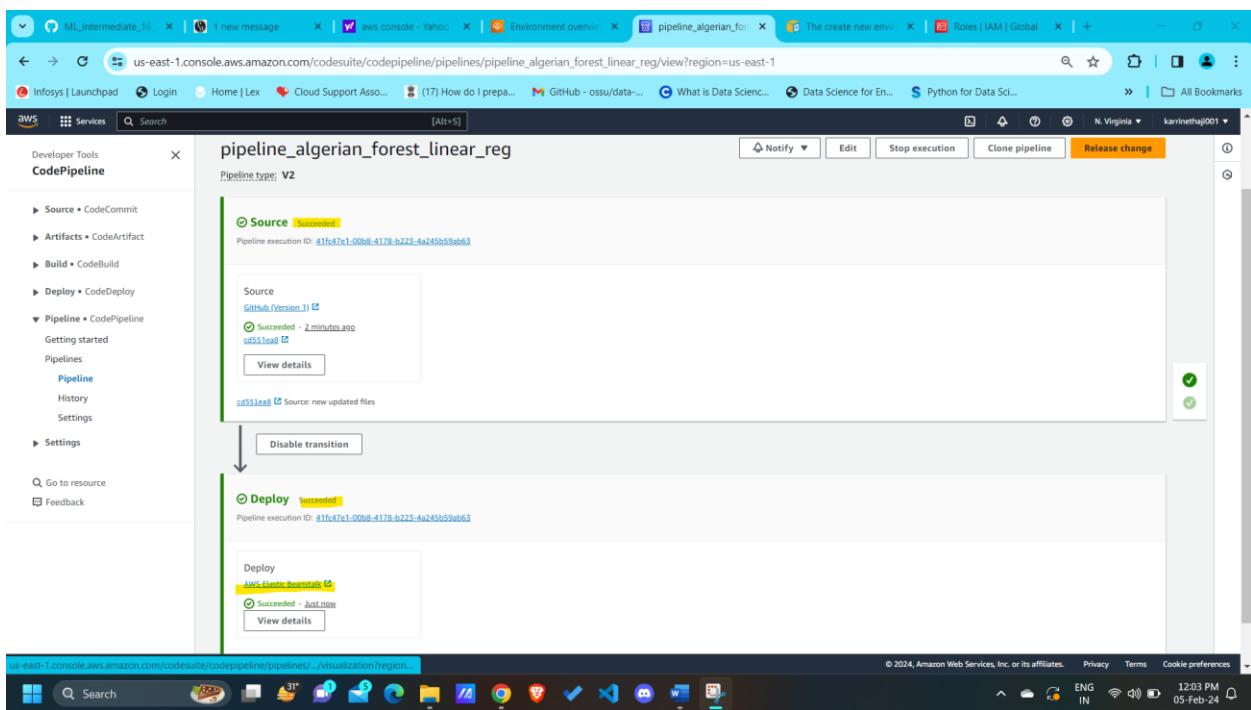
→ Skip the stage 3: Build stage



→ 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**”

The screenshot shows the AWS Elastic Beanstalk console with the URL https://us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/application/overview?applicationName=app_algerian_forest_linear_reg. The left sidebar shows the navigation path: Elastic Beanstalk > Applications > app_algerian_forest_linear_reg. The main content area displays the 'Application app_algerian_forest_linear_reg environments (1) info' table. The table has columns: Environment name, Health, Date created, Domain, Running versions, Platform, and Platform state. One environment is listed: Appalgerianforestlinearreg-env, with a warning icon, created on February 5, 2024, at 11:57:25. The domain is Appalgerianforestlinearreg-env.12345678901234567890123456789012.elasticbeanstalk.com, running version code-pipeline-170711466..., platform Python 3.11, and supported status.

- Click on “Domain” (highlighted)
- Our output will shown in the aws cloud environment.

The screenshot shows a web browser window with the URL <http://Appalgerianforestlinearreg-env.12345678901234567890123456789012.elasticbeanstalk.com>. The page content is "Hi welcome to home page of flask application". The browser toolbar at the bottom shows various icons for different applications like File Explorer, Task View, and Control Panel.

- 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

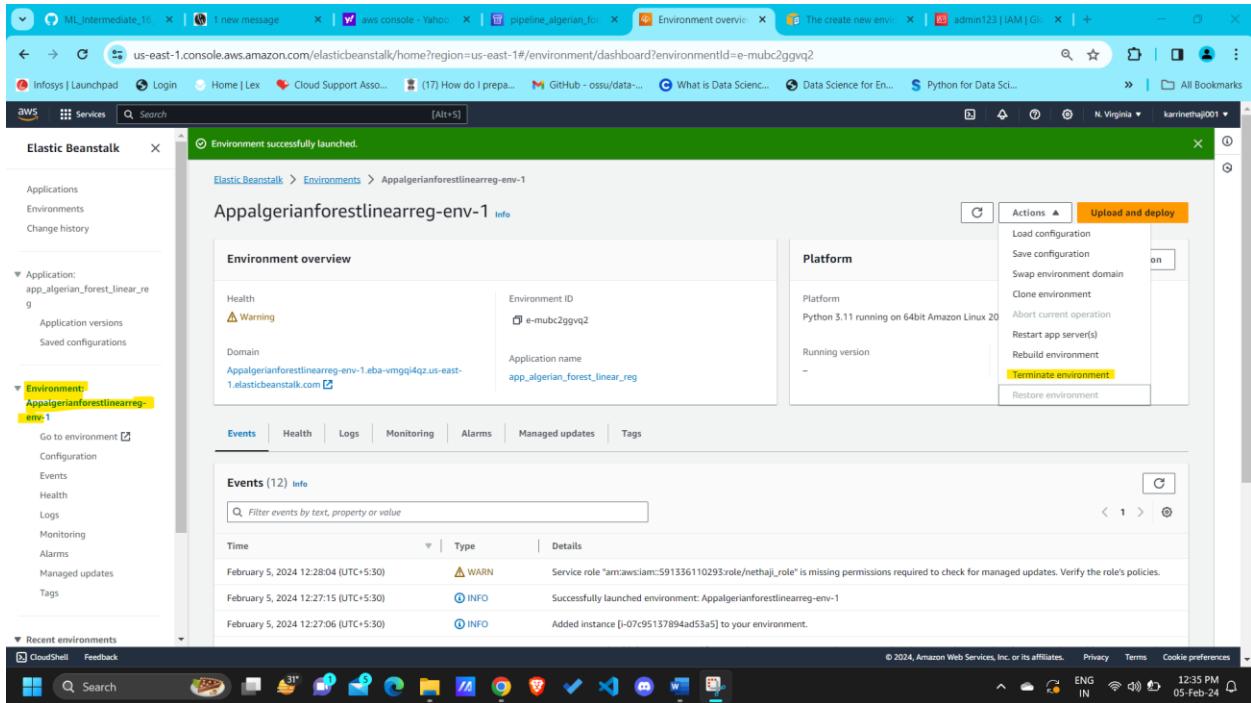
Temperature
RH
Ws
Rain
FFMC
DMC
ISI
Classes
Region
Predict

THE FWI prediction is **8.422212854315845**



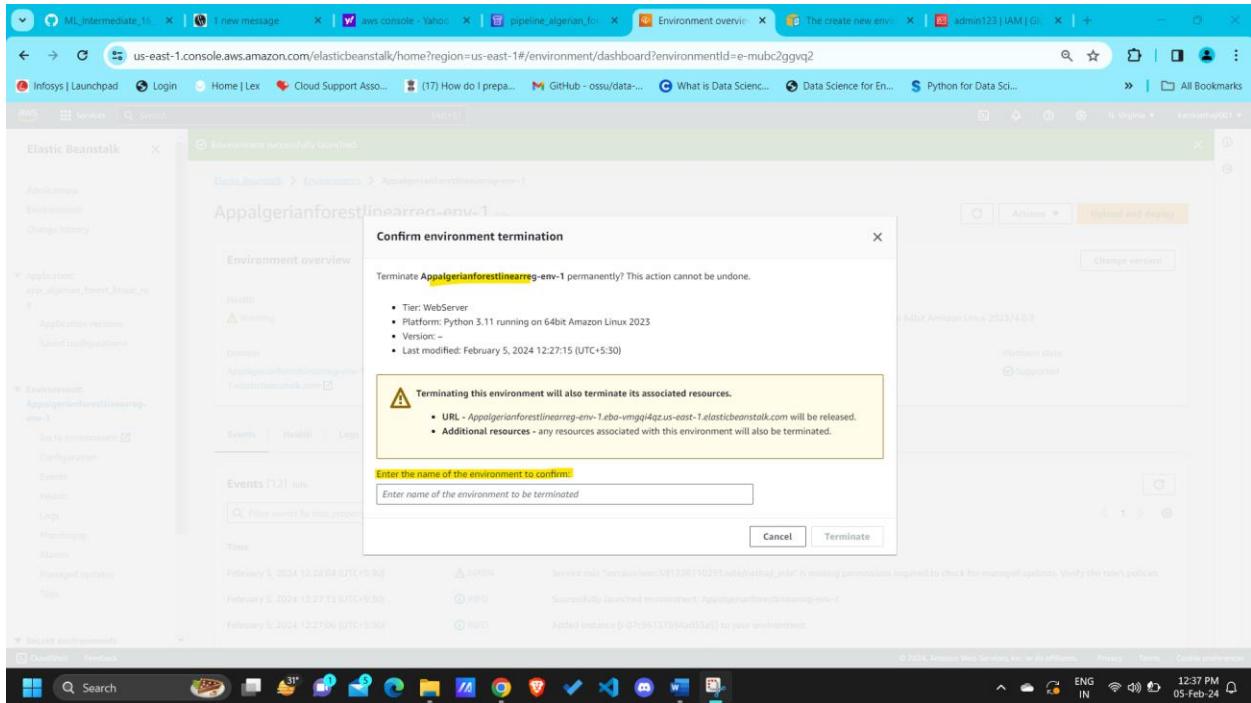
After deploying the project, **terminate the environment** using the below procedure:

Terminating the environment:

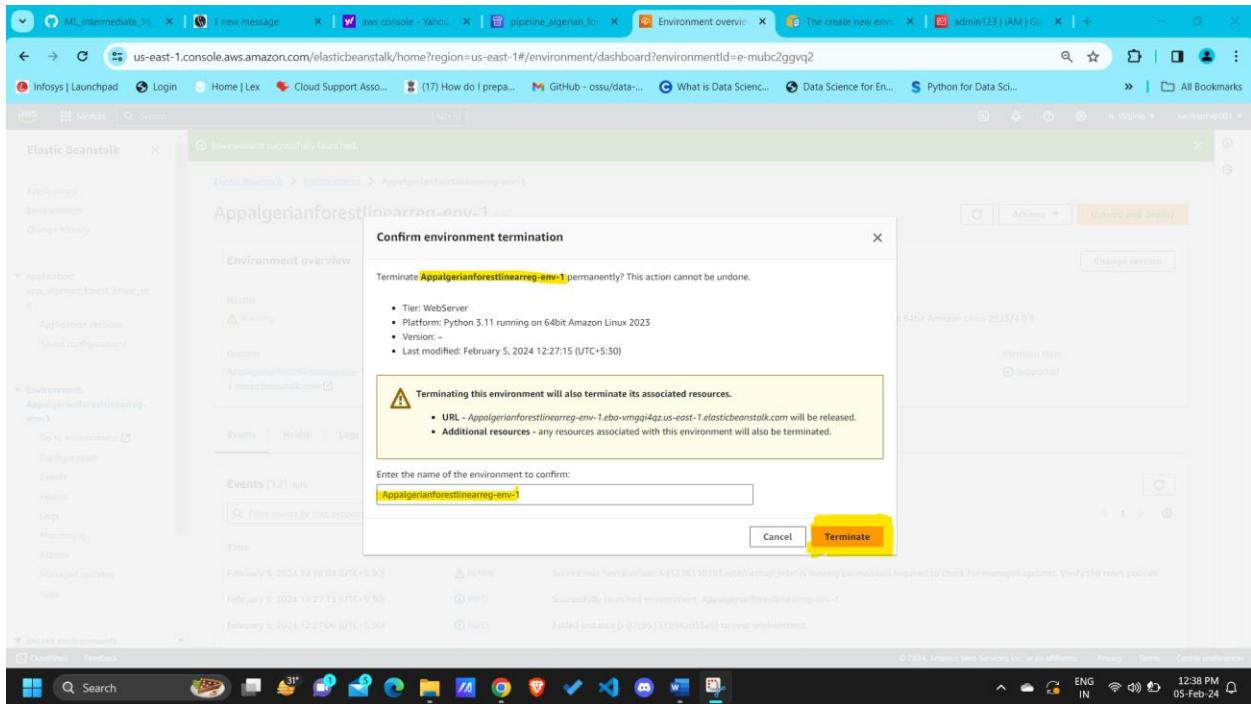


The screenshot shows the AWS Elastic Beanstalk Environment Overview page for the environment "Appalgerianforestlinearreg-env-1". The left sidebar shows the application "app_algerian_forest_linear_re" and the environment "Appalgerianforestlinearreg-env-1". The main panel displays the environment overview, including health status (Warning), environment ID (e-mubc2ggvq2), domain (Appalgerianforestlinearreg-env-1.elasticbeanstalk.com), and application name (app_algerian_forest_linear_re). On the right, the "Actions" menu is open, showing options like "Load configuration", "Save configuration", "Swap environment domain", "Clone environment", "Abort current operation", "Restart app server(s)", "Rebuild environment", and "Terminate environment". The "Terminate environment" option is highlighted with a yellow box.

→ Click on “Terminate environment”



The screenshot shows the "Confirm environment termination" dialog box. It asks if you want to terminate the environment "Appalgerianforestlinearreg-env-1" permanently. It lists the environment's details: Tier: WebServer, Platform: Python 3.11 running on 64bit Amazon Linux 2023, Version: -, Last modified: February 5, 2024 12:27:15 (UTC+5:30). A warning message states that terminating the environment will also terminate its associated resources, specifically mentioning the URL Appalgerianforestlinearreg-env-1.elasticbeanstalk.com. An input field "Enter the name of the environment to confirm:" contains the text "Appalgerianforestlinearreg-env-1". At the bottom are "Cancel" and "Terminate" buttons.



→ Click on “Terminate”

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