AISSMS

COLLEGE OF ENGINEERING

ज्ञानम् सकलजनहिताय

Approved by AICTE, New Delhi, Recognized by Government of Maharashtra
Affiliated to Savitribai Phule Pune University and recognized 2(f) and 12(B) by UGC
(Id.No. PU/PN/Engg./093 (1992)
Accredited by NAAC with "A+" Grade | NBA - 7 UG Programmes

Department of Computer Engineering

"CC Mini-project Report"

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

Submitted By

Name of the Student: Piyusha Rajendra Supe

Roll No: 23CO315

Under the Guidance of

Prof. V. M. Kanavde

ALL INDIA SHRI SHIVAJI MEMORIAL SOCIETY'S COLLEGE OF ENGINEERING

PUNE-411001

Academic Year: 2024-25(Term-II)

Savitribai Phule Pune University

AISSMS
COLLEGE OF ENGINEERING
ज्ञानम् सकलजनहिताय

Approved by AICTE, New Delhi, Recognized by Government of Maharashtra
Affiliated to Savitribai Phule Pune University and recognized 2(f) and 12(B) by UGC
(Id.No. PU/PN/Engg./093 (1992)
Accredited by NAAC with "A+" Grade | NBA - 7 UG Programmes

Department of Computer Engineering

CERTIFICATE

This is to certify that **Piyusha Rajendra Supe** from **Third Year Computer Engineering** has successfully completed her work titled "**Cloud computing Mini-project**" at AISSMS College of Engineering, Pune in the partial fulfillment of the Bachelor's Degree in Engineering.

Prof. V. M. Kanavde	**Dr. S. V. Athawale**	**Dr. D. S. Bormane**
(Faculty Guide)	(Head of Department)	(Principal)
Computer Engineering	Computer Engineering	AISSMSCOE, Pune

ACKNOWLEDGEMENT

It is with immense gratitude and respect that I take this opportunity to acknowledge the invaluable support and guidance I have received during the course of my mini project in Cloud Computing. This project has been a significant learning experience, and its successful completion would not have been possible without the constant support and encouragement of many individuals. First and foremost, I would like to express my heartfelt thanks to the entire faculty team for their unwavering support and insightful guidance throughout the project. Their valuable feedback, mentorship, and encouragement have played a crucial role in shaping the direction and quality of this work. I am sincerely grateful for the knowledge shared with me, which has significantly deepened my understanding of cloud platforms, services, and deployment models. I would also like to extend my sincere appreciation to the technical staff, colleagues, and fellow students who contributed directly or indirectly to the successful execution of this project. Their valuable inputs, discussions, and assistance were instrumental in refining my approach and overcoming challenges. Finally, I am deeply thankful to everyone who helped me gain a clearer perspective on Cloud Computing concepts, inspired innovative thinking, and motivated me to keep learning and improving. This project has not only enhanced my technical skills but also enriched my overall learning experience. I remain grateful to all who played a part in making this endeavour a success

Academic Year: 2024-2025

Piyusha Rajendra Supe (23CO315)

TABLE OF CONTENTS

ABSTRACT

This project focuses on developing and deploying a machine learning-based web application for predicting house prices using AWS Elastic Beanstalk. The goal is to create an end-to-end solution where users can input property features through a web interface and receive an estimated house price generated by a trained machine learning model. The project uses Python for backend development, Flask for the web framework, and HTML/CSS for the frontend interface. The machine learning component is built using the Scikit-learn library, with a Random Forest Regressor as the final predictive model. The dataset used for model training includes various features that influence house prices, such as the number of rooms, square footage, location details, number of bathrooms, and lot size. The data was cleaned and preprocessed through steps including handling missing values, encoding categorical variables, and scaling numerical features. Feature selection was performed to identify the most impactful variables.

Several machine learning algorithms were tested, including Linear Regression, Decision Trees, and Random Forest. Based on performance metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared (R^2), the Random Forest Regressor showed the best predictive accuracy and was chosen for deployment. The model was serialized using joblib to enable integration into a web application. A lightweight web application was created using Flask. The frontend allows users to input property details via a simple HTML form. These inputs are sent to the backend, where the trained model processes them and returns a price prediction, which is then displayed to the user. The interface is clean and user-friendly, designed with basic HTML and CSS.

To make the application accessible online and ensure scalability, it was deployed on AWS using Elastic Beanstalk. Elastic Beanstalk simplifies cloud deployment by automating server setup, application hosting, and scaling. The deployment involved preparing the project directory with a requirements.txt and a Procfile, then uploading it to the AWS Elastic Beanstalk environment via the AWS Management Console. This deployment allows the application to be available to users over the internet with minimal manual infrastructure management. It demonstrates how a machine learning model can be transformed into a usable tool that delivers real-time predictions to end-users. The use of AWS Elastic Beanstalk provides an efficient way to deploy applications without deep cloud expertise, making it ideal for small to medium-scale ML projects.

In summary, this project combines machine learning, web development, and cloud deployment into a functional, real-world application. It showcases the practical application of Random Forest in regression problems and highlights the accessibility of ML-powered solutions when paired with cloud services. Future improvements could include enhancing the user interface, adding data visualization, or incorporating more complex models.

INTRODUCTION

Machine learning (ML) is transforming industries by enabling intelligent, data-driven solutions for complex problems. In the real estate sector, predicting house prices is a classic application of ML that offers practical value to buyers, sellers, agents, and developers. An accurate prediction system can assist users in evaluating property value, budgeting, and making more informed decisions. This project focuses on developing a machine learning-based web application that predicts house prices using user-input features and deploying it on the cloud using Amazon Web Services (AWS) Elastic Beanstalk.

The core of the application is a machine learning model developed in Python using the Scikit-learn library. The dataset used includes common housing features such as number of bedrooms and bathrooms, total square footage, lot size, and other structural and locational details. After preprocessing the data — including handling missing values, encoding categorical variables, and scaling numerical values — several regression algorithms were tested. The Random Forest Regressor was selected for its strong performance and robustness in handling non-linear data relationships. The trained model is then serialized using joblib for easy reuse in the web application.

To make the model accessible to users through an interactive interface, a web application was built using Flask, a lightweight Python web framework. The frontend is designed using HTML and CSS and allows users to input housing features through a simple form. Once submitted, the Flask backend processes the inputs and uses the trained model to return an estimated price, which is displayed on the page.

For deployment, the application is hosted using **Amazon Elastic Beanstalk**, a Platform-as-a-Service (PaaS) offering from AWS that simplifies the deployment and scaling of web applications. Elastic Beanstalk supports multiple programming languages and frameworks, including Python and Flask. It automatically handles infrastructure provisioning, load balancing, environment monitoring, and scaling, allowing developers to focus on writing code rather than managing servers. With Elastic Beanstalk, the application is deployed using a requirements.txt file and a Procfile, and is accessible via a public URL with minimal setup.

This project demonstrates how machine learning, web development, and cloud computing can be integrated to build a complete end-to-end solution. It provides a practical tool for predicting house prices and serves as a foundation for future enhancements such as real-time data integration, better UI design, or more advanced predictive models.

PROBLEM STATEMENT

Title: Deploying a machine learning application of House price prediction on Cloud using the AWS Elastic Beanstalk

Description:

- While machine learning models are commonly built in local environments, deploying them for real-time, public use remains a major challenge, especially for beginners or small teams.
- Traditional deployment methods involve manual infrastructure setup, server management, and scaling — tasks that require significant cloud expertise.
- There is a need for a simplified, cloud-based approach to deploy machine learning applications that ensures high availability, scalability, and ease of access.
- Most students and developers struggle to bridge the gap between ML development and real-world deployment due to the complexity of managing cloud infrastructure.

AWS Elastic Beanstalk, a Platform-as-a-Service (PaaS), offers an ideal solution by automating the deployment, scaling, and management of applications without needing to handle low-level cloud configurations.

This project addresses the problem by:

- Building a house price prediction model using machine learning.
- Creating a user-friendly web interface using Flask.
- Deploying the complete application on AWS Elastic Beanstalk to demonstrate how cloud platforms can simplify ML deployment.

The focus is on showcasing how cloud services can make machine learning applications production-ready with minimal setup and maximum scalability.

OVERVIEW

This mini-project demonstrates the practical use of cloud computing—specifically AWS Elastic Beanstalk—for deploying a machine learning-powered web application that predicts house prices. The focus is on using cloud services to simplify deployment, increase accessibility, and ensure scalability without requiring manual infrastructure management.

The project is structured into three main components:

- **Machine Learning Model**
  - Built using Python and Scikit-learn.
  - Trained on housing data containing features like square footage, number of bedrooms/bathrooms, etc.
  - The selected model is a Random Forest Regressor, chosen for its accuracy and robustness.
  - The model is serialized using joblib for easy integration with the web app.
- **Web Application (Frontend + Backend)**
  - Developed using the Flask framework in Python.
  - HTML and CSS are used for building the user interface.
  - Users input property details via a form; the backend processes the input and returns a price prediction.
- **Cloud Deployment using AWS Elastic Beanstalk**
  - **Elastic Beanstalk** is a **Platform-as-a-Service (PaaS)** that automates the deployment and management of applications.
  - Supports multiple languages and frameworks, including Python and Flask.
  - Automatically handles infrastructure provisioning, load balancing, scaling, and application health monitoring.
  - Simplifies the deployment process to just uploading code and configuration files.
  - Requires minimal setup: only a requirements.txt for dependencies and a Procfile to specify the application entry point.

Key Benefits of Using AWS Elastic Beanstalk in This Project:

- No need to manually manage EC2 instances, load balancers, or network settings.
- Easy scaling options based on traffic demand.
- Integrated monitoring and logging via AWS Cloud Watch.
- Faster deployment cycles compared to traditional methods.
- Ideal for students and developers who want cloud capabilities without deep DevOps knowledge.

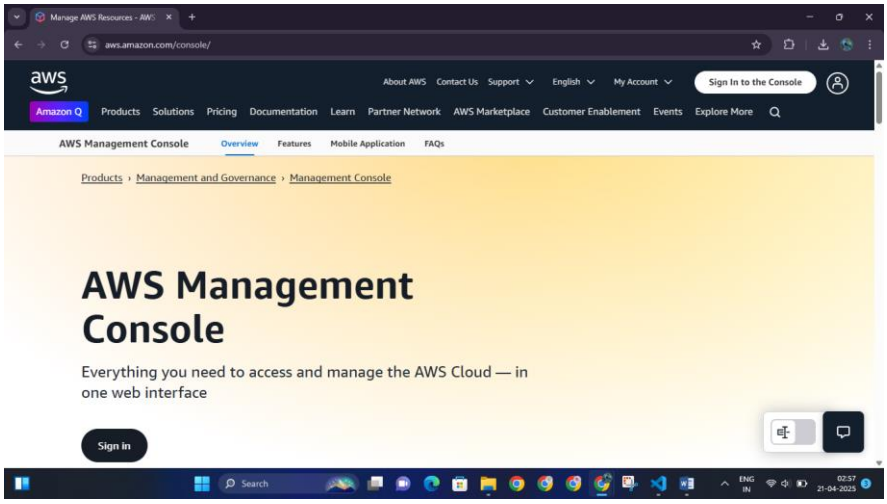
Additional AWS Services Involved (Optional/Extendable):

- **Amazon S3** for storing static assets or model files.
- **IAM roles** for secure access control.
- **Cloud Watch** for application monitoring and log tracking.

IMPLEMENTATION AND METHODOLOGY

Step 1: Sign Up and Log in to AWS

1. **Go to AWS**:
   Open your browser and go to AWS.
2. **Create an account**:
   - Click **Sign Up** and follow the prompts to create a new account. You'll need to provide billing information (AWS offers a **Free Tier** for new users).
   - **Sign in** once your account is set up.



Step 2: Prepare Your Flask App

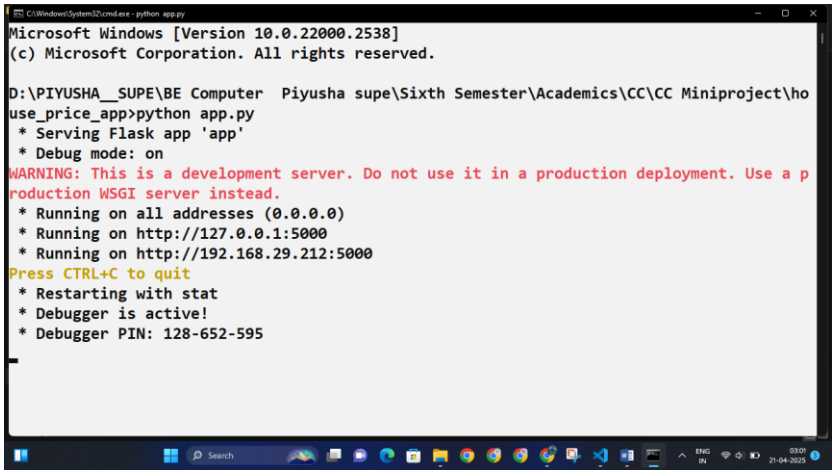
1. **Get your Flask app ready**: (**Testing the application locally**)
   - You should already have your Flask app ready (the one with the machine learning model). Make sure your app is working locally on your computer first.
   - You need to prepare two files for AWS:
     - requirements.txt: This file tells AWS what libraries your app needs. To create it, run the following in your project folder:

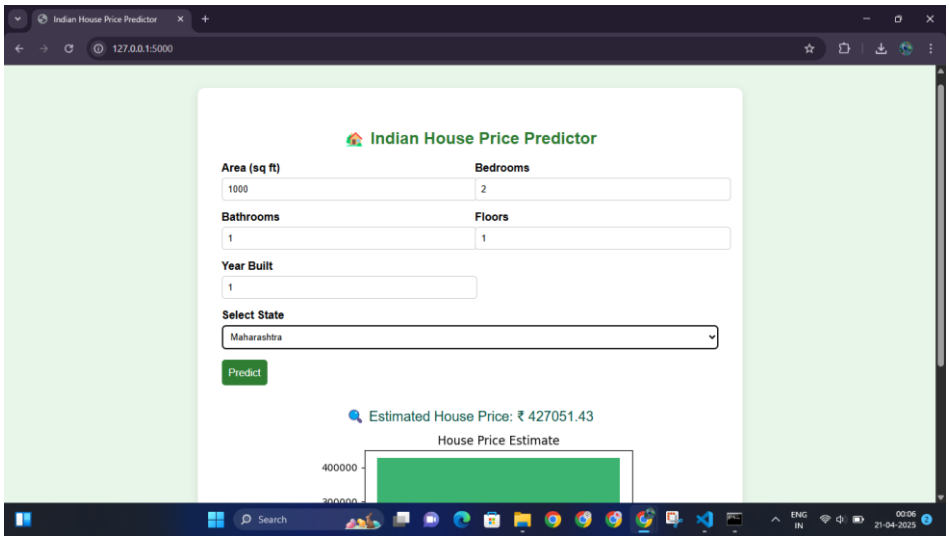
       pip freeze > requirements.txt

     - Procfile: This file tells Elastic Beanstalk how to run your app. Create a new file called Procfile (no extension) in your project folder, and add this line:

       web: python app.py

       Make sure your app.py file is the main script that runs your Flask app.



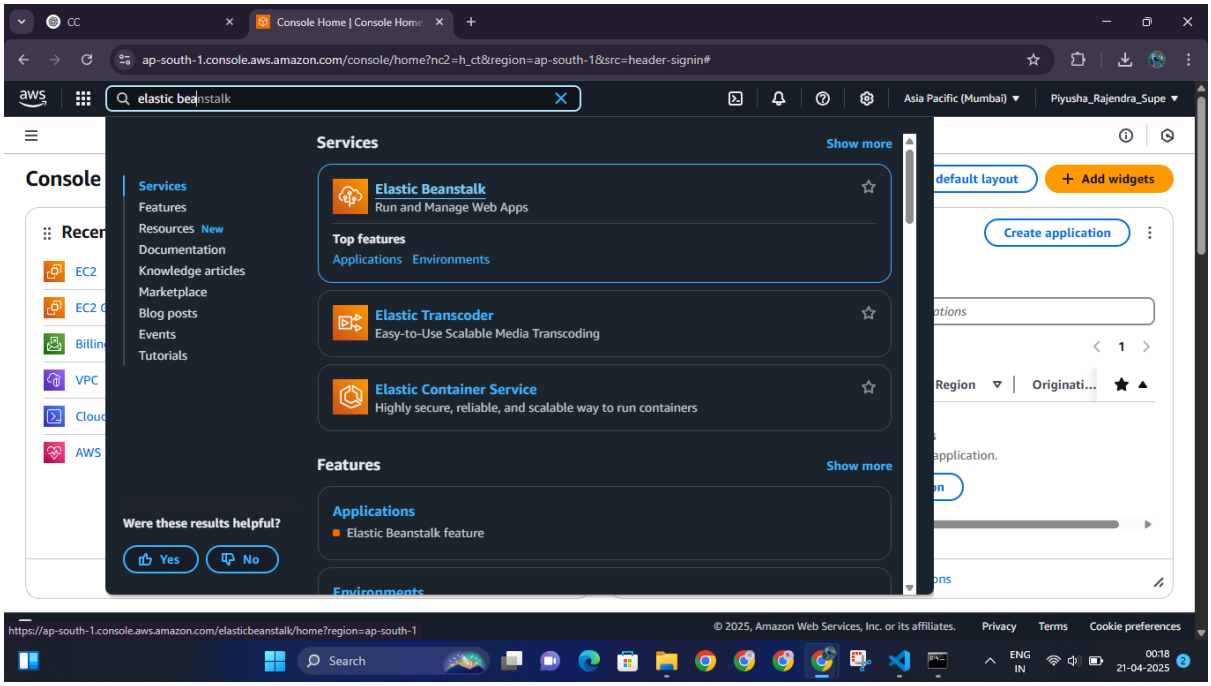


2. **Update** **app.py**:
   Make sure your Flask app listens on port 8080, which is the default for AWS:

```
if __name__ == "__main__":
        app.run (debug=True, host='0.0.0.0', port=8080)
```

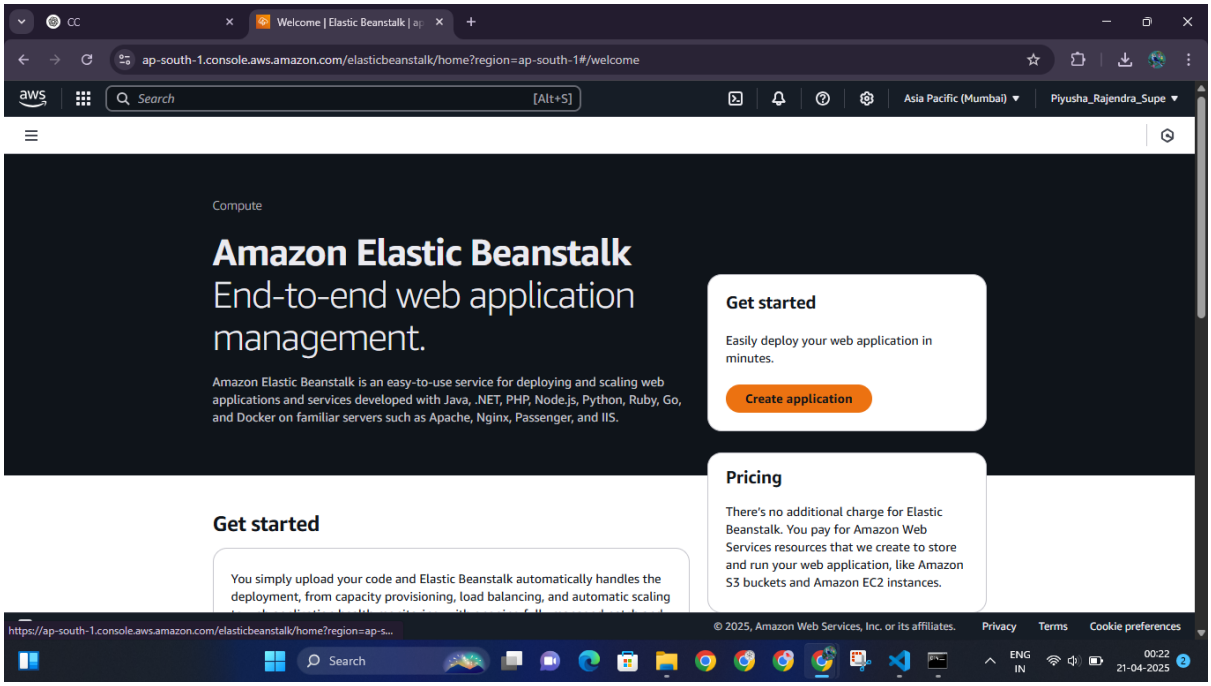
Step 3: Access AWS Console

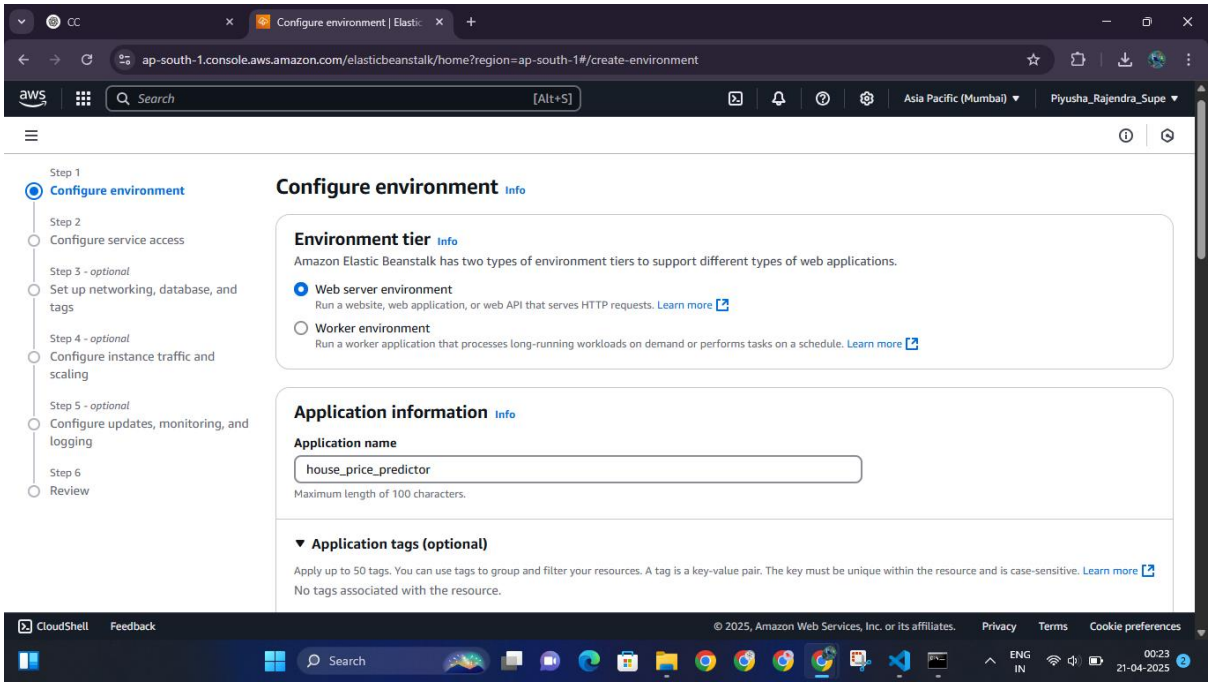
1. **Go to AWS Management Console**:
   - Open your browser and go to AWS Management Console.
   - **Sign in** with the AWS account you created.

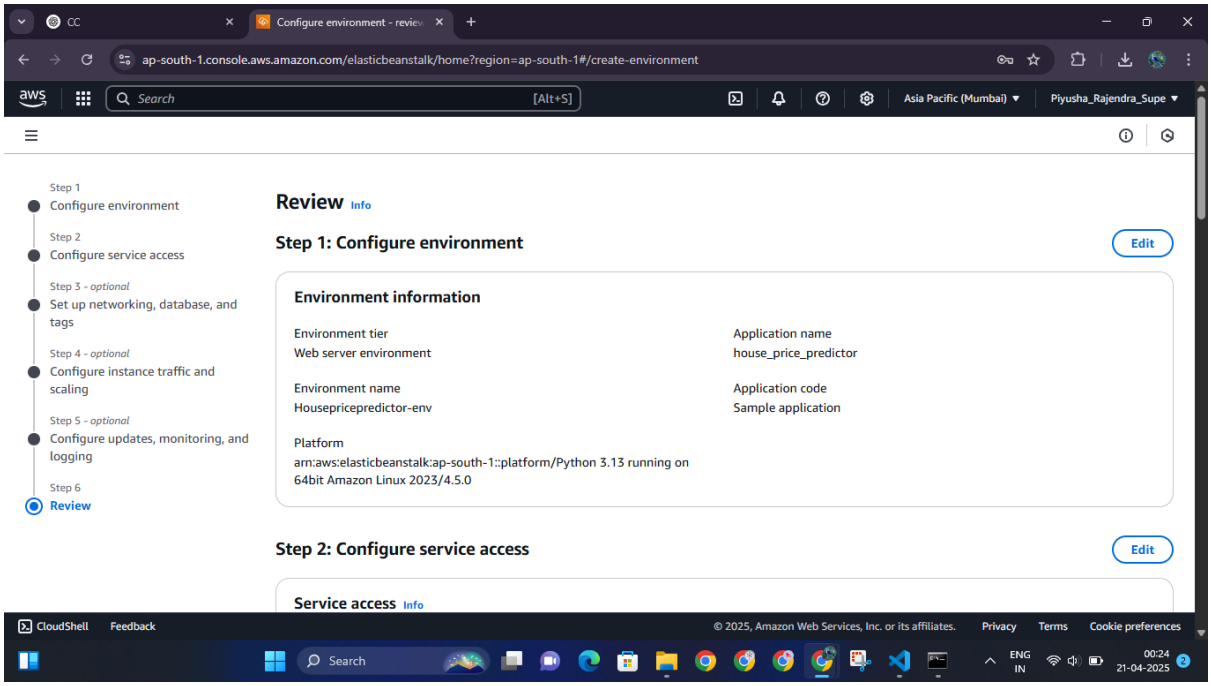


Step 4: Create an Elastic Beanstalk Application

1. **Search for Elastic Beanstalk**:
   - In the **Search bar** at the top, type **Elastic Beanstalk** and click on it.
2. **Create a new application**:
   - In the Elastic Beanstalk console, click **Create Application**.
   - **Give your app a name**: You can name it something like house-price-predictor.
   - **Platform**: Choose **Python**.
   - **Environment Tier**: Select **Web Server Environment**.
   - Click **Create Application**.



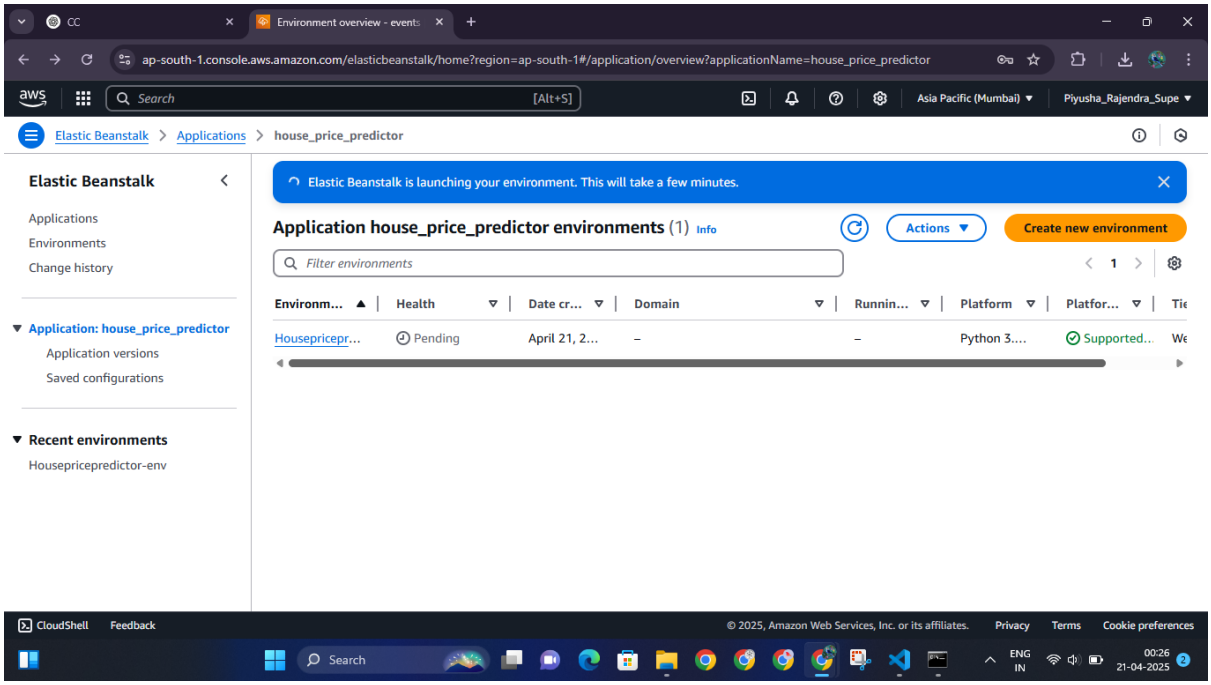


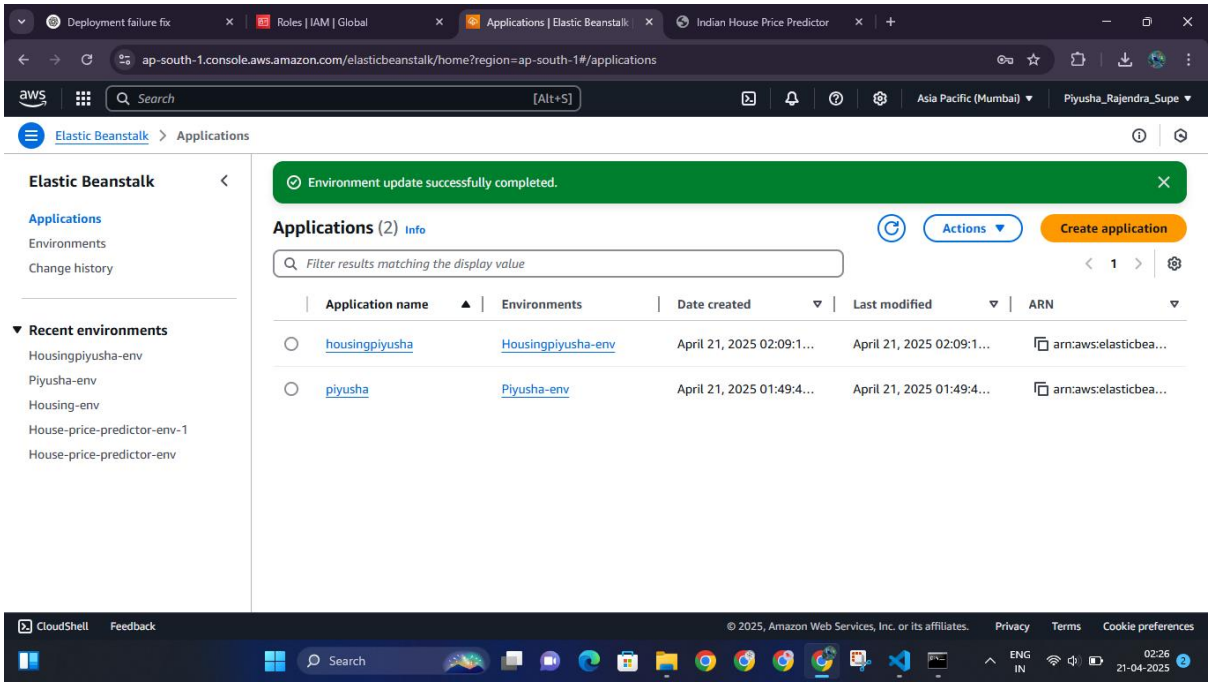


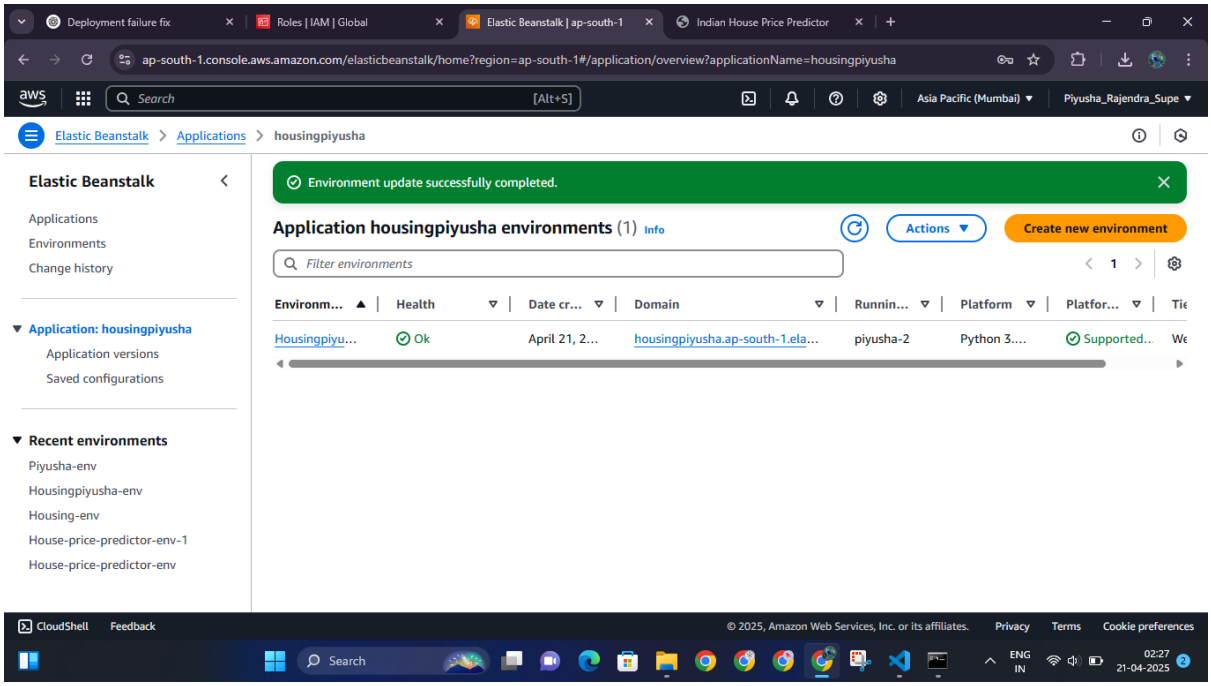
Step 5: Create an Elastic Beanstalk Environment

1. **Click on your newly created application**:
   - After the app is created, click on it from the list to go to the app dashboard.
2. **Create a new environment**:

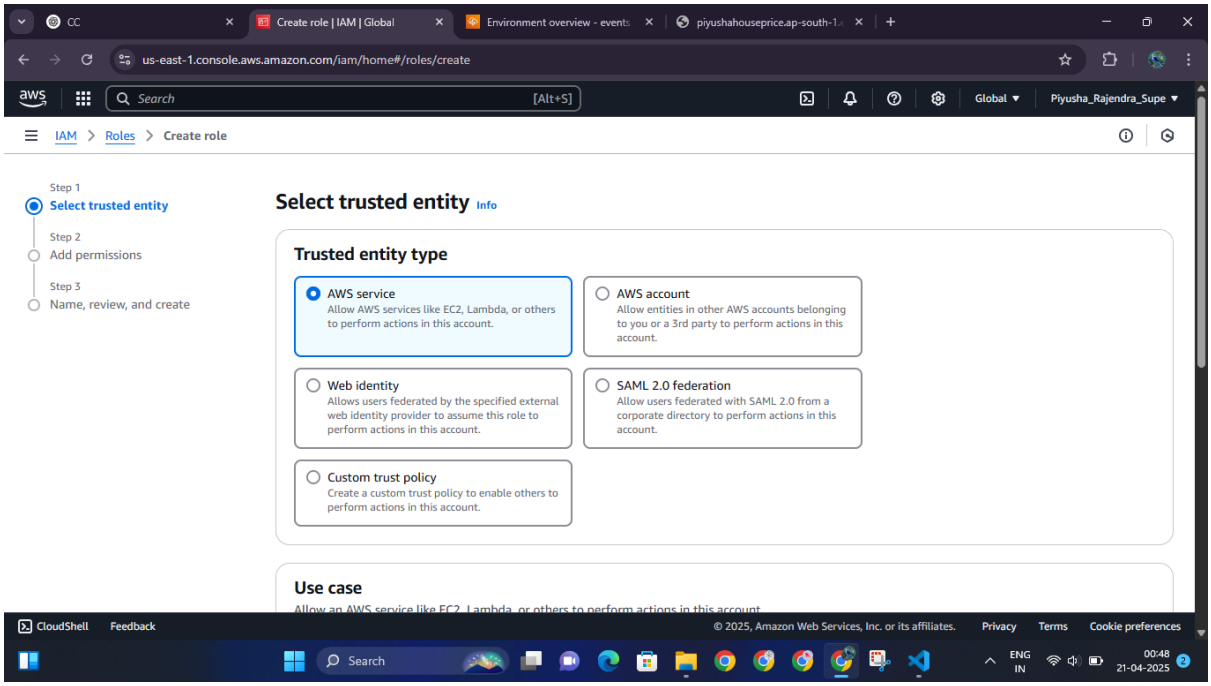
- Click on the **Create Environment** button.
- **Choose environment type**: Select **Web Server Environment**.
- **DNS name**: Enter a name for your app's URL (e.g., house-price-predictor).
- **Platform**: Select **Python**.
- Click **Next**.

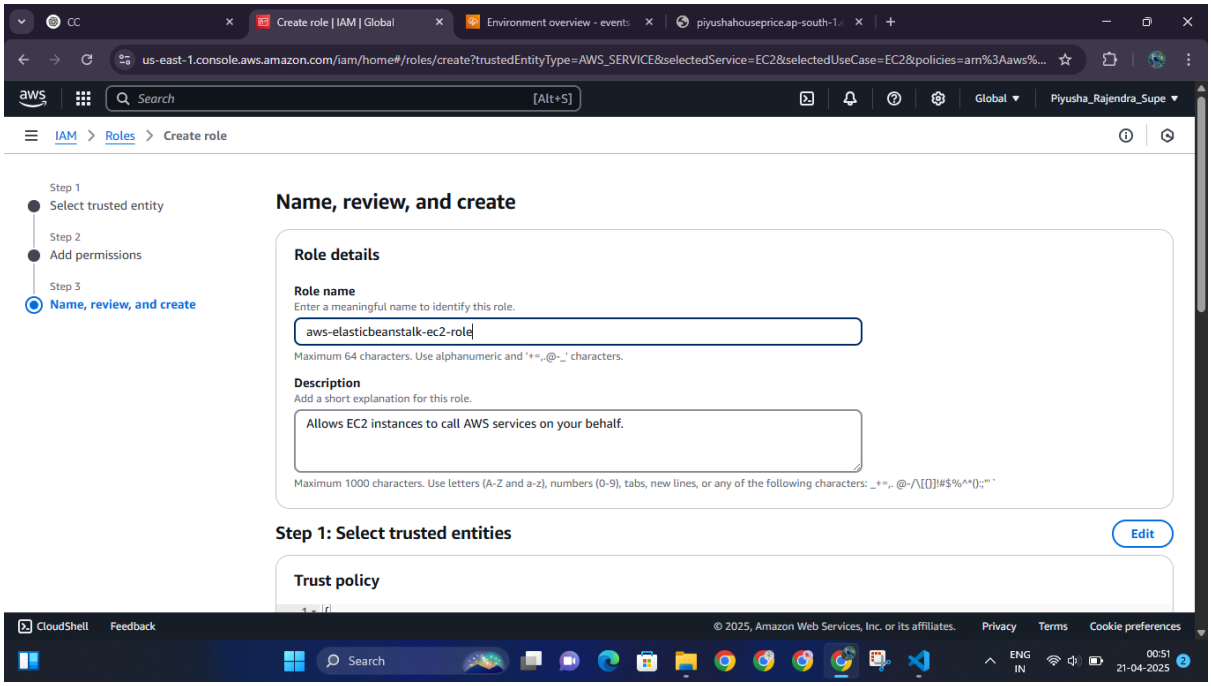


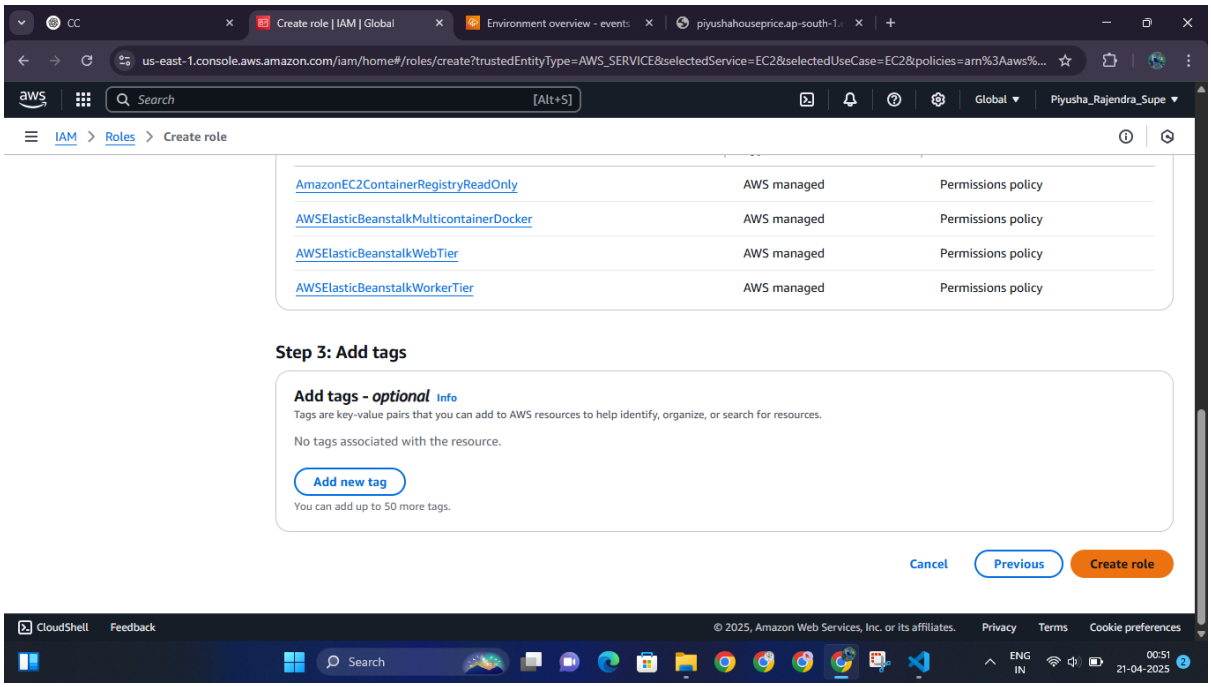


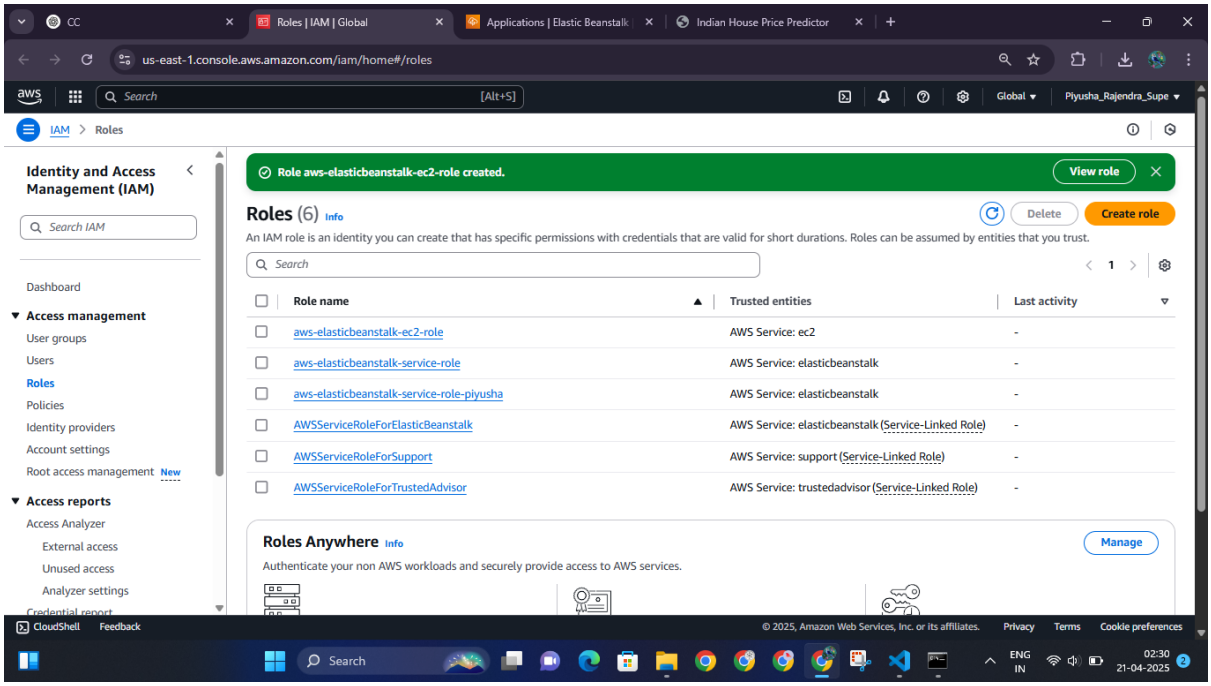


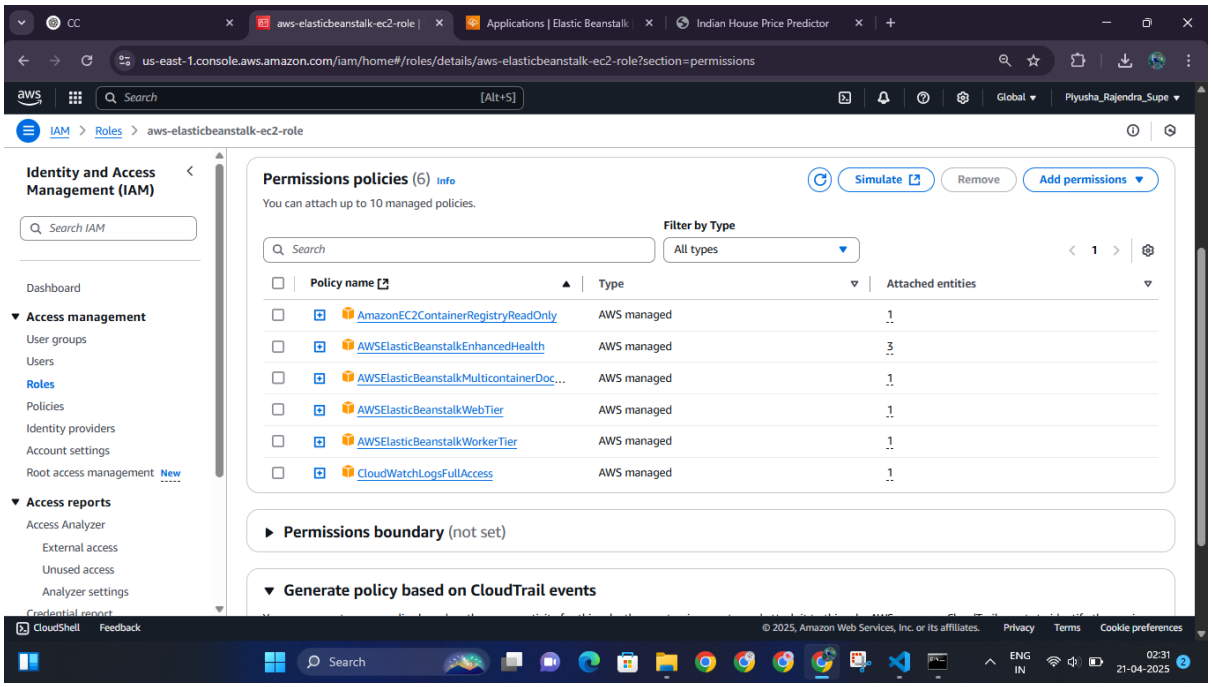
Make sure the following roles are created and the permissions are set to them:




Create role | IAM | Global
Environment overview - events
us-east-1.console.aws.amazon.com/iam/home#/roles/create?trustedEntityType=AWS_SERVICE&selectedService=EC2&selectedUseCase=EC2&policies=arn%3Aaws%...
Search [Alt+S]
Global
Piyusha_Rajendra_Supe
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
aws-elasticbeanstalk-ec2-role
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{}]!#$%^*();"'`
Step 1: Select trusted entities
Edit
Trust policy
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ENG
IN
00:51
21-04-2025

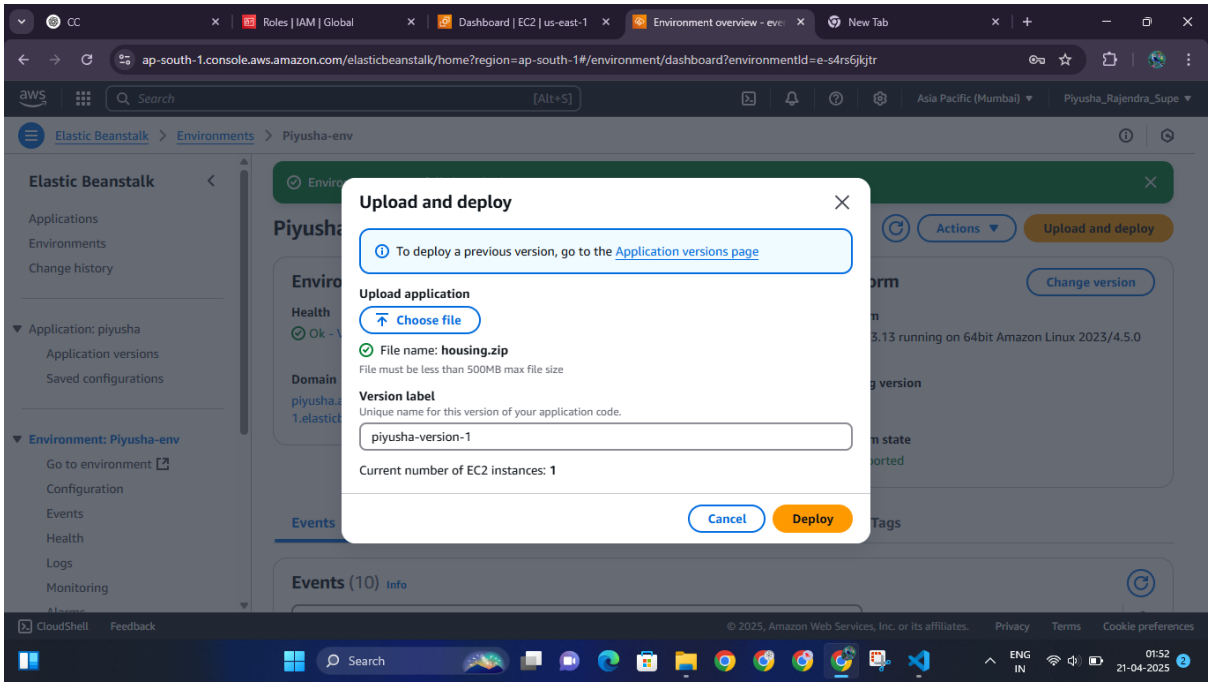

Create role | IAM | Global
Environment overview - events
us-east-1.console.aws.amazon.com/iam/home#/roles/create?trustedEntityType=AWS_SERVICE&selectedService=EC2&selectedUseCase=EC2&policies=arn%3Aaws%...
Search [Alt+S]
Global
Piyusha_Rajendra_Supe
IAM > Roles > Create role
AmazonEC2ContainerRegistryReadOnly AWS managed Permissions policy
AWSElasticBeanstalkMulticontainerDocker AWS managed Permissions policy
AWSElasticBeanstalkWebTier AWS managed Permissions policy
AWSElasticBeanstalkWorkerTier AWS managed Permissions policy
Step 3: Add tags
Add tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.
Cancel
Previous
Create role
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ENG
IN
00:51
21-04-2025





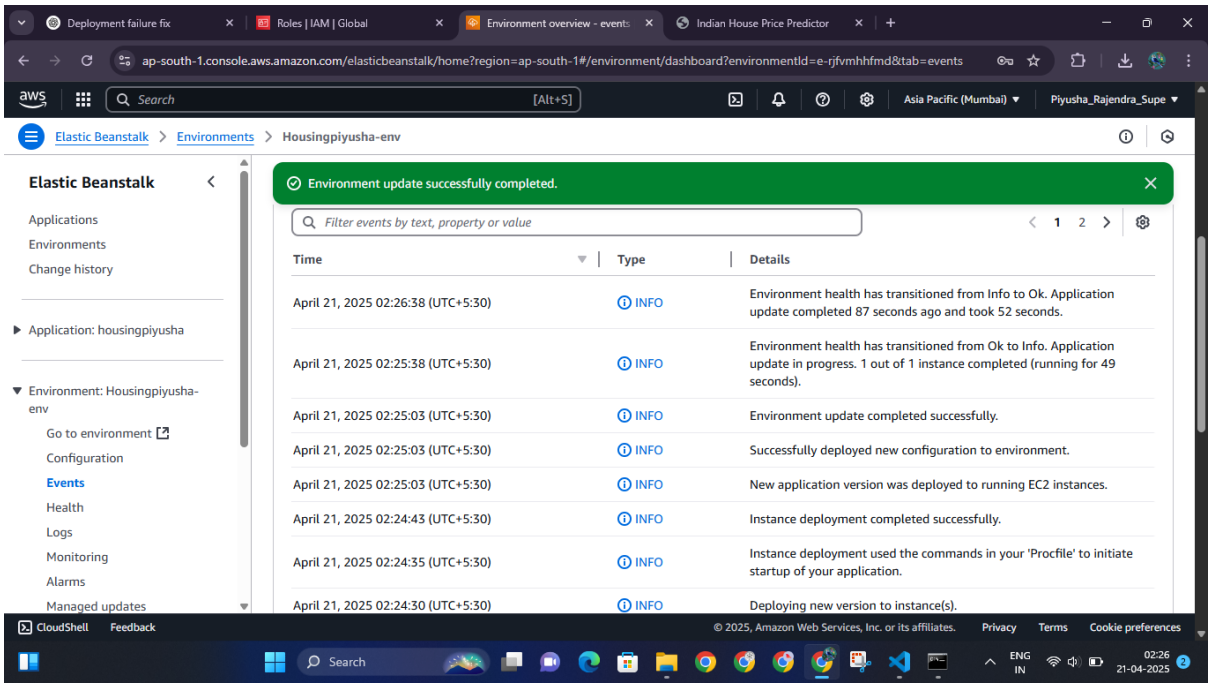
Step 6: Upload Your App

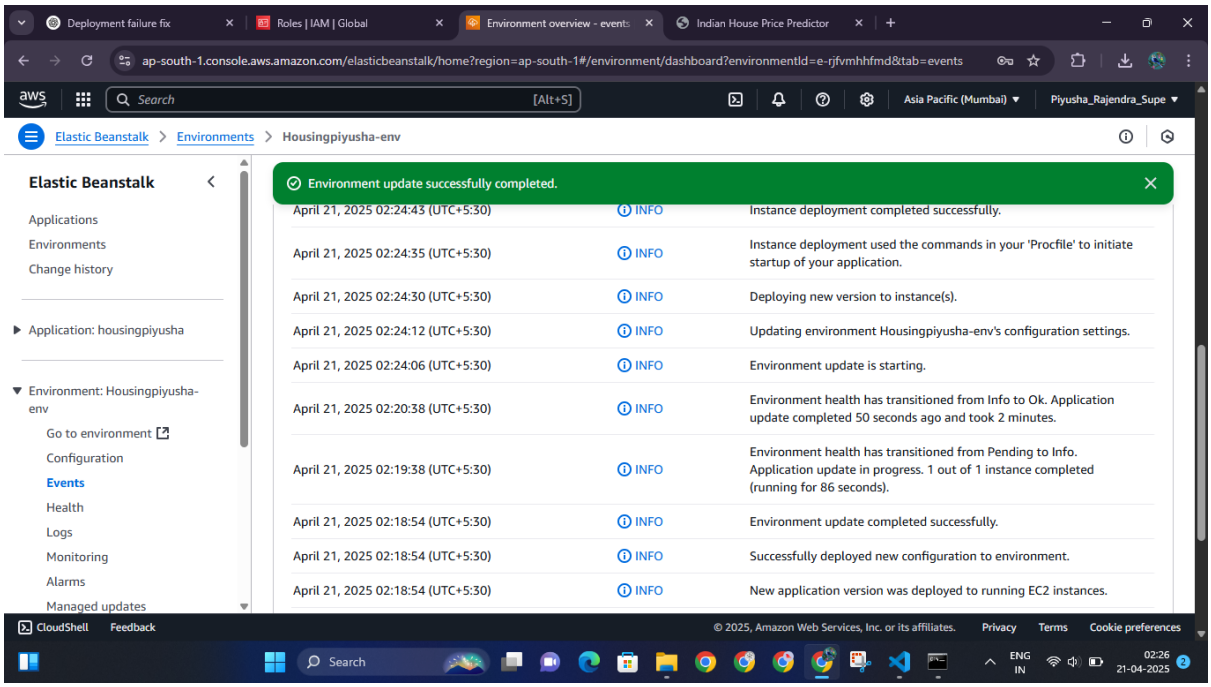
1. **Prepare your app ZIP file**:
   - Go to your project folder and **ZIP** all the files in it, including app.py, requirements.txt, and Procfile.
2. **Upload the ZIP file**:
   - Under **Application Version**, click **Upload your code**.
   - Click the **Choose file** button and select the ZIP file you just created.
   - After uploading, click **Next**.

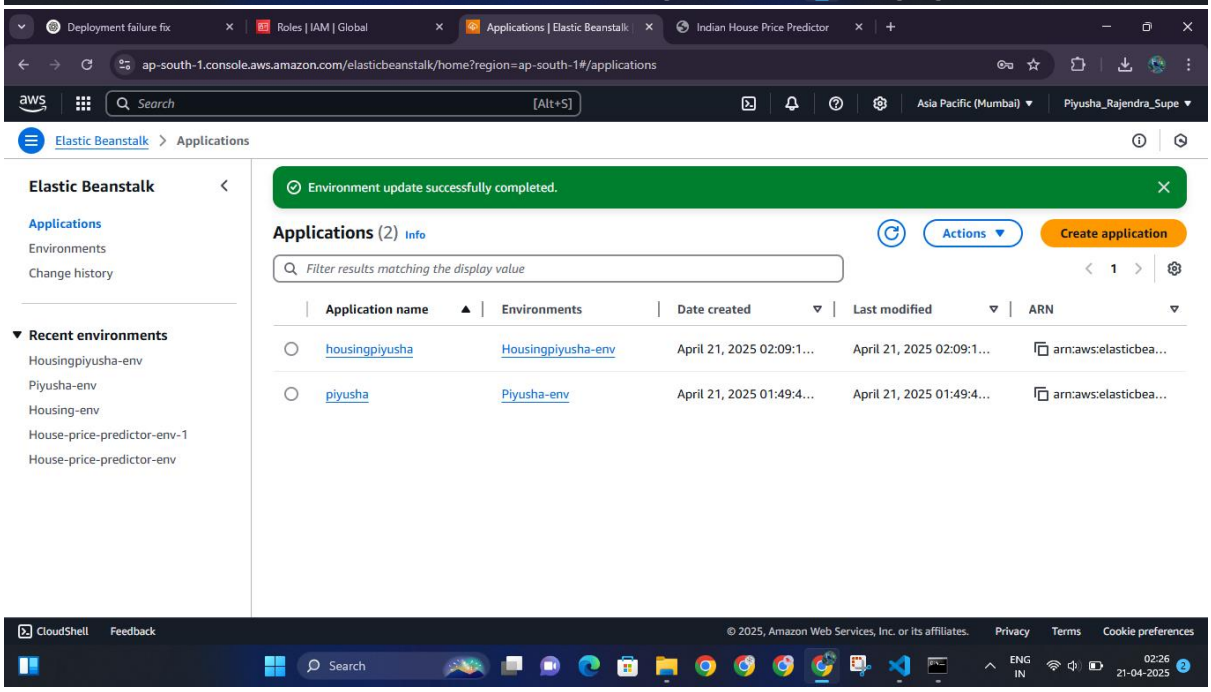


Step 7: Review and Launch the Application

1. **Review your environment**:
   - AWS will show a review of your environment settings (like DNS name, platform, etc.).
   - Check everything, then click **Launch**.
2. **Wait for Elastic Beanstalk to set up**:
   - AWS will now take a few minutes to provision your environment and deploy your app.
   - You will see a **status page** indicating when the environment is ready.

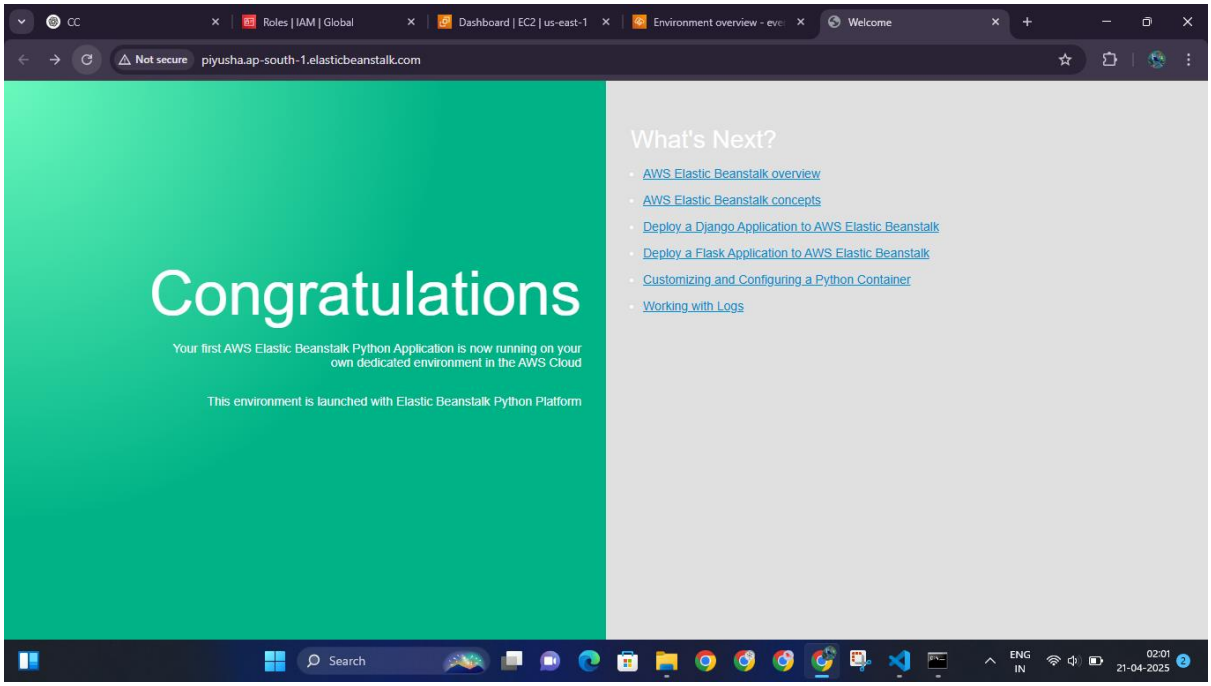


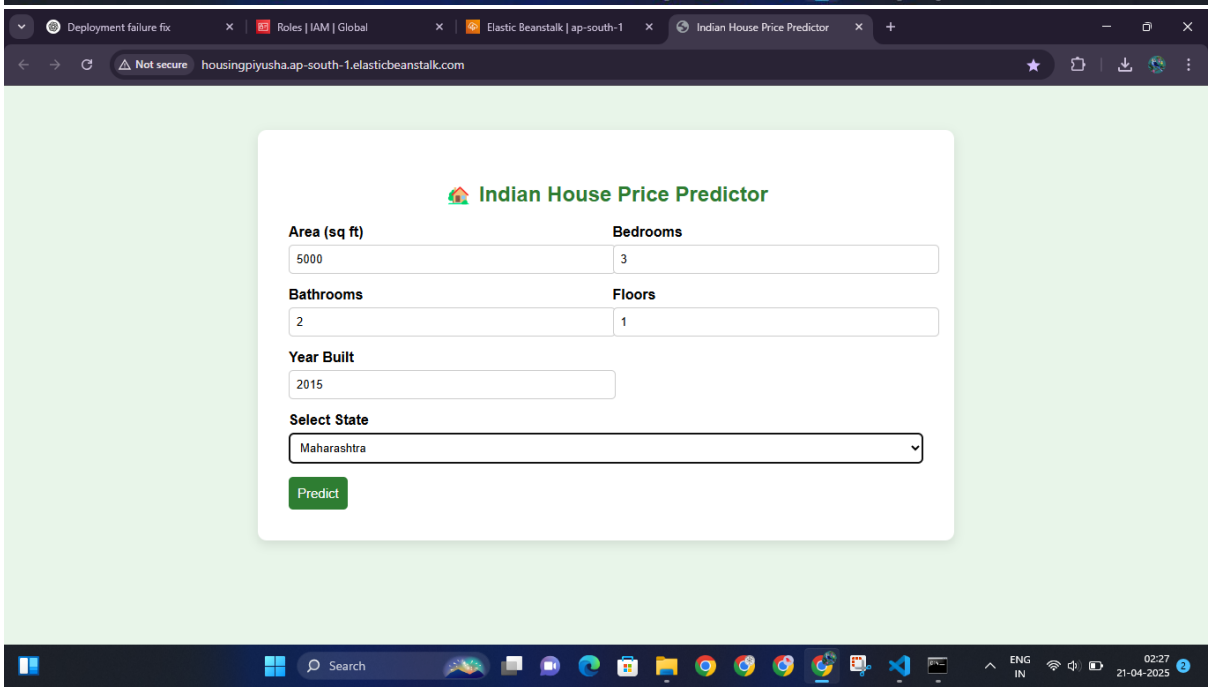


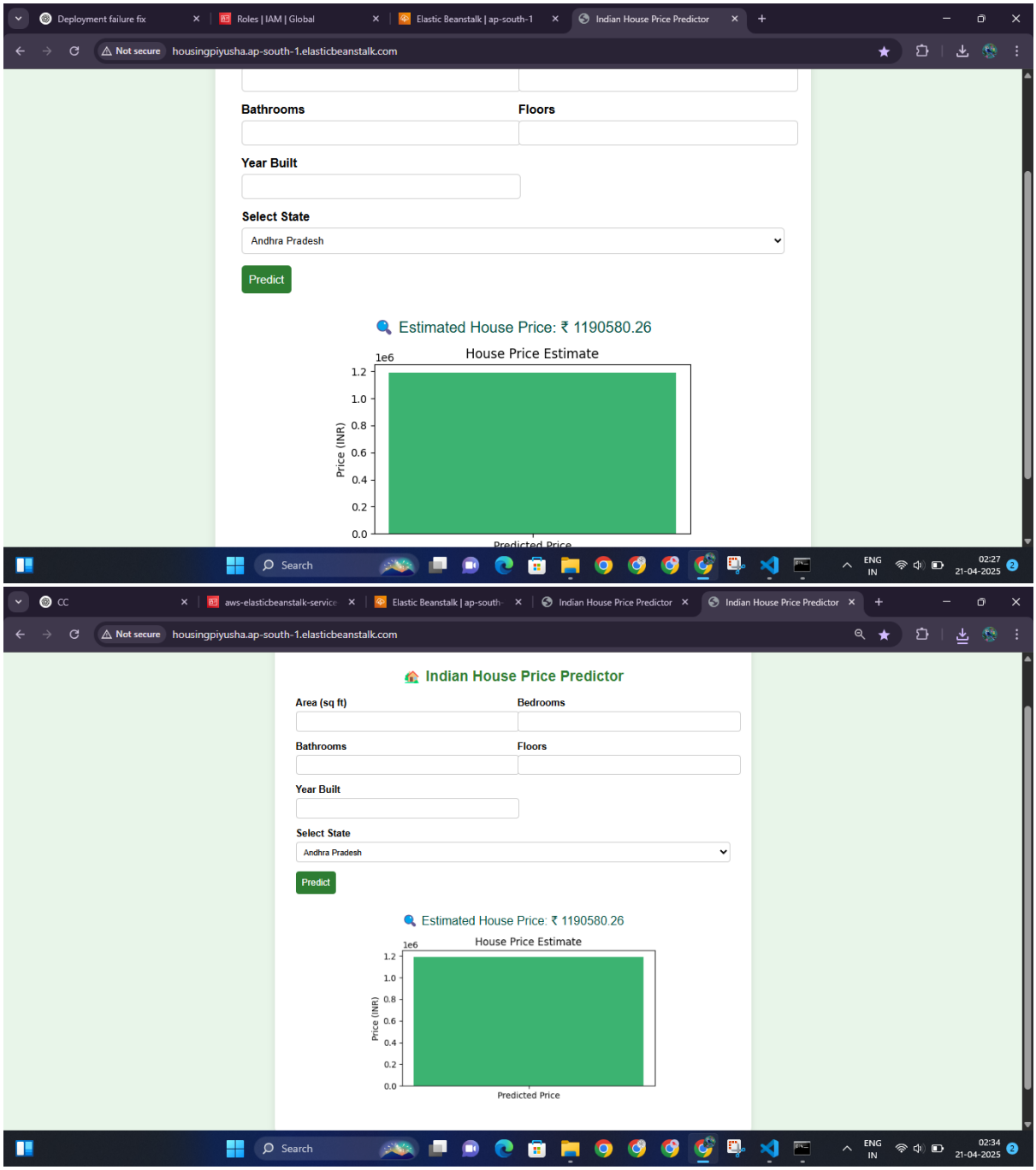


Step 8: Access Your Flask App

1. **Get your app URL**:
   - Once the environment is created, you'll see a **URL** for your app in the Elastic Beanstalk dashboard (e.g., http://house-price-predictor.elasticbeanstalk.com).
   - Click on this link, and your Flask app will open in a new tab!

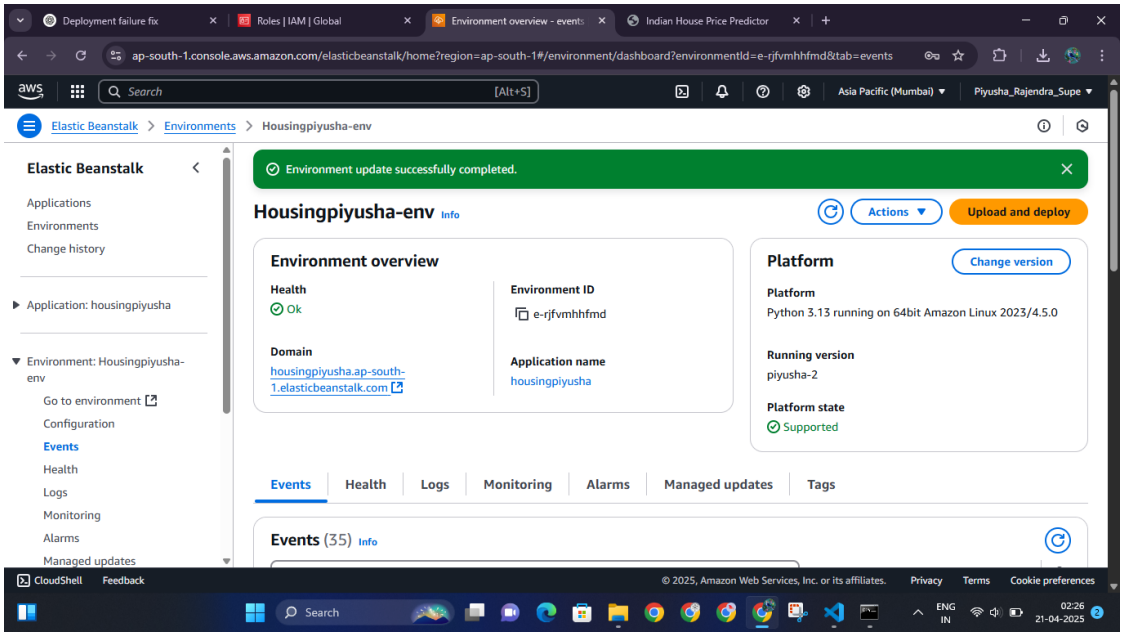

Not secure
piyusha.ap-south-1.elasticbeanstalk.com
Congratulations
Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud
This environment is launched with Elastic Beanstalk Python Platform
What's Next?
AWS Elastic Beanstalk overview
AWS Elastic Beanstalk concepts
Deploy a Django Application to AWS Elastic Beanstalk
Deploy a Flask Application to AWS Elastic Beanstalk
Customizing and Configuring a Python Container
Working with Logs


Not secure
housingpiyusha.ap-south-1.elasticbeanstalk.com
Indian House Price Predictor
Area (sq ft)
5000
Bedrooms
3
Bathrooms
2
Floors
1
Year Built
2015
Select State
Maharashtra
Predict



Step 9: Monitor Your App

1. **Check the health of your app**:
   - Go back to the **Elastic Beanstalk** dashboard.
   - You can see whether your app is healthy (Green), or if there are any issues (Red).
2. **View logs**:
   - If your app is not working, you can check the **logs** from the Elastic Beanstalk console.
   - Click **Logs** > **Request Logs** to get detailed logs that can help troubleshoot errors.



Step 10: Clean Up (Optional)

Once you're done testing or if you want to stop the app, you can **terminate the environment**:

1. **Terminate the environment**:
    - In the Elastic Beanstalk console, go to your environment.
    - Click **Actions** > **Terminate Environment**.

Confirm the termination to stop any resources and prevent charges.

FUNCTIONALITY AND ADVANTAGE

Functionality of the Application

- Accepts user input for house features (e.g., number of rooms, area, bathrooms).
- Processes input through a trained Random Forest machine learning model.
- Predicts house prices based on the provided features.
- Provides real-time output via a web interface.
- Frontend built using HTML and CSS for a simple, user-friendly experience.
- Flask backend handles data input, model integration, and prediction logic.
- Fully deployed and accessible online using AWS Elastic Beanstalk.

Advantages of the Project (Cloud-Focused)

- **Simplified Deployment:** AWS Elastic Beanstalk manages server provisioning and environment setup, streamlining the deployment process.
- **Public Accessibility:** The application is hosted online and can be accessed from any web-enabled device.
- **Scalability:** Elastic Beanstalk automatically scales the application based on incoming traffic and resource needs.
- **No Manual Infrastructure Management:** Eliminates the need to configure EC2 instances, networking, or load balancers manually.
- **Secure and Configurable:** Can be extended with IAM roles, security groups, and HTTPS for secure data handling.
- **Monitoring and Logging:** Integrated with AWS Cloud Watch for real-time monitoring and application log tracking.
- **Time-Efficient:** Greatly reduces the time and effort required to deploy a machine learning application to the cloud.
- **Educational Value:** Provides hands-on experience with cloud deployment, aligning with cloud computing course objectives

CONCLUSION

This project demonstrates the practical application of cloud computing concepts through the deployment of a web application using **AWS Elastic Beanstalk**. By leveraging Elastic Beanstalk's Platform-as-a-Service (PaaS) model, the project showcases how developers can easily deploy, manage, and scale web applications without the need for in-depth knowledge of cloud infrastructure. Elastic Beanstalk simplifies the complexities traditionally associated with cloud deployment—such as provisioning servers, configuring load balancers, and managing auto-scaling. With minimal configuration, the application was successfully deployed and made publicly accessible, illustrating how AWS services can reduce deployment time and operational overhead. The use of Elastic Beanstalk also provides built-in support for monitoring, logging, and scaling, which are essential components of any production-grade cloud application. This highlights the advantages of cloud-native platforms in enabling high availability, reliability, and maintainability of services with minimal manual effort.

Overall, the project reinforces key cloud computing principles such as scalability, automation, resource abstraction, and ease of deployment. It serves as a practical example of how modern cloud platforms like AWS can empower developers and students to build and deploy real-world applications quickly and efficiently, aligning perfectly with the core objectives of cloud computing education.

REFERENCES

- Amazon Web Services. *What is AWS Elastic Beanstalk?* Retrieved from https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html
- Amazon Web Services. *Deploying a Flask Application to AWS Elastic Beanstalk*. Retrieved from https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html
- Amazon Web Services. *Elastic Beanstalk Developer Guide*. Retrieved from https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3.html
- Beanstalk Tutorials. *Deploying Python Applications on AWS Elastic Beanstalk*. Retrieved from https://realpython.com/flask-by-example-part-1-project-setup/
- Kratzke, N., & Quint, P.-C. (2017). *Understanding Cloud-native Applications after 10 Years of Cloud Computing – A Systematic Mapping Study*. *Journal of Systems and Software*, 126, 1–16. https://doi.org/10.1016/j.jss.2017.01.001