Harnessing EC2,Vpc & Mobaxterm for cutting-Edge Entity Hosting

A PROJECT REPORT

Submitted by

POSHIKA B	**113322243075**
RESHNI U	**113322243084**
RAMYA Y	**113322243082**
SANTHOSHINI S	**113322243094**

BACHELOR OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

VELAMMAL INSTITUTE OF TECHNOLOGY

CHENNAI 601 204

ANNA UNIVERSITY: CHENNAI 600 025

ANNA UNIVERSITY CHENNAI: 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"Harnessing EC2, VPC & MOBAXTERM FOR CUTTING-EDGE ENTITY HOSTING"** is the Bonafide work of **"POSHIKA B-113322243075, RESHNI U-113322243084, RAMYA Y-113322243082, SANTHOSHINI S-113322243094"** who carried out the project work under my supervision.

SIGNATURE

DR.S.PADMAPRIYA, M.E, PH.D
PROFESSOR,
HEAD OF THE DEPARTMENT,

Artificial intelligence and Data Science,
Velammal Institute of Technology,
Velammal Gardens, Panchetti,
Panchetti, Chennai-601 204.

SIGNATURE

MR. DINESH KUMAR K
ASSISTANT PROFESSOR,
NM COORDINATOR,

Artificial intelligence and Data Science, Velammal Institute of Technology, Velammal Gardens, Panchetti, Panchetti, Chennai-601 204.

ACKNOWLEDGEMENT

We are personally indebted to many who had helped us during the course of this project work. Our deepest gratitude to the **God Almighty.**

We are greatly and profoundly thankful to our beloved Chairman **Thiru.M.V.Muthuramalingam** for facilitating us with this opportunity. Our sincere thanks to our respected Director **Thiru.M.V.M Sasi Kumar** for his consent to take up this project work and make it a great success.

We are also thankful to our Advisors **Shri.K.Razak**, **Shri.M.Vaasu**, our Principal **Dr.N.Balaji** and our Vice Principal **Dr.S.Soundararajan** for their never ending encouragement that drives ustowards innovation.

We are extremely thankful to our Head of the Department **Dr.S.PadmaPriya and** Naan Mudhalvan Coordinator **Mr.Dinesh Kumar**, for their valuable teachings and suggestions.

The Acknowledgment would be incomplete if we would not mention word of thanks to our Parents.Teaching and Non-Teaching Staffs, Administrative Staffs and Friends for their motivation and support throughout the project.

Finally,we thank all those who directly or indirectly contributed to the successful completion of this project .Your contributions have been a vital part of our succes

TABLE OF CONTENTS

Contents

CHAPTER 1

INTRODUCTION

In the rapidly advancing world of cloud computing, businesses are leveraging technologies like Amazon Web Services (AWS) and MobaXterm to create secure, scalable, and efficient hosting environments. AWS's Elastic Compute Cloud (EC2) and Virtual Private Cloud (VPC) stand out for their ability to provide on-demand computing power and isolated virtual networks, while MobaXterm simplifies remote access and system management, forming an integrated solution for modern infrastructure needs.

Amazon EC2 offers scalable virtual servers tailored to specific workloads, allowing businesses to pay only for what they use. With deployment across multiple regions, EC2 reduces latency and supports seamless scaling during high traffic or growth. AWS VPC enhances this by enabling businesses to define private networks, control traffic flow, and isolate resources through public and private subnets. Advanced security integrations, such as IAM and Security Groups, ensure robust protection against unauthorized access.

MobaXterm complements these services by offering an intuitive interface for securely managing EC2 instances. Supporting protocols like SSH, RDP, and SFTP, it allows administrators to efficiently handle tasks like configuration management, software deployment, and troubleshooting. Features like X11 forwarding and persistent SSH sessions further streamline workflows, enabling seamless remote management.

Together, EC2, VPC, and MobaXterm provide a comprehensive cloud solution. Businesses can host applications in secure VPC environments, with databases isolated in private subnets and public-facing services accessible via load balancers. MobaXterm ensures administrators can monitor and manage these systems securely, enhancing efficiency while protecting sensitive data.

This synergy is particularly valuable for modern development practices like Continuous Integration/Continuous Deployment (CI/CD). Teams can deploy automated pipelines, with EC2 and VPC maintaining secure production environments and MobaXterm facilitating resource management. This integration empowers organizations to innovate quickly, deliver high-quality services, and adapt to evolving market needs.

CHAPTER 2

PROJECT OVERVIEW

Project Overview

The project aims to design and implement a modern hosting infrastructure using Amazon Web Services (AWS) by integrating Elastic Compute Cloud (EC2), Virtual Private Cloud (VPC), and MobaXterm. This solution ensures secure, scalable, and efficient hosting of critical applications while enabling seamless management and remote access for administrators.

Purpose and Goals

The primary goal is to develop a robust cloud environment that meets the demands of modern businesses by leveraging AWS's flexibility, security, and scalability. Key objectives include:

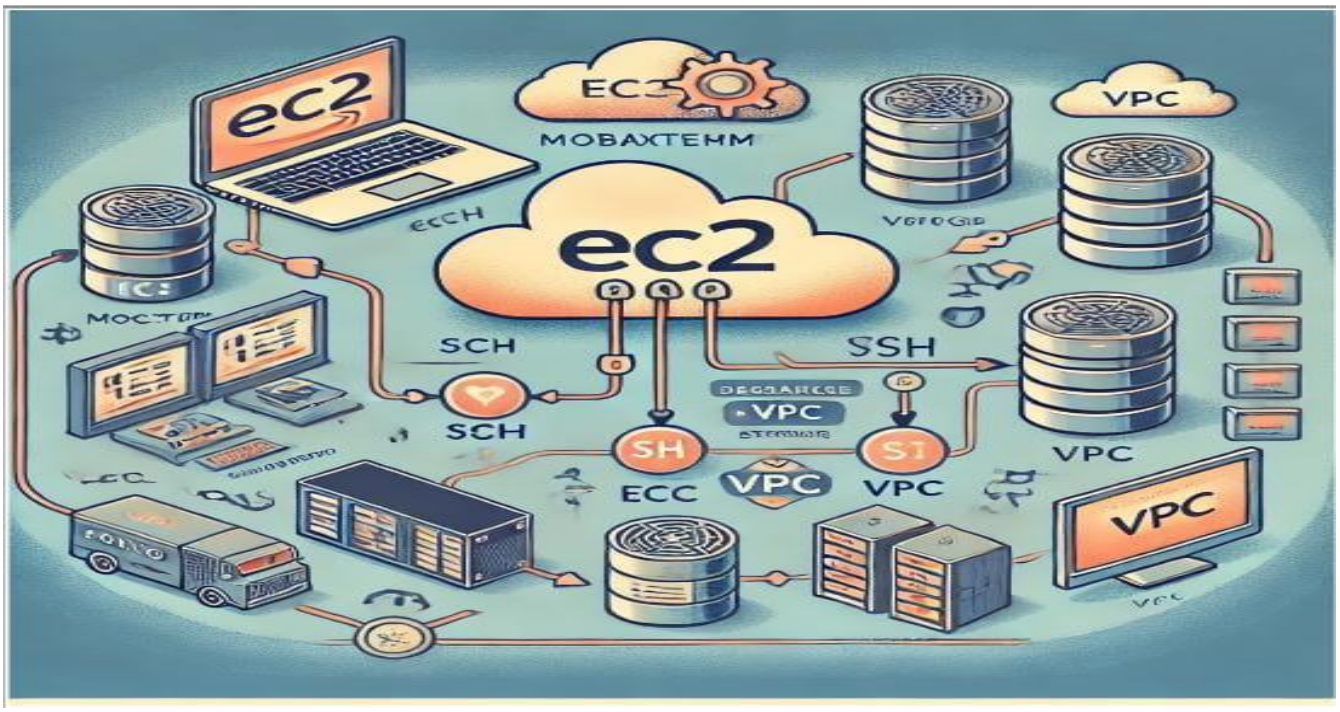
- **Dynamic Scalability:** Using EC2 instances with auto-scaling groups to optimize performance and cost.
- **Secure Networking:** Creating isolated environments with VPC, including private and public subnets, secure routing, and access controls.
- **Efficient Management:** Employing MobaXterm for remote access and streamlined system management.
- **High Availability:** Utilizing Elastic Load Balancing (ELB) and multiple Availability Zones to ensure fault tolerance and minimal downtime.
- **Cost Efficiency:** Adopting AWS’s pay-as-you-go model, reserved, and spot instances for optimized spending.

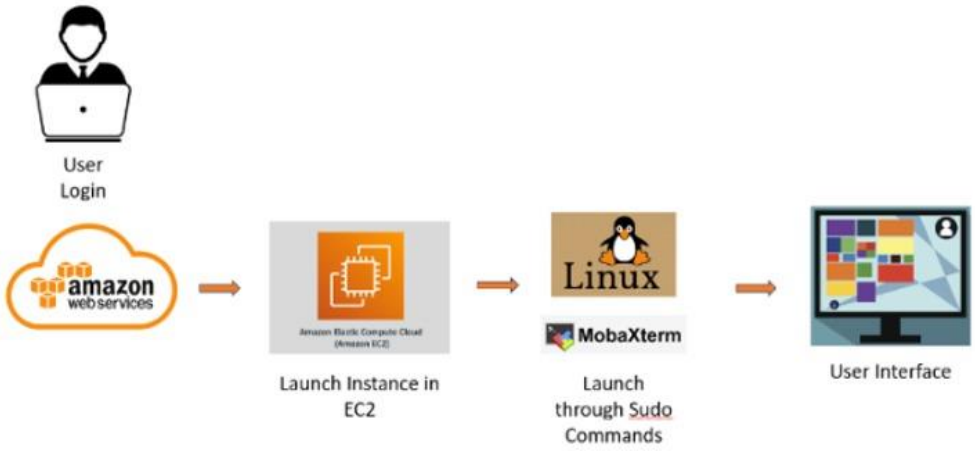
Key Features

- **EC2 for Compute Power:** On-demand, scalable instances across global regions with diverse configurations for different workloads.
- **VPC for Secure Networking:** Custom IP ranges, subnets, and granular access controls ensure isolation and protection of sensitive resources.
- **MobaXterm for Remote Access:** Secure, multi-session SSH/RDP access, file transfers, and persistent sessions simplify cloud management.
- **Security Measures:** IAM for role-based access.

CHAPTER 3

ARCHITECTURE





CHAPTER 4

SETUP INSTRUCTIONS

1. Setup AWS EC2 Instances

- **Create AWS Account**: Register at AWS Management Console.
- **Launch EC2 Instance**:
  - Choose an AMI (e.g., Amazon Linux 2 or Ubuntu).
  - Select an instance type (e.g., `t2.micro`).
  - Configure VPC, Subnet, and enable Public IP for internet access.
  - Set storage size and security groups (allow SSH-22, HTTP-80, HTTPS-443).
  - Create/select Key Pair for SSH access.
- **Access via SSH**: Use MobaXterm:
  - Enter EC2 Public IP in "Session > SSH".
  - Add `.pem` key for authentication.

2. Setup AWS VPC

- **Create VPC**: Define name and IP range (e.g., `10.0.0.0/16`). Enable DNS.
- **Create Subnets**:
  - Public Subnet: For web servers, allow internet access.
  - Private Subnet: For databases, block direct internet access.
- **Route Tables**: Assign internet routes for public subnet; add NAT gateway for private subnet (if needed).
- **Internet Gateway**: Attach to VPC for public internet access.
- **Security Groups**: Define inbound rules for required ports (e.g., SSH, HTTP, database).

3. Setup Database

- **Amazon RDS** (Relational):
  - Choose DB engine (e.g., MySQL), configure size, storage, and place in private subnet.
  - Note RDS endpoint for EC2 connection.
- **Amazon DynamoDB** (NoSQL):
  - Create a table with primary key, configure throughput, and optionally set VPC endpoints.

4. Use MobaXterm

- **Connect to EC2**: SSH into instance using Public IP and Key Pair.
- **Transfer Files**: Drag and drop files to EC2 via SFTP in MobaXterm.
- **Access Logs**: View logs via terminal (e.g., `cat /var/log/nginx/access.log`).

5. Setup Web Server

- **Install Web Server**:
  - For Apache: `sudo apt update && sudo apt install apache2`.
  - For Nginx: `sudo apt update && sudo apt install nginx`.
  - Enable service: `sudo systemctl start <service> && sudo systemctl enable <service>`.
- **Deploy Application**:
  - Upload website files (e.g., `/var/www/html` for Apache, `/usr/share/nginx/html` for Nginx).
  - Test: Open EC2 Public IP in a browser.

6. Optional: Setup Custom Domain

- Use Route 53: Create hosted zone and add an A record pointing to EC2 Public IP.
- Update Domain Registrar settings with Route 53 nameservers.

7. Security and Maintenance

- **Monitor with CloudWatch**: Set alarms for resource usage and receive notifications.
- **Backups**: Use RDS snapshots, DynamoDB backups, and S3 for storage.

This guide covers essential steps while ensuring scalability and security.

CHAPTER 5

RUNNING THE APPLICATION

1. Prerequisites

- **AWS EC2 Instance**: Running with correct instance type and security settings.
- **VPC Setup**: Proper subnet, route table, and security group configuration.
- **Mobaxterm**: Configured for SSH access with the `.pem` key.
- **Environment**: Install required runtime (Node.js, Python, etc.).

2. Access EC2 Instance

- Use Mobaxterm:
  - Enter EC2 Public IP in "Session > SSH".
  - Specify username (`ec2-user` for Amazon Linux, `ubuntu` for Ubuntu).
  - Provide `.pem` key path.
  - Connect to access the terminal.

3. Install Dependencies

- **Python Example**:

```
bash
Copy code
sudo apt install python3 python3-pip -y
pip3 install -r requirements.txt
```

4. Database Setup

- **AWS RDS**: Ensure connectivity from EC2 via VPC and security groups.
- **Local Database**:

```
bash
Copy code
sudo apt install mysql-server -y
sudo systemctl start mysql
sudo mysql_secure_installation
```

5. Start the Application

- **Python Example**:

```
bash
Copy code
flask run --host=0.0.0.0 --port=80
python3 manage.py runserver 0.0.0.0:80
```

6. Test the Application

- Access the app via `http://<EC2-Public-IP>:<port>`.
- Test API endpoints with tools like Postman or `curl`.
- Check logs for issues:

```
bash
Copy code
tail -f /path/to/your/logfile.log
```

7. Security Group Configuration

- Allow inbound traffic on necessary ports (22 for SSH, 80 for HTTP, 443 for HTTPS) via the AWS EC2 Console.

8. Restart or Shutdown

- Use commands specific to your app setup or process manager (e.g., `pm2`, `systemctl`).

CHAPTER 6

API DOCUMENTATION

API documentation provides a clear and concise description of the functionality and endpoints that your application exposes to other systems or clients. Below is a template for creating **API Documentation** for a typical RESTful API hosted on AWS EC2. This document will include endpoint descriptions, request parameters, response formats, and example usage.

Base URL

http://<your-ec2-public-ip> or http://<your-domain.com>

https://<your-ec2-public-ip> or https://<your-domain.com>

Example Request:

Bash

```
curl -X POST http://<your-ec2-public-ip>/api/products \
  -H "Authorization: Bearer <Your_Token>" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Product 3",
    "description": "Description of Product 3",
    "price": 39.99
  }'
```

Example Response:

json

```
{
  "message": "Product created successfully",
  "productId": "3"}
```

CHAPTER 7

AUTHENTICATION & AUTHORIZATION

Authentication & Authorization

Authentication and authorization are critical aspects of any API to ensure that only authorized users can access and modify data. Below is an explanation of how authentication and authorization work in a typical API setup, especially in a system hosted on AWS EC2.

1. Authentication

Authentication is the process of verifying the identity of a user or client application. It ensures that the user making the request is who they say they are.
For the API, **JWT (JSON Web Tokens)** is commonly used for authentication. The API authenticates users via a **login** process, where they provide credentials (username and password), and in return, they receive an authentication token (JWT). This token is then used in subsequent requests to verify the user's identity.

2. Authorization

Authorization is the process of determining whether an authenticated user has the necessary permissions to access a particular resource. Once the identity is verified via authentication, the system checks if the user has the correct **permissions** to perform the requested action (read, write, delete, etc.).
In the API, authorization is handled by checking the claims contained within the **JWT**. These claims can represent roles or permissions (e.g., "admin", "user", "editor") that determine what actions a user can perform.

CHAPTER 8

TESTING

Testing is an essential part of the development lifecycle, ensuring that an application behaves as expected, performs well under load, and is secure from vulnerabilities. For an API-based application hosted on AWS EC2, the testing process can involve several types of tests including **unit tests**, **integration tests**, **end-to-end tests**, and **performance tests**. Below is an overview of different testing strategies and tools that can be used to test the application and its components.

1. Unit Testing

- **Goal**: Test individual components in isolation (e.g., backend logic, database operations).
- **Tools**: PyTest (Python), Mocha/Chai, Jest (JavaScript).

2. Integration Testing

- **Goal**: Test interactions between APIs, databases, and external services.
- **Tools**: Postman (manual), Supertest, Jest (automated).

3. End-to-End (E2E) Testing

- **Goal**: Test full workflows to simulate user interactions (UI to backend).
- **Tools**: Cypress, Selenium, Puppeteer, TestCafe.

4. Performance Testing

- **Goal**: Ensure scalability and reliability under varying loads.
- **Types**: Load, Stress, Scalability tests.
- **Tools**: JMeter, Artillery, Locust, Gatling.

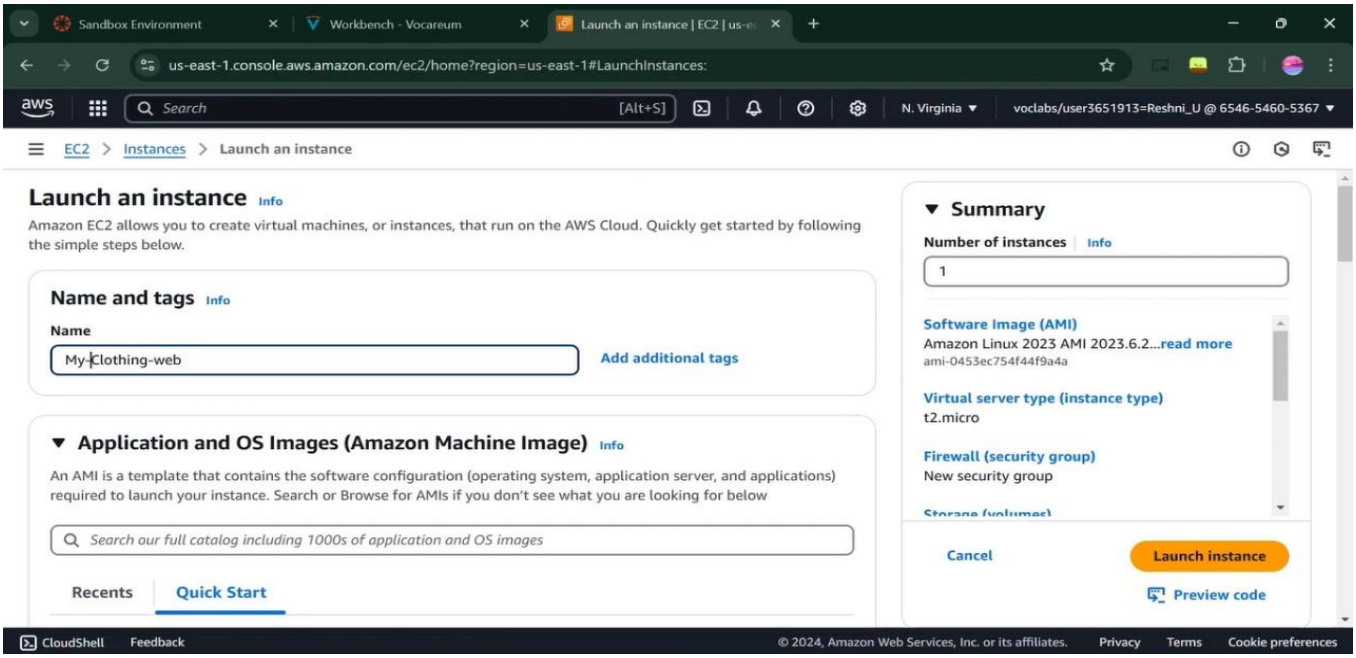
5. Security Testing

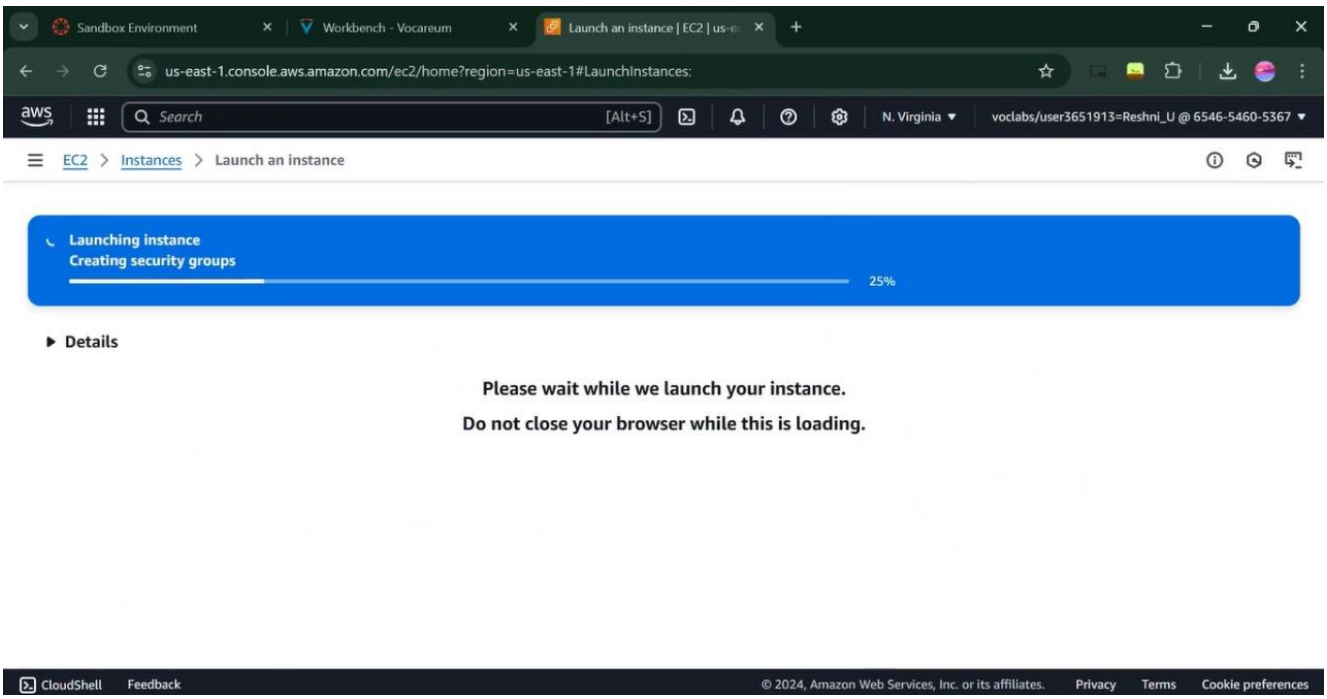
- **Goal**: Identify vulnerabilities (SQL injection, XSS, CSRF) and validate authentication/authorization.
- **Tools**: OWASP ZAP, Burp Suite, Snyk.

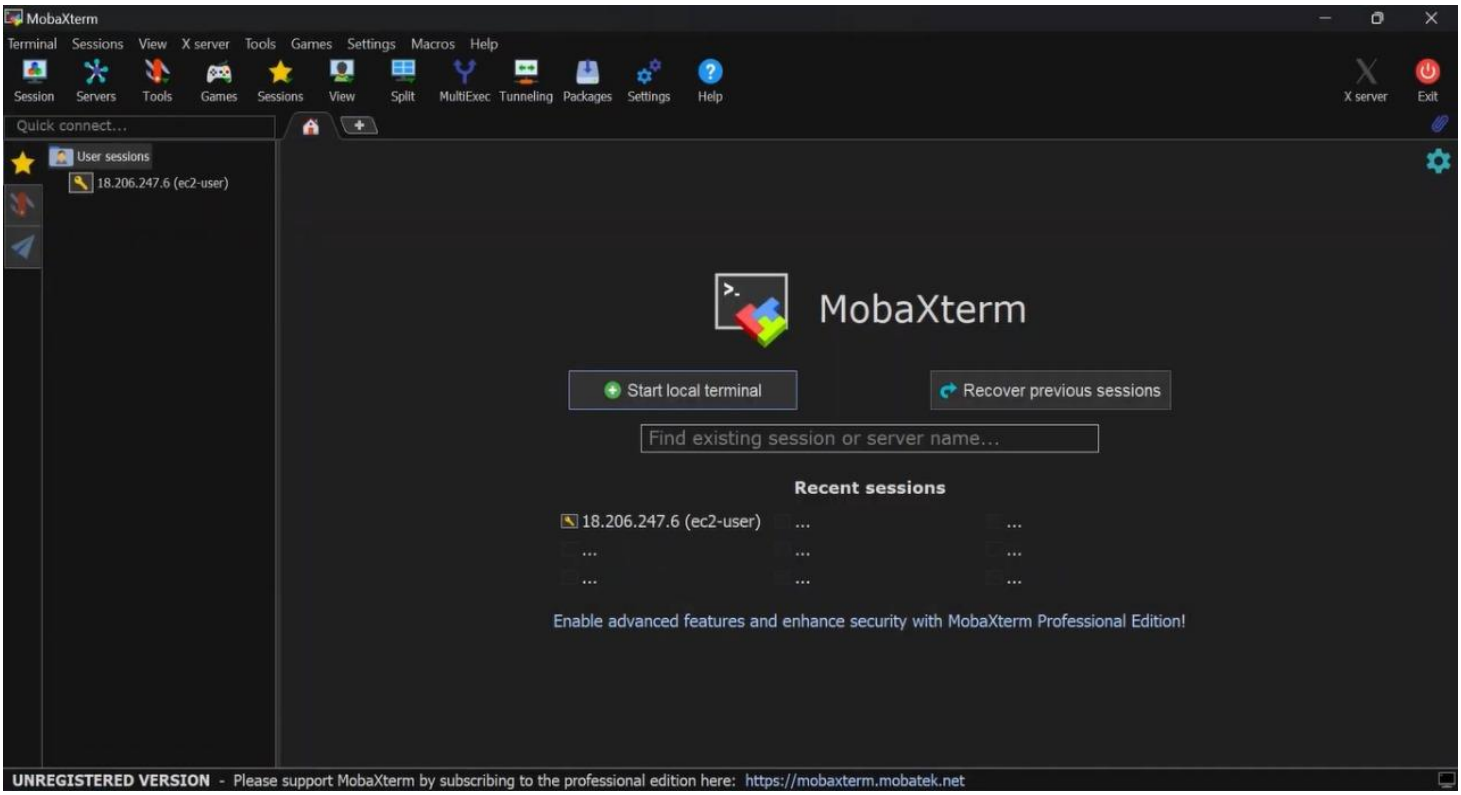
CHAPTER 9

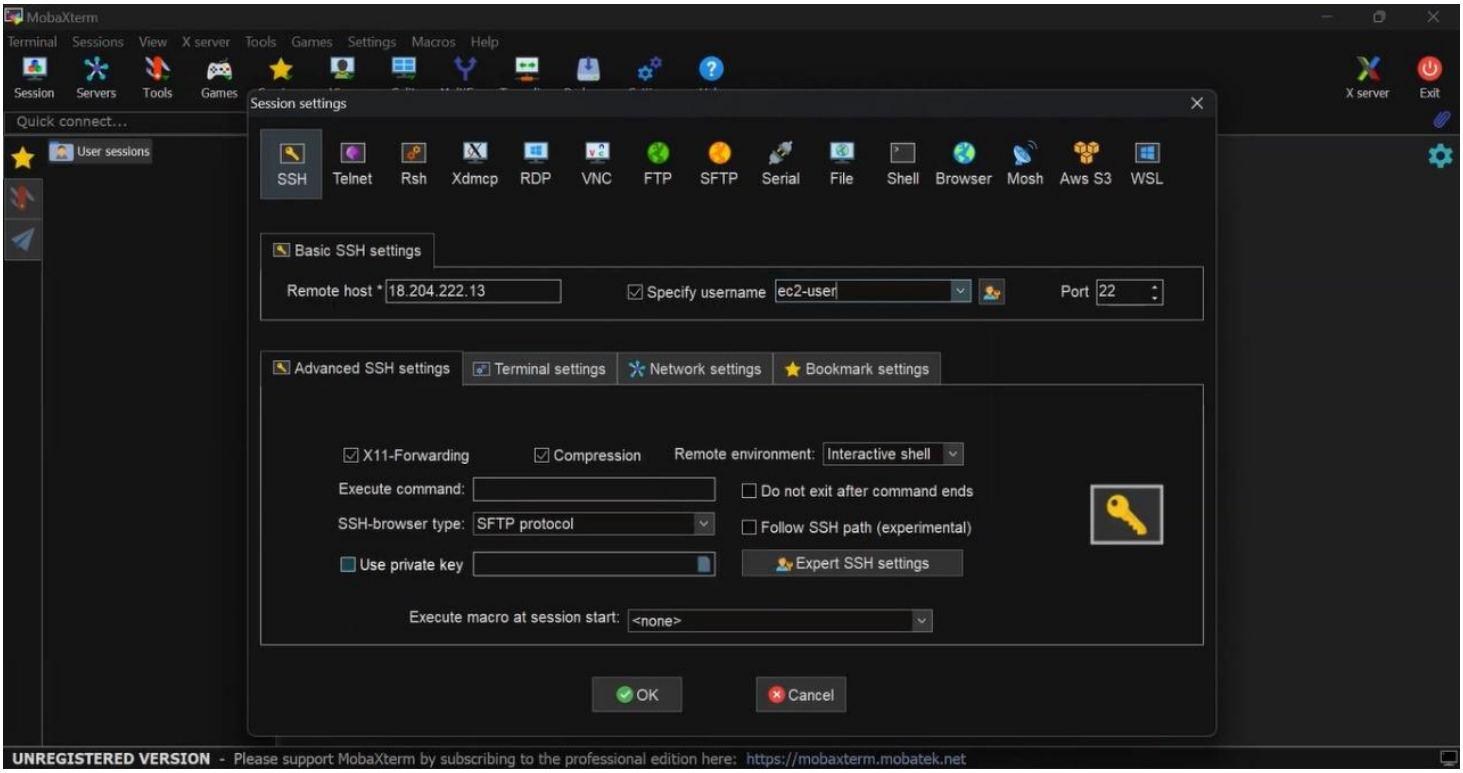
SCREENSHOTS

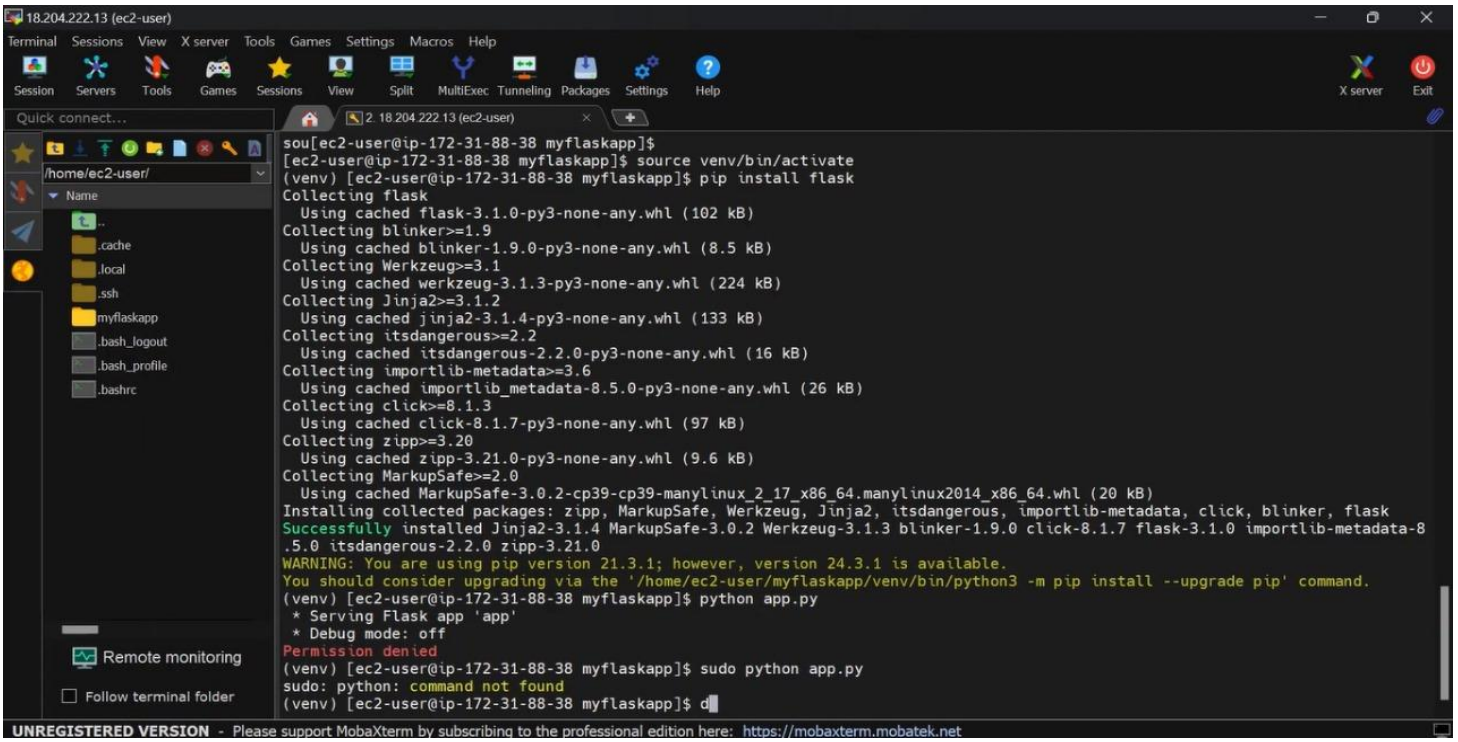
https://github.com/rea-git/NM2024TMID16176.git

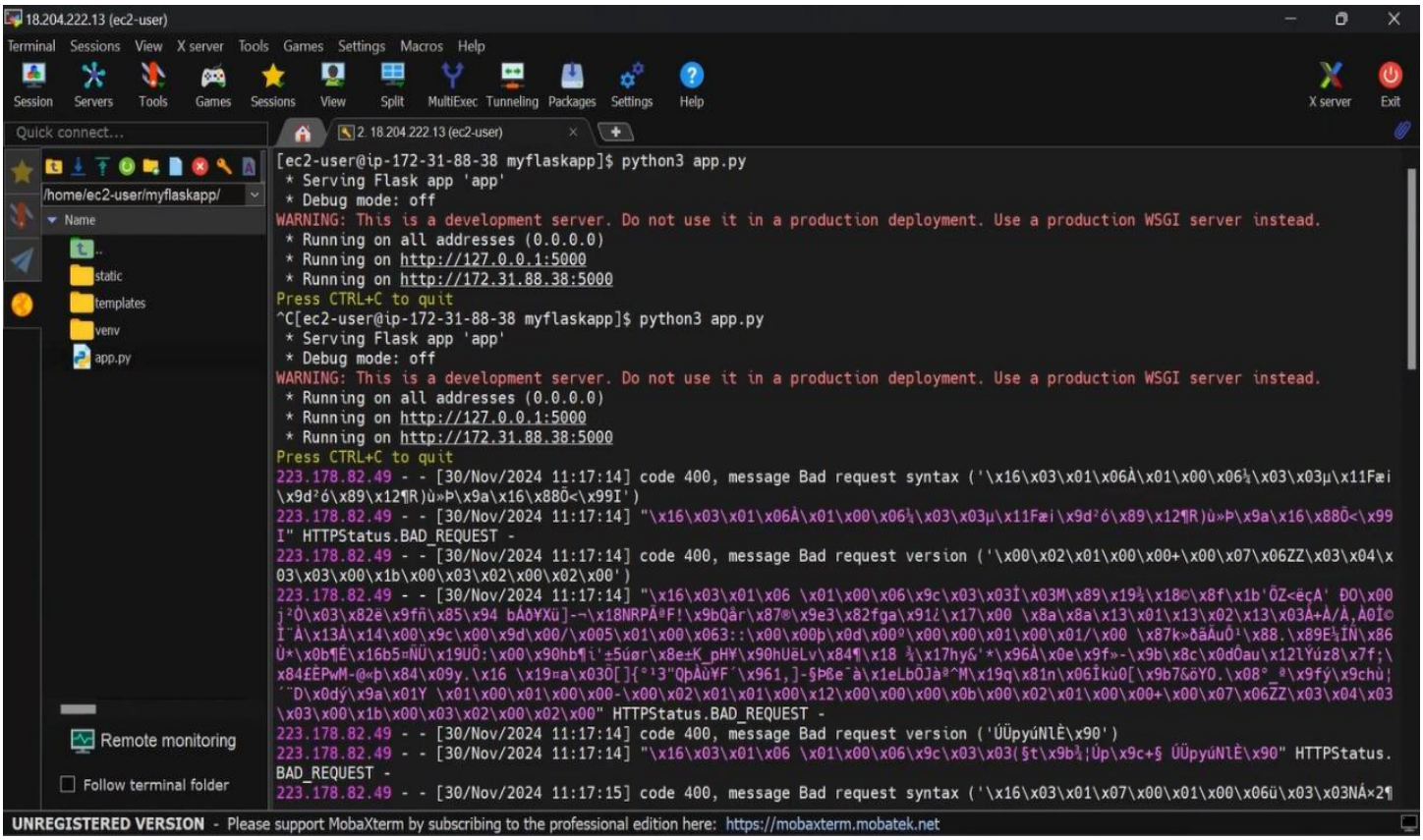


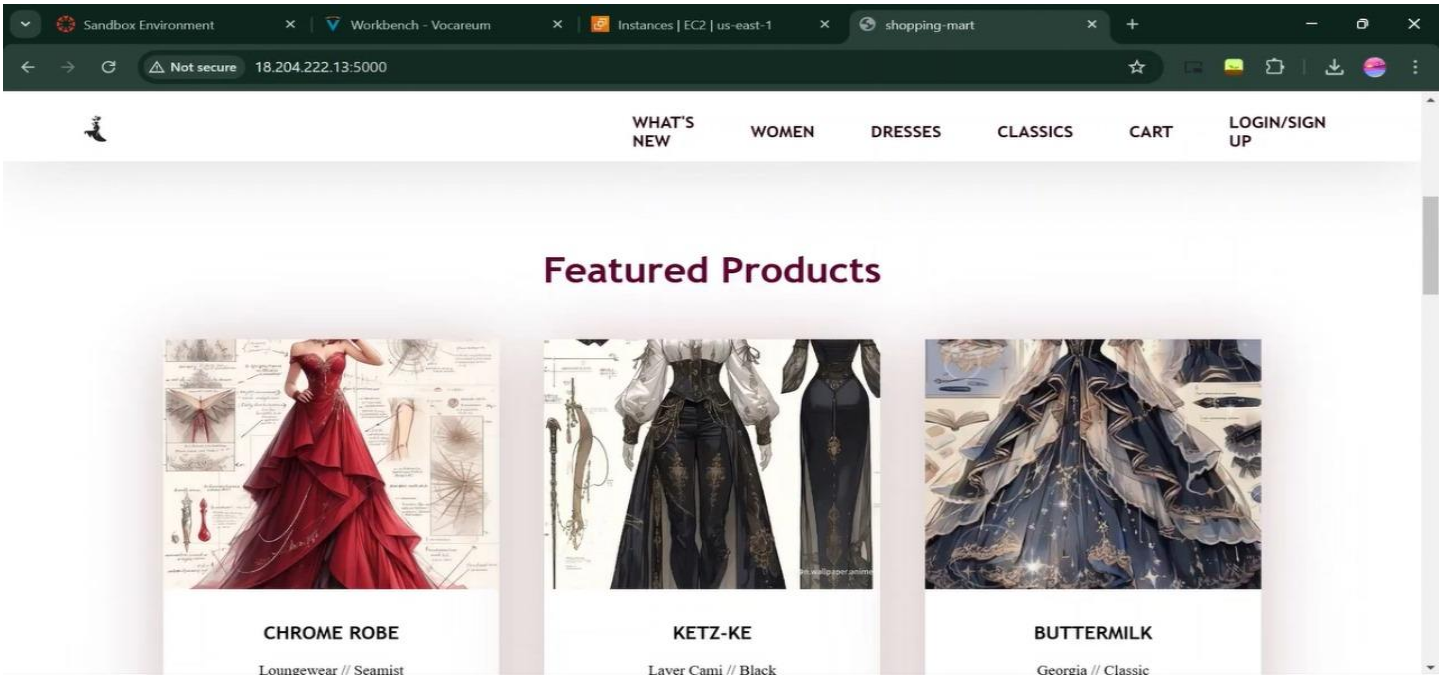


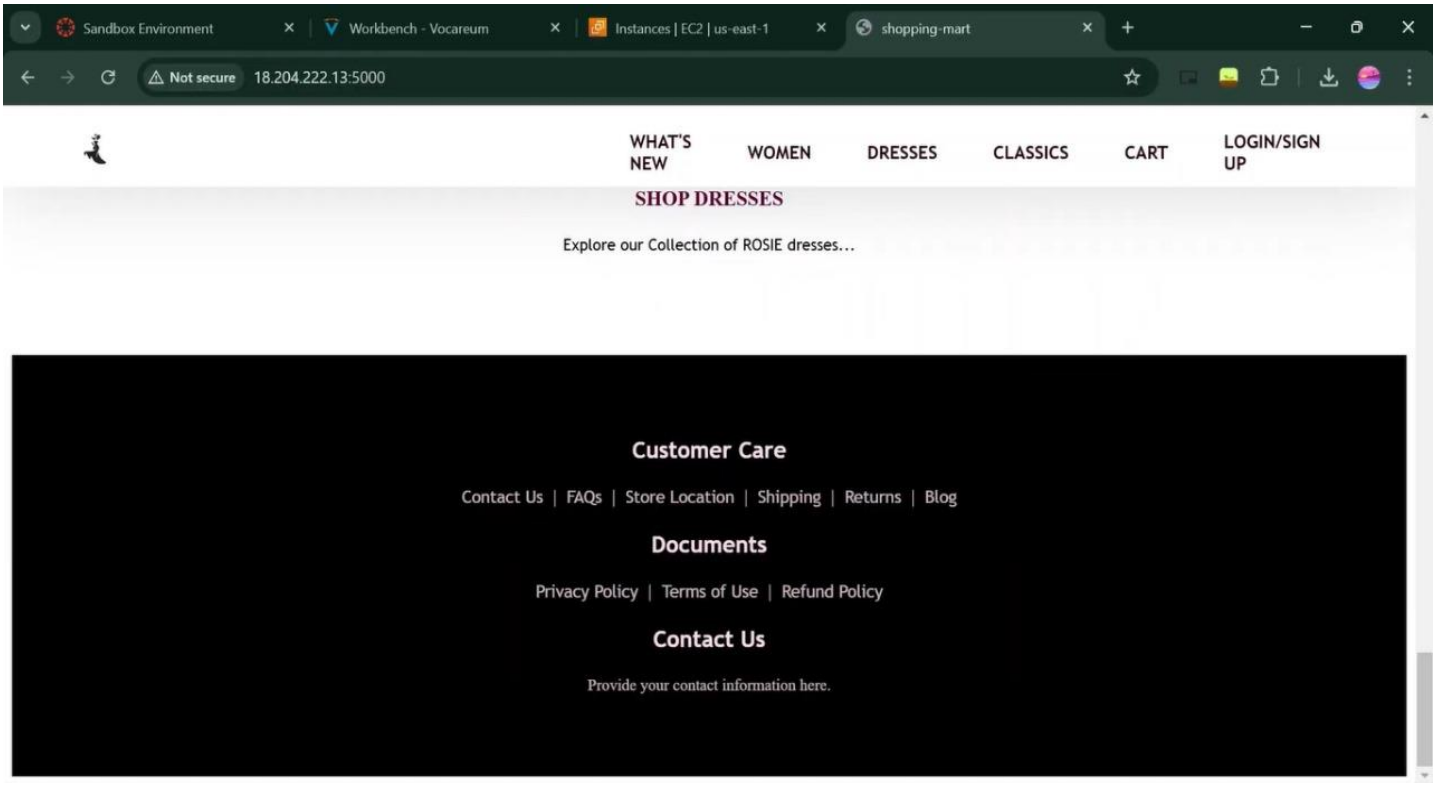

MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
User sessions
18.206.247.6 (ec2-user)
MobaXterm
Start local terminal
Recover previous sessions
Find existing session or server name...
Recent sessions
18.206.247.6 (ec2-user)
Enable advanced features and enhance security with MobaXterm Professional Edition!
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net


MobaXterm
Session settings
SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial File Shell Browser Mosh Aws S3 WSL
Basic SSH settings
Remote host * 18.204.222.13
Specify username ec2-user
Port 22
Advanced SSH settings
Terminal settings
Network settings
Bookmark settings
X11-Forwarding
Compression
Remote environment: Interactive shell
Execute command:
Do not exit after command ends
SSH-browser type: SFTP protocol
Follow SSH path (experimental)
Use private key
Expert SSH settings
Execute macro at session start: <none>
OK
Cancel
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net


18.204.222.13 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. 18.204.222.13 (ec2-user)
/home/ec2-user/
Name
.cache
.local
.ssh
myflaskapp
.bash_logout
.bash_profile
.bashrc
Remote monitoring
Follow terminal folder
sou[ec2-user@ip-172-31-88-38 myflaskapp]$
[ec2-user@ip-172-31-88-38 myflaskapp]$ source venv/bin/activate
(venv) [ec2-user@ip-172-31-88-38 myflaskapp]$ pip install flask
Collecting flask
Using cached flask-3.1.0-py3-none-any.whl (102 kB)
Collecting blinker>=1.9
Using cached blinker-1.9.0-py3-none-any.whl (8.5 kB)
Collecting Werkzeug>=3.1
Using cached werkzeug-3.1.3-py3-none-any.whl (224 kB)
Collecting Jinja2>=3.1.2
Using cached jinja2-3.1.4-py3-none-any.whl (133 kB)
Collecting itsdangerous>=2.2
Using cached itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting importlib-metadata>=3.6
Using cached importlib_metadata-8.5.0-py3-none-any.whl (26 kB)
Collecting click>=8.1.3
Using cached click-8.1.7-py3-none-any.whl (97 kB)
Collecting zipp>=3.20
Using cached zipp-3.21.0-py3-none-any.whl (9.6 kB)
Collecting MarkupSafe>=2.0
Using cached MarkupSafe-3.0.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (20 kB)
Installing collected packages: zipp, MarkupSafe, Werkzeug, Jinja2, itsdangerous, importlib-metadata, click, blinker, flask
Successfully installed Jinja2-3.1.4 MarkupSafe-3.0.2 Werkzeug-3.1.3 blinker-1.9.0 click-8.1.7 flask-3.1.0 importlib-metadata-8.5.0 itsdangerous-2.2.0 zipp-3.21.0
WARNING: You are using pip version 21.3.1; however, version 24.3.1 is available.
You should consider upgrading via the '/home/ec2-user/myflaskapp/venv/bin/python3 -m pip install --upgrade pip' command.
(venv) [ec2-user@ip-172-31-88-38 myflaskapp]$ python app.py
* Serving Flask app 'app'
* Debug mode: off
Permission denied
(venv) [ec2-user@ip-172-31-88-38 myflaskapp]$ sudo python app.py
sudo: python: command not found
(venv) [ec2-user@ip-172-31-88-38 myflaskapp]$ d
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net


18.204.222.13 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. 18.204.222.13 (ec2-user)
/home/ec2-user/myflaskapp/
Name
static
templates
venv
app.py
Remote monitoring
Follow terminal folder
[ec2-user@ip-172-31-88-38 myflaskapp]$ python3 app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.31.88.38:5000
Press CTRL+C to quit
^C[ec2-user@ip-172-31-88-38 myflaskapp]$ python3 app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.31.88.38:5000
Press CTRL+C to quit
223.178.82.49 - - [30/Nov/2024 11:17:14] code 400, message Bad request syntax ('\x16\x03\x01\x06Å\x01\x00\x06¾\x03\x03µ\x11Fæi\x9d²ó\x89\x12¶R)ù»Þ\x9a\x16\x88Õ<\x99I')
223.178.82.49 - - [30/Nov/2024 11:17:14] "\x16\x03\x01\x06Å\x01\x00\x06¾\x03\x03µ\x11Fæi\x9d²ó\x89\x12¶R)ù»Þ\x9a\x16\x88Õ<\x99I" HTTPStatus.BAD_REQUEST -
223.178.82.49 - - [30/Nov/2024 11:17:14] code 400, message Bad request version ('\x00\x02\x01\x00\x00+\x00\x07\x06ZZ\x03\x04\x03\x03\x03\x00\x1b\x00\x03\x02\x00\x02\x00')
223.178.82.49 - - [30/Nov/2024 11:17:14] "\x16\x03\x01\x06 \x01\x00\x06\x9c\x03\x03Ì\x03M\x89\x19¾\x18©\x8f\x1b'ÕZ<ëçA' ÐO\x00j²Ò\x03\x82ë\x9fñ\x85\x94 bÁð¥Xü]-¬\x18NRPÅª F!\x9bQâr\x87®\x9e3\x82fga\x91¿\x17\x00 \x8a\x8a\x13\x01\x13\x02\x13\x03ÀÀ,À0Ì©Ì¨Ä\x13À\x14\x00\x9c\x00\x9d\x00/\x005\x01\x00\x063::\x00\x00b\x0d\x00º\x00\x00\x01\x00\x01/\x00 \x87k»ðãÂuÛ¹\x88.\x89E¾ÍN\x86Ù*\x0b¶Ê\x16b5¤ÑÜ\x19UÕ:\x00\x90hb¶i'±5úør\x8e±K_pHÝ\x90hUëLv\x84¶\x18 ¾\x17hy&'*\x96Á\x0e\x9f»-\x9b\x8c\x0dÔau\x12lÝúz8\x7f;\x84£ÉPwM-@«þ\x84\x09y.\x16 \x19¤a\x03Ô[]{°¹3"QþÀÙ¥F´\x961,]-§Þ®e¯à\x1eLbOJà²^M\x19q\x81n\x06Íkù0[\x9b7&öYO.\x08° ¸ª\x9fý\x9chù¦´¨D\x0dý\x9a\x01Y \x01\x00\x01\x00\x00-\x00\x02\x01\x01\x00\x12\x00\x00\x00\x0b\x00\x02\x01\x00\x00+\x00\x07\x06ZZ\x03\x04\x03\x03\x00\x1b\x00\x03\x02\x00\x02\x00" HTTPStatus.BAD_REQUEST -
223.178.82.49 - - [30/Nov/2024 11:17:14] code 400, message Bad request version ('ÚÜpyúNlÈ\x90')
223.178.82.49 - - [30/Nov/2024 11:17:14] "\x16\x03\x01\x06 \x01\x00\x06\x9c\x03\x03(§t\x9b¼¦Úp\x9c+§ ÚÜpyúNlÈ\x90" HTTPStatus.BAD_REQUEST -
223.178.82.49 - - [30/Nov/2024 11:17:15] code 400, message Bad request syntax ('\x16\x03\x01\x07\x00\x01\x00\x06ü\x03\x03NÁ×2¶
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net


Sandbox Environment
Workbench - Vocareum
Instances | EC2 | us-east-1
shopping-mart
Not secure
18.204.222.13:5000
WHAT'S NEW
WOMEN
DRESSES
CLASSICS
CART
LOGIN/SIGN UP
Featured Products
CHROME ROBE
Loungewear // Seamist
KETZ-KE
Layer Cami // Black
BUTTERMILK
Georgia // Classic


Sandbox Environment
Workbench - Vocareum
Instances | EC2 | us-east-1
shopping-mart
Not secure
18.204.222.13:5000
WHAT'S NEW
WOMEN
DRESSES
CLASSICS
CART
LOGIN/SIGN UP
SHOP DRESSES
Explore our Collection of ROSIE dresses...
Customer Care
Contact Us | FAQs | Store Location | Shipping | Returns | Blog
Documents
Privacy Policy | Terms of Use | Refund Policy
Contact Us
Provide your contact information here.


aws academy
Poshika Babu
Certificate of Completion for
AWS Academy Graduate - AWS Academy Cloud Foundations
Course hours completed
20 hours
Issued on
11/07/2024
Digital badge
https://www.credly.com/go/ISOKxrIC


aws academy
Reshni U
Certificate of Completion for
AWS Academy Graduate - AWS Academy Cloud Foundations
Course hours completed
20 hours
Issued on
11/07/2024
Digital badge
https://www.credly.com/go/u2okvAD2


aws academy
Ramya Y
Certificate of Completion for
AWS Academy Graduate - AWS Academy Cloud Foundations
Course hours completed
20 hours
Issued on
11/07/2024
Digital badge
https://www.credly.com/go/uRndy6mt
aws academy
Santhoshini S
Certificate of Completion for
AWS Academy Graduate - AWS Academy Cloud Foundations
Course hours completed
20 hours
Issued on
11/07/2024
Digital badge
https://www.credly.com/go/6NMyD2nn