

## CI/CD Deployment Assignment

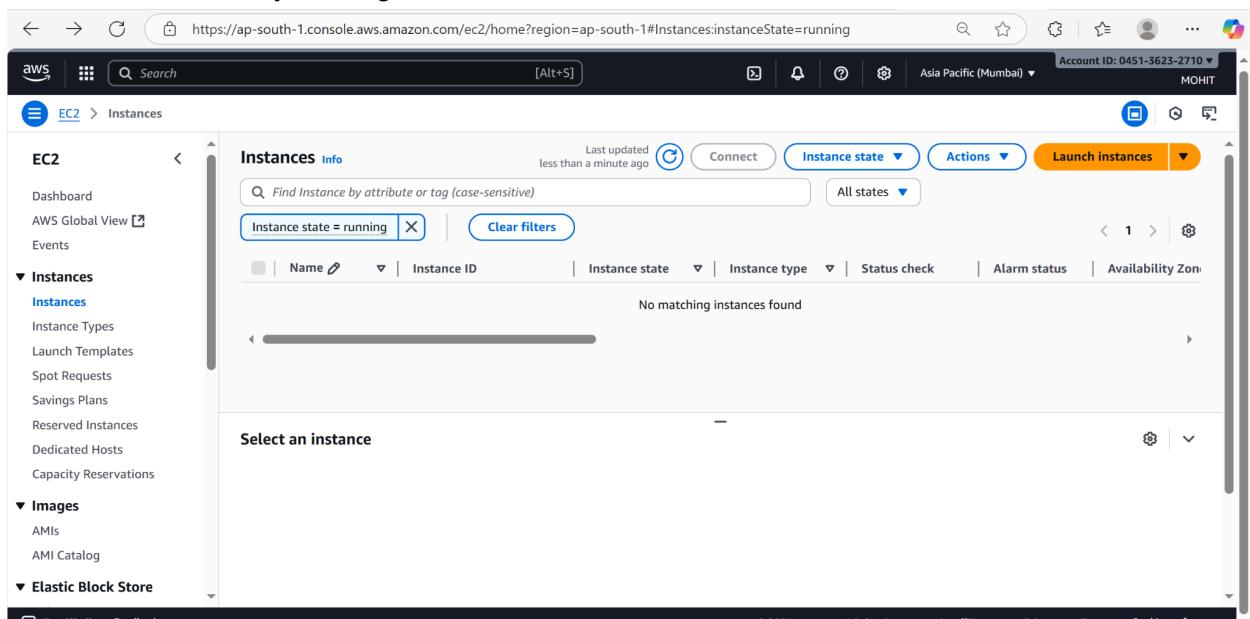
You are required to deploy a **Flask backend** and an **Express frontend** on an **Amazon EC2 instance**. Additionally, implement a **CI/CD pipeline** using Jenkins to automate the deployment process.

---

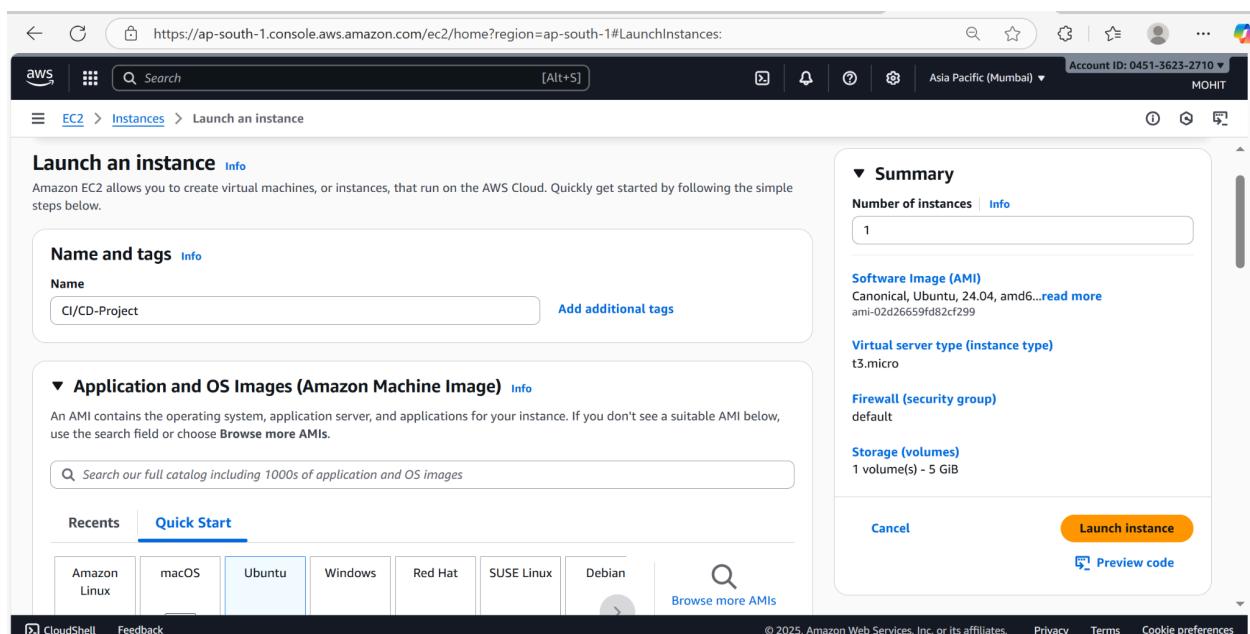
### Part 1: Deploy Flask and Express on a Single EC2 Instance

1. **Objective:**
  - Deploy both the Flask backend and the Express frontend on a **single Amazon EC2 instance**.
2. **Steps:**
  - **Provisioning the EC2 Instance:**
    - Launch an EC2 instance on AWS (you can use a free-tier eligible instance).
    - SSH into the instance and install the following dependencies:
      - Python for Flask.
      - Node.js for Express.
      - Git for pulling the application code.
  - **Application Setup:**
    - Clone the Flask and Express repositories onto the EC2 instance.
    - Install the required dependencies for both applications using `pip` and `npm`.
    - Configure both applications to run on different ports (e.g., Flask on port 5000 and Express on port 3000).
    - Start the applications using process managers like `pm2` or `systemd` to ensure they remain active.
3. **Deliverables:**
  - A running EC2 instance with Flask and Express accessible via the instance's public IP.
  - A description or diagram of the deployed architecture.

## Create an instance by clicking on the launch instances



The screenshot shows the AWS EC2 Instances page. The left sidebar is expanded, showing categories like EC2, Instances, Images, and Elastic Block Store. Under Instances, 'Instances' is selected. The main content area is titled 'Instances Info' and shows a search bar with 'Find Instance by attribute or tag (case-sensitive)' and a filter button 'Instance state = running'. Below the search bar are buttons for 'Connect', 'Actions', and 'Launch instances'. A message 'No matching instances found' is displayed. At the bottom, there's a section titled 'Select an instance'.

The screenshot shows the 'Launch an instance' wizard. The top navigation bar includes 'CloudShell' and 'Feedback'. The main content area has a heading 'Launch an instance' with a sub-section 'Name and tags'. It shows a 'Name' input field containing 'CI/CD-Project' and a 'Add additional tags' link. Below this is a section for 'Application and OS Images (Amazon Machine Image)'. It features a search bar and tabs for 'Recent' and 'Quick Start'. Under 'Quick Start', there are buttons for 'Amazon Linux', 'macOS', 'Ubuntu' (which is highlighted in blue), 'Windows', 'Red Hat', 'SUSE Linux', and 'Debian'. A 'Browse more AMIs' link is also present. To the right, a 'Summary' section shows 'Number of instances' set to 1. Other summary sections include 'Software Image (AMI)', 'Virtual server type (instance type)', 'Firewall (security group)', and 'Storage (volumes)'. At the bottom right are 'Cancel', 'Launch instance', and 'Preview code' buttons.

Now click on connect and run following code on terminal

The screenshot shows the AWS EC2 Connect interface. At the top, it displays the URL <https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#ConnectToInstance:instanceId=i-0946ef7191ef7e1b>. The page title is "EC2 > Instances > i-0946ef7191ef7e1b > Connect to instance". On the right, it shows "Account ID: 0451-3623-2710" and "MOHIT". Below the title, there's a "Connect" section with tabs for "Info", "EC2 Instance Connect", "Session Manager", and "SSH client" (which is selected). The "Info" tab contains a brief description: "Connect to an instance using the browser-based client." Below this, under "Instance ID", it shows "i-0946ef7191ef7e1b (CI/CD-Project)". A numbered list provides instructions: 1. Open an SSH client. 2. Locate your private key file. The key used to launch this instance is `hello-aws.pem`. 3. Run this command, if necessary, to ensure your key is not publicly viewable.  
`chmod 400 "hello-aws.pem"`. 4. Connect to your instance using its Public DNS:  
`ec2-13-232-225-230.ap-south-1.compute.amazonaws.com`. An "Example:" section shows the command:  
`ssh -i "hello-aws.pem" ubuntu@ec2-13-232-225-230.ap-south-1.compute.amazonaws.com`. A note at the bottom states: "Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username." A "Cancel" button is at the bottom right.

The screenshot shows a terminal window titled "CICD" with the command `ssh -i "hello-aws.pem" ubuntu@ec2-13-232-225-230.ap-south-1.compute.amazonaws.com` running. The terminal output is as follows:

```
PS C:\Users\dell\Downloads\CICD> ssh -i "hello-aws.pem" ubuntu@ec2-13-232-225-230.ap-south-1.compute.amazonaws.com
ubuntu@ip-172-31-33-69:~$ 1 -----BFGTN RSA PRTVATF KFY-----
System load: 0.0 Temperature: -273.1 C
Usage of /: 19.9% of 8.65GB Processes: 115
Memory usage: 22% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.31.33.69

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

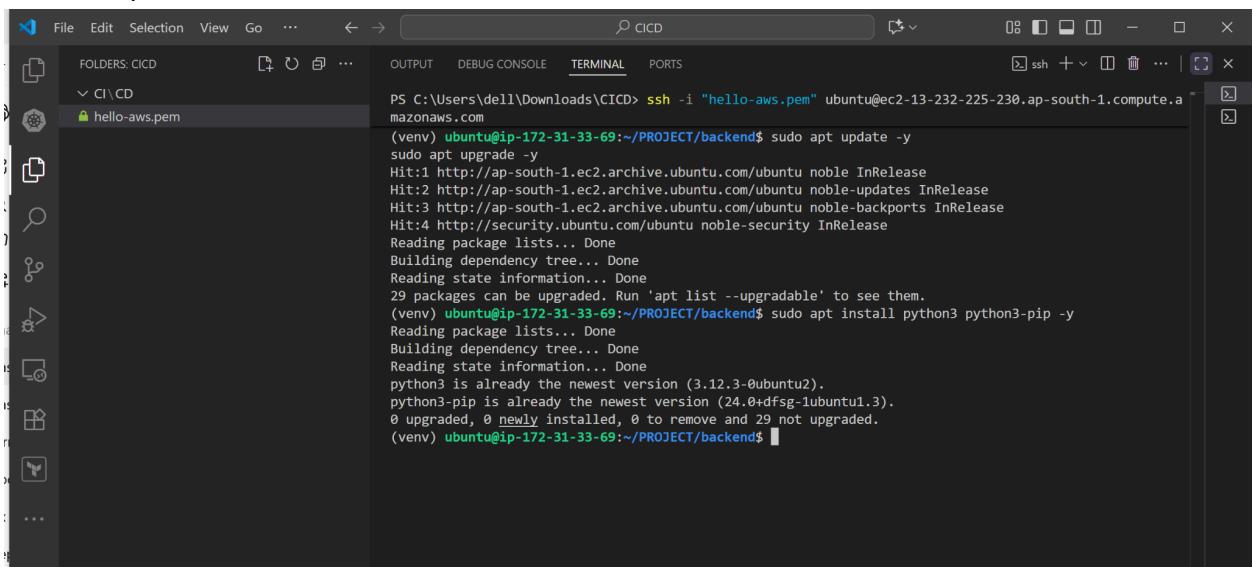
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

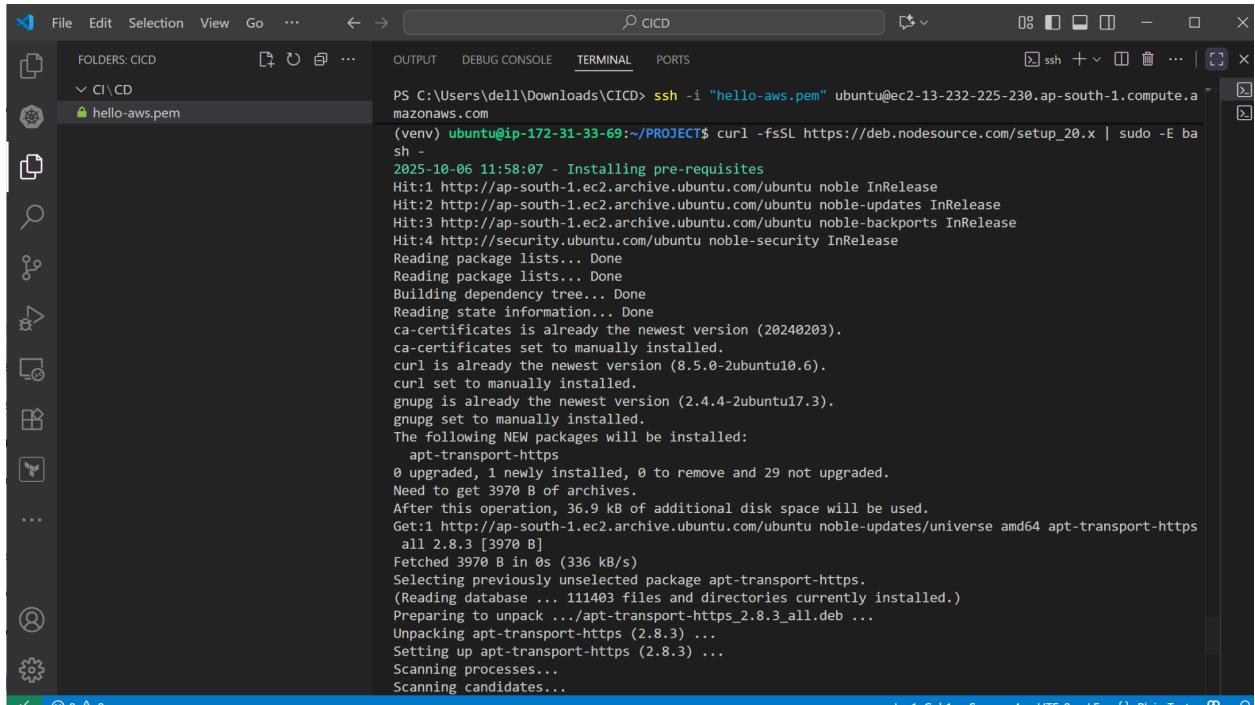
ubuntu@ip-172-31-33-69:~$
```

## Install Dependencies



The screenshot shows a terminal window titled "CICD" with the following command history:

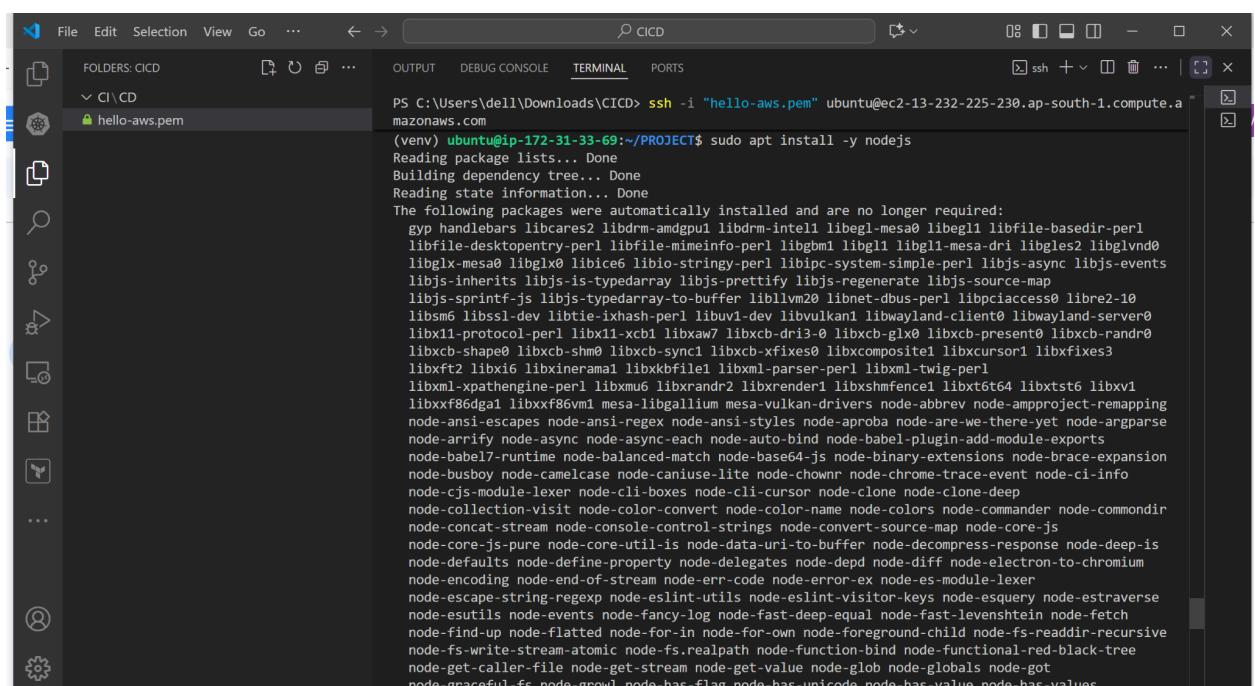
```
PS C:\Users\dell\Downloads\CICD> ssh -i "hello-aws.pem" ubuntu@ec2-13-232-225-230.ap-south-1.compute.amazonaws.com
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/backend$ sudo apt update -y
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
29 packages can be upgraded. Run 'apt list --upgradable' to see them.
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/backend$ sudo apt install python3 python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
python3-pip is already the newest version (24.0+dfsg-1ubuntu1.3).
0 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/backend$
```



```

PS C:\Users\dell\Downloads\CICD> ssh -i "hello-aws.pem" ubuntu@ec2-13-232-225-230.ap-south-1.compute.amazonaws.com
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
2025-10-06 11:58:07 - Installing pre-requisites
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 29 not upgraded.
Need to get 3970 B of archives.
After this operation, 36.9 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 apt-transport-https all 2.8.3 [3970 B]
Fetched 3970 B in 0s (336 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 111403 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.8.3_all.deb ...
Unpacking apt-transport-https (2.8.3) ...
Setting up apt-transport-https (2.8.3) ...
Scanning processes...
Scanning candidates...

```



```

PS C:\Users\dell\Downloads\CICD> ssh -i "hello-aws.pem" ubuntu@ec2-13-232-225-230.ap-south-1.compute.amazonaws.com
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ sudo apt install -y nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gyp handlebars libcares2 libdrm-amdgpu1 libdrm-intel1 libegl1 libfile-basedir-perl
  libfile-desktopentry-perl libfile-mimeinfo-perl libgbm1 libgl1_mesa-dri libgles2 libglvnd0
  libglx-mesa0 libglx0 libice6 libio-stringy-perl libipc-system-simple-perl libjs-async libjs-events
  libjs-inherits libjs-is-typedarray libjs-prettify libjs-regenerate libjs-source-map
  libjs-sprintf-js libjs-typedarray-to-buffer libllm20 libnet-dbus-perl libpiciaccess0 libre2-10
  libsm6 libssl-dev libtie-ixhash-perl libv1-dev libvulkan1 libwayland-client0 libwayland-server0
  libx11-protocol-perl libx11-xcb1 libxaw7 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0
  libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcompositel1 libxcursor1 libxfixed3
  libxft2 libxi6 libxinerama1 libxcbfile1 libxml1-parser-perl libxml-twig-perl
  libxml-xpathengine-perl libxmlmu6 libxrandr2 libxrender1 libxshmfence1 libxt6t64 libxtst6 libxv1
  libxxf86dga1 libxxf86vm1 mesa-libgallium mesa-vulkan-drivers node-abrev node-amp-project-remapping
  node-ansi-escapes node-ansi-rege node-ansi-styles node-aproba node-are-we-there-yet node-argparse
  node-arrify node-async node-async-each node-auto-bind node-babel-plugin-add-module-exports
  node-babel7-runtime node-balanced-match node-base64-js node-binary-extensions node-brace-expansion
  node-busboy node-camelcase node-canuse-lite node-chown node-chrome-trace-event node-ci-info
  node-cjs-module-lexer node-cli-boxes node-cli-cursor node-clone node-clone-deep
  node-collection-visit node-color-convert node-color-name node-colors node-commander node-commandir
  node-concat-stream node-console-control-strings node-convert-source-map node-core-js
  node-core-js-pure node-core-util-is node-data-uri-to-buffer node-decompress-response node-deep-is
  node-defaults node-define-property node-delegates node-depd node-diff node-electron-to-chromium
  node-encoding node-end-of-stream node-err-code node-error-ex node-es-module-lexer
  node-escape-string-regexp node-eslint-utils node-eslint-visitor-keys node-esquery node-estraverse
  node-esutils node-events node-fancy-log node-fast-deep-equal node-fast-levenshtein node-fetch
  node-find-up node-flatted node-for-in node-for-own node-foreground-child node-fs-readaddr-recursive
  node-fs-write-stream node-fs.realpath node-function-bind node-functional-red-black-tree
  node-get-caller-file node-get-stream node-get-value node-glob node-globals node-got
  node-graceful-fs node-growl node-has-flag node-has-unicode node-has-value node-has-values

```

```

(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ python3 --version
Python 3.12.3
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ pip3 --version
pip 24.0 from /home/ubuntu/PROJECT/backend/venv/lib/python3.12/site-packages/pip (python 3.12)
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ node -v
v20.19.5
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ npm -v
10.8.2
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ 

```

## Install Git

The screenshot shows a terminal window titled "CICD" with the following command history:

```
PS C:\Users\dell\Downloads\CICD> ssh -i "hello-aws.pem" ubuntu@ec2-13-232-225-230.ap-south-1.compute.amazonaws.com
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ python3 --version
Python 3.12.3
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ pip3 --version
pip 24.0 from /home/ubuntu/PROJECT/backend/venv/lib/python3.12/site-packages/pip (python 3.12)
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ node -v
v20.19.5
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ npm -v
10.8.2
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ sudo apt install git -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
The following packages were automatically installed and are no longer required:
  gyp handlebars libcares2 libdrm-amdgpu1 libdrm-intel1 libegl-mesa0 libegl1 libfile-basedir-perl
  libfile-desktopentry-perl libfile-mimeinfo-perl libgbm1 libgl1-mesa-dri libgles2 libglvnd0
  libglx-mesa0 libglx0 libice6 libio-stringy-perl libipc-system-simple-perl libjs-asyn libjs-events
  libjs-inherits libjs-is-typedarray libjs-prettify libjs-regenerate libjs-source-map
  libjs-sprintf.js libjs-typedarray-to-buffer libl10n20 libnet-dbus-perl libpciaccess0 libre2-10
  libsm6 libssl-dev libtie-ixhash-perl libuv1-dev libvulkan1 libwayland-client0 libwayland-server0
  libxml1-protocol-perl libxml1-xcb1 libxmlaw7 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0
  libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcompositel libxcursor1 libxfixed3
  libxft2 libxi6 libxinerama1 libxkbfile1 libxml-parser-perl libxml-twig-perl
  libxml-xpathengine-perl libxmlmu6 libxrandr2 libxshmfence1 libxtst64 libxv1
  libxxfb6dga1 libxf86vm1 mesa-mesa-vulkan-drivers node-abrev node-ampprop node-remapping
  node-ansi-escapes node-ansi-regex node-ansi-styles node-aproba node-are-we-there-yet node-argparse
  node-arrify node-async node-async-each node-auto-bind node-babel-plugin-add-module-exports
  node-babel7-runtime node-balanced-match node-base64-js node-binary-extensions node-brace-expansion
  node-busboy node-camelcase node-canisue-lite node-chown node-chrome-trace-event node-ci-info
  node-cjs-module-lexer node-cli-boxes node-cli-cursor node-clone node-clone-deep
  node-collection-visit node-color-convert node-color-name node-colors node-commander node-commandir
```

## Now clone frontend and backend code from git repository

The screenshot shows a GitHub repository page for "PROJECT". The "Code" tab is selected, displaying the repository structure:

- main
- 1 Branch
- 0 Tags

The repository contains the following files:

- Mohit7819 Add files via upload
- backend
- frontend
- .gitignore
- README.md
- docker-compose.yml
- explanation.pdf

On the right side, there is a "Clone" section with three options:

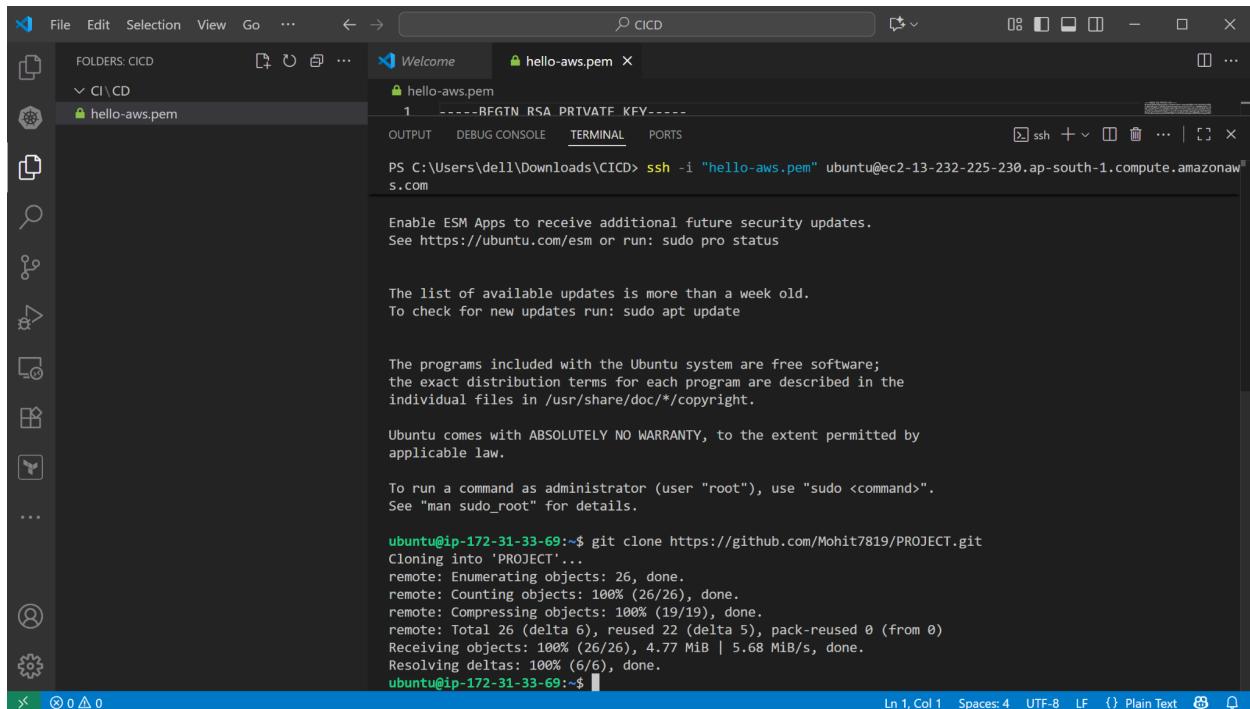
- HTTPS: <https://github.com/Mohit7819/PROJECT.git>
- SSH: (disabled)
- GitHub CLI: (disabled)

Below the clone options are links to "Open with GitHub Desktop" and "Download ZIP".

The repository details on the right include:

- About: No description, website, or topics provided.
- Activity: 0 events
- Stars: 0 stars
- Watching: 0 watching
- Forks: 0 forks
- Releases: No releases published. [Create a new release](#)
- Packages: No packages published. [Publish your first package](#)

Now run git clone <https://github.com/Mohit7819/PROJECT.git> in terminal



```
File Edit Selection View Go ... Welcome hello-aws.pem
FOLDERS: CICD
CI\CD hello-aws.pem
hello-aws.pem
1 -----RFGTN RSA PRTVATE KFY-----
OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\dell\Downloads\CICD> ssh -i "hello-aws.pem" ubuntu@ec2-13-232-225-230.ap-south-1.compute.amazonaws.com
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-33-69:~$ git clone https://github.com/Mohit7819/PROJECT.git
Cloning into 'PROJECT'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 26 (delta 6), reused 22 (delta 5), pack-reused 0 (from 0)
Receiving objects: 100% (26/26), 4.77 MiB | 5.68 MiB/s, done.
Resolving deltas: 100% (6/6), done.
ubuntu@ip-172-31-33-69:~$
```

Run Apps with PM2

```
0 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$ sudo npm install -g pm2
added 133 packages in 9s

13 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.6.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.6.1
npm notice To update run: npm install -g npm@11.6.1
npm notice
(venv) ubuntu@ip-172-31-33-69:~/PROJECT$
```

```
Access pm2 files in ~/.pm2
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/backend$ pm2 start app.py --name flask-backend --interpreter python3
[PM2] Starting /home/ubuntu/PROJECT/backend/app.py in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	o	status	cpu	memory
0	flask-backend	fork	0	online	0%	8.5mb

```
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/backend$
```

```
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/frontend$ pm2 start app.js --name express-frontend
[PM2] Starting /home/ubuntu/PROJECT/frontend/app.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	ø	status	cpu	memory
1	express-frontend	fork	0	online	0%	22.2mb
0	flask-backend	fork	0	online	0%	32.0mb

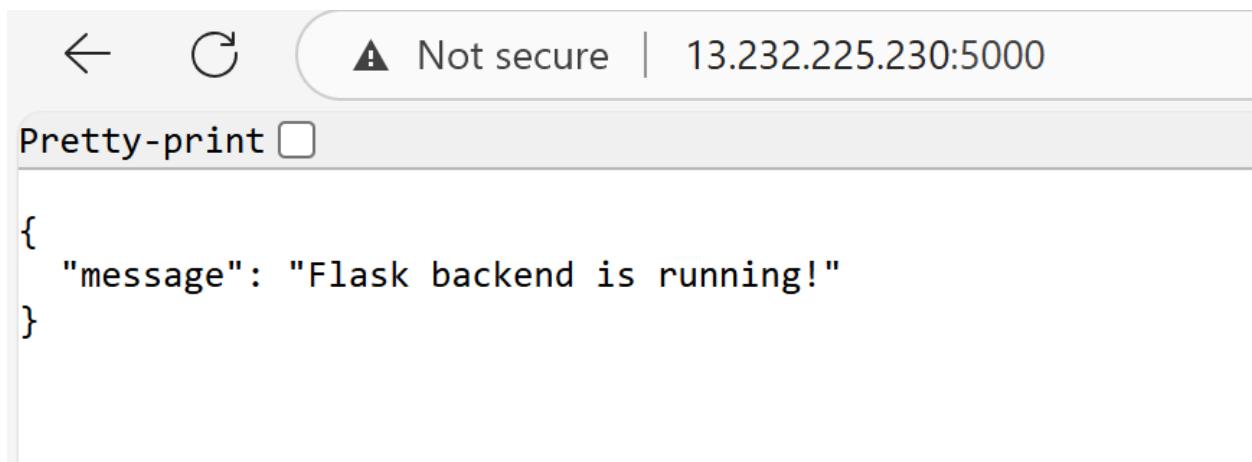
```
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/frontend$ █
```

## Save PM2 Process

```
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/frontend$ pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /home/ubuntu/.pm2/dump.pm2
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/frontend$ pm2 startup
[PM2] Init System found: systemd
[PM2] To setup the Startup Script, copy/paste the following command:
sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u ubuntu --hp /home/ubuntu
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/frontend$ █
```

## Tests apps

- Flask: <http://13.232.225.230:5000>



- Express:

A screenshot of a web browser window. The address bar shows the URL `13.232.225.230:3000`. The page title is **Send Data to Flask Backend**. There are two input fields: one for Name containing "cicd" and one for Email containing "cicd@gmail.com". A "Submit" button is below the fields.

Name:

Email:

Submit

A screenshot of a web browser window. The address bar shows the URL `13.232.225.230:3000/submit-form`. The page title is **Flask Backend Response**. It displays a message: **Message:** Data received successfully! followed by the submitted data: **Name:** cicd and **Email:** cicd@gmail.com. A blue Go back link is at the bottom.

**Message:** Data received successfully!

**Name:** cicd

**Email:** cicd@gmail.com

[Go back](#)

For stopping the app

```
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/frontend$ pm2 stop express-frontend
[PM2] Applying action stopProcessId on app [express-frontend](ids: [ 1 ])
[PM2] [express-frontend](1) ✓
```

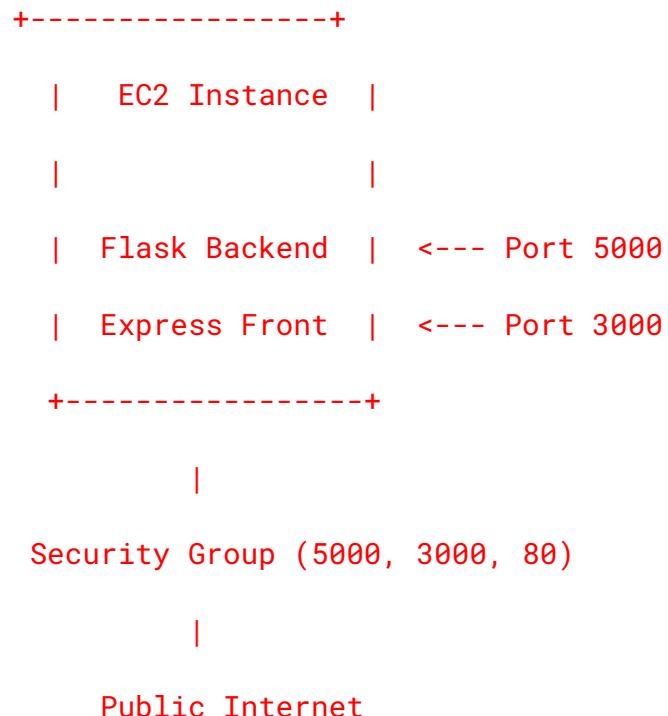
<b>id</b>	<b>name</b>	<b>mode</b>	<b>⌚</b>	<b>status</b>	<b>cpu</b>	<b>memory</b>
<b>1</b>	express-frontend	fork	0	<b>stopped</b>	0%	0b
<b>0</b>	flask-backend	fork	0	<b>online</b>	0%	32.0mb

```
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/frontend$ pm2 stop flask-backend
[PM2] Applying action stopProcessId on app [flask-backend](ids: [ 0 ])
[PM2] [flask-backend](0) ✓
```

<b>id</b>	<b>name</b>	<b>mode</b>	<b>⌚</b>	<b>status</b>	<b>cpu</b>	<b>memory</b>
<b>1</b>	express-frontend	fork	0	<b>stopped</b>	0%	0b
<b>0</b>	flask-backend	fork	0	<b>stopped</b>	0%	0b

```
(venv) ubuntu@ip-172-31-33-69:~/PROJECT/frontend$ █
```

## Architecture Diagram



## Part 2: Implement CI/CD Pipeline Using Jenkins

1. **Objective:**
  - Automate the deployment of Flask and Express applications using Jenkins.
2. **Steps:**
  - **Install Jenkins:**
    - Install Jenkins on the same EC2 instance or on a separate machine.
    - Configure Jenkins by installing essential plugins like Git, NodeJS, and Python.
  - **Set Up Jenkins Pipeline:**
    - Create two separate Jenkins pipelines for the Flask and Express applications.
    - **Pipeline Steps:**
      - Pull the latest code from the respective Git repositories.
      - Install dependencies for Flask (`pip install -r requirements.txt`) and Express (`npm install`).
      - Restart the applications using the process manager (e.g., `pm2 restart <app>`).
  - **Triggering the Pipeline:**
    - Set up a GitHub webhook to trigger the Jenkins pipeline on every push to the repositories.
    -
  - **Optional Enhancements:**
    - Add testing stages to the pipeline for both applications.
    - Configure environment variables in Jenkins for managing sensitive data like API keys.
3. **Deliverables:**
  - A fully functional CI/CD pipeline that automates the deployment process for Flask and Express.
  - A Jenkins pipeline script (`Jenkinsfile`) for each application.
  - Evidence of the pipeline working (e.g., screenshots of successful builds and deployments).

## Creating instances by clicking on launch instances.

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed, and the main area displays the 'Instances' section under the 'Info' tab. A search bar at the top right contains the placeholder 'Find Instance by attribute or tag (case-sensitive)'. Below it, a filter bar shows 'Instance state = running' and a 'Clear filters' button. The main table header includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. A message 'No matching instances found' is centered below the table. At the bottom of the page, there is a 'Select an instance' section and a footer with copyright information and links to Privacy, Terms, and Cookie preferences.

<https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances>

**Launch an instance** [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

**Name and tags** [Info](#)

Name  Add additional tags

**Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recent OS Images: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian

[CloudShell](#) [Feedback](#)

**Summary**

Number of instances [Info](#)

1

Software Image (AMI)  
Canonical, Ubuntu, 24.04, amd64... [read more](#)  
ami-02d26659fd82cf299

Virtual server type (instance type)  
t3.micro

Firewall (security group)  
default

Storage (volumes)  
1 volume(s) - 10 GiB

[Cancel](#) [Launch instance](#) [Preview code](#)

[2025, Amazon Web Services, Inc. or its affiliates.](#) [Privacy](#) [Terms](#) [Cookie preferences](#)

<https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:instanceState=running>

**Instances** (1/1) [Info](#)

Last updated less than a minute ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive)

Instance state = running [X](#) [Clear filters](#)

<input checked="" type="checkbox"/>	Name <a href="#">D</a>	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input checked="" type="checkbox"/>	ci/cd	i-0d7d395228003aee0	<span>Running</span> <a href="#">Q</a> <a href="#">Q</a>	t3.micro	<span>Initializing</span> <a href="#">Q</a>	<a href="#">View alarms +</a>	ap-south-1a

**i-0d7d395228003aee0 (ci/cd)**

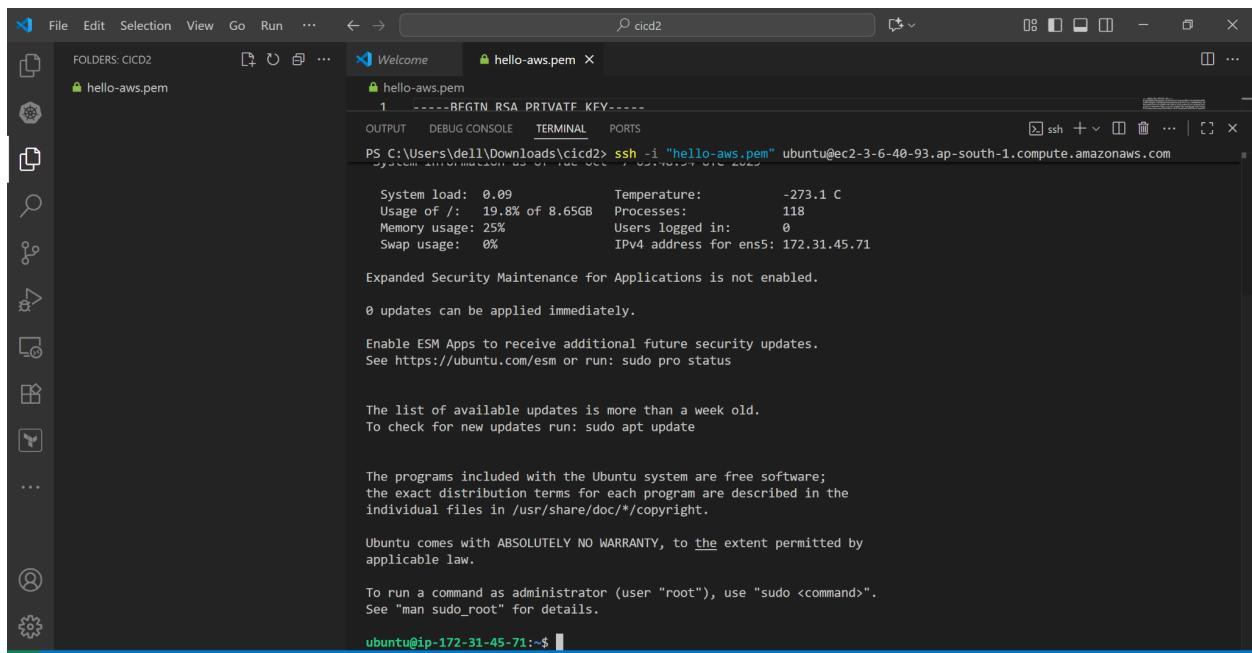
[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

**Instance summary** [Info](#)

Instance ID <a href="#">i-0d7d395228003aee0</a>	Public IPv4 address <a href="#">3.6.40.93   open address</a>	Private IPv4 addresses <a href="#">172.31.45.71</a>
--	---	--

Now connect instances to terminal by clicking on the connect and paste the given command in your terminal.

The screenshot shows the AWS EC2 Connect interface. At the top, there's a navigation bar with tabs for EC2, Instances, and Connect to instance. Below that, a sub-navigation bar shows 'EC2 Instance Connect' selected. The main content area is titled 'Connect Info' and contains instructions for connecting via an SSH client. It lists four steps: 1. Open an SSH client, 2. Locate your private key file (hello-aws.pem), 3. Run the command chmod 400 "hello-aws.pem", and 4. Connect to your instance using its Public DNS (ec2-3-6-40-93.ap-south-1.compute.amazonaws.com). An example command is provided at the bottom: ssh -i "hello-aws.pem" ubuntu@ec2-3-6-40-93.ap-south-1.compute.amazonaws.com.



System update + create **deployer** user by following commands:

```
sudo apt update -y && sudo apt upgrade -y
```

```
# create deployer
```

```
sudo adduser --disabled-password --gecos "" deployer
```

```
sudo usermod -aG sudo deployer
```

```
# create .ssh for deployer (used later)

sudo mkdir -p /home/deployer/.ssh

sudo chown -R deployer:deployer /home/deployer/.ssh

sudo chmod 700 /home/deployer/.ssh
```

The screenshot shows a terminal window titled "cid2" with the following session details:

- Folders: CID2
- File: hello-aws.pem
- SSH Session: hello-aws.pem (connected to ubuntu@ec2-3-6-40-93.ap-south-1.compute.amazonaws.com)
- Terminal tab is selected.
- Output pane shows the command history and systemctl logs.
- Debug Console and Ports tabs are also visible.

```
PS C:\Users\dell\Downloads\cid2> ssh -i "hello-aws.pem" ubuntu@ec2-3-6-40-93.ap-south-1.compute.amazonaws.com
systemctl restart ModemManager.service
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart serial-getty@ttyS0.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
ubuntu @ session #1: apt[1604], sshd[1129,1239]
ubuntu @ user manager service: systemd[1134]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-45-71:~$ sudo adduser --disabled-password --gecos "" deployer
info: Adding user `deployer' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `deployer' (1001) ...
info: Adding new user `deployer' (1001) with group `deployer' (1001) ...
info: Creating home directory `/home/deployer' ...
info: Copying files from `/etc/skel' ...
info: Adding new user `deployer' to supplemental / extra groups `users' ...
info: Adding user `deployer' to group `users' ...
ubuntu@ip-172-31-45-71:~$ sudo usermod -aG sudo deployer
ubuntu@ip-172-31-45-71:~$ sudo mkdir -p /home/deployer/.ssh
ubuntu@ip-172-31-45-71:~$ sudo chown -R deployer:deployer /home/deployer/.ssh
ubuntu@ip-172-31-45-71:~$ sudo chmod 700 /home/deployer/.ssh
ubuntu@ip-172-31-45-71:~$
```

Install required packages (Java, Git, Node, Python, rsync, etc.) and pm2

```
# basic tools

sudo apt install -y git python3 python3-venv python3-pip build-essential rsync curl

# install openjdk (Jenkins requirement)

sudo apt install -y openjdk-11-jdk

java -version

# Node.js (LTS)

curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

```
sudo apt-get install -y nodejs  
node -v && npm -v  
  
# global pm2 (for Node and python service management)  
sudo npm install -g pm2  
pm2 -v
```

```

c:\Users\dell\Downloads\cicd2> ssh -i "hello-aws.pem" ubuntu@ec2-3-6-40-93.ap-south-1.compute.amazonaws.com
Running kernel version:
6.14.0-1011-aws
Diagnostics:
The currently running kernel version is not the expected kernel version 6.14.0-1014-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart serial-getty@ttyS0.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
ubuntu @ session #1: sshd[1129,1239]
ubuntu @ user manager service: systemd[1134]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-45-71:~$ java -version
openjdk version "11.0.28" 2025-07-15
OpenJDK Runtime Environment (build 11.0.28+6-post-Ubuntu-1ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 11.0.28+6-post-Ubuntu-1ubuntu124.04.1, mixed mode, sharing)
ubuntu@ip-172-31-45-71:~$ 

Ln 1, Col 1  Spaces:4  UTF-8  LF  {} Plain Text  ⚙  🔍
```

```

c:\Users\dell\Downloads\cicd2> ssh -i "hello-aws.pem" ubuntu@ec2-3-6-40-93.ap-south-1.compute.amazonaws.com
Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart serial-getty@ttyS0.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
ubuntu @ session #1: sshd[1129,1239]
ubuntu @ user manager service: systemd[1134]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-45-71:~$ sudo npm install -g pm2
:)

added 133 packages in 9s

13 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.6.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.6.1
npm notice To update run: npm install -g npm@11.6.1
npm notice
ubuntu@ip-172-31-45-71:~$ 

Ln 1, Col 1  Spaces:4  UTF-8  LF  {} Plain Text  ⚙  🔍
```

## Install Jenkins (official repo) and start service

```
# add Jenkins repo (modern method)
```

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

```
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]  
https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list
```

```
sudo apt update
```

```
sudo apt install -y jenkins
```

```
# enable and start
```

```
sudo systemctl enable --now jenkins
```

```
sudo systemctl status jenkins
```

```

c:\Users\dell\Downloads\cicd2> ssh -i "hello-aws.pem" ubuntu@ec2-3-6-40-93.ap-south-1.compute.amazonaws.com
13 packages are looking for funding
  run `npm fund` for details
npm notice New major version of npm available! 10.8.2 -> 11.6.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.6.1
npm notice To update run: npm install -g npm@11.6.1
npm notice
ubuntu@ip-172-31-45-71:~$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
ubuntu@ip-172-31-45-71:~$ echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list
deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/
ubuntu@ip-172-31-45-71:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:7 https://deb.nodesource.com/node_18.x nodistro InRelease
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [29.7 kB]
Hit:9 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 32.6 kB in 1s (51.7 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
ubuntu@ip-172-31-45-71:~$ sudo apt install -y jenkins
ln 1, Col 1 Spaces:4 UTF-8 LF {} Plain Text

at Hudson.webAppMain$5.run(WebAppMain.java:277)
ubuntu@ip-172-31-45-71:~$ sudo systemctl enable --now jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-45-71:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
  Active: active (running) since Tue 2025-10-07 04:28:01 UTC; 4s ago
    Main PID: 19189 (java)
       Tasks: 40 (limit: 1008)
      Memory: 307.4M (peak: 323.4M)
        CPU: 21.914s
       CGroup: /system.slice/jenkins.service
           └─19189 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jen
Oct 07 04:27:53 ip-172-31-45-71 jenkins[19189]: [LF]> This may also be found at: /var/lib/jenkins/secrets/initialAdminP
Oct 07 04:27:53 ip-172-31-45-71 jenkins[19189]: [LF]>
Oct 07 04:27:53 ip-172-31-45-71 jenkins[19189]: [LF]> ****
Oct 07 04:27:53 ip-172-31-45-71 jenkins[19189]: [LF]> ****
Oct 07 04:27:53 ip-172-31-45-71 jenkins[19189]: [LF]> ****
Oct 07 04:28:00 ip-172-31-45-71 jenkins[19189]: 2025-10-07 04:28:00.987+0000 [id=32]           INFO      jenkins.InitRe>
Oct 07 04:28:01 ip-172-31-45-71 jenkins[19189]: 2025-10-07 04:28:01.017+0000 [id=23]           INFO      hudson.lifecycle>
Oct 07 04:28:01 ip-172-31-45-71 systemd[1]: Started Jenkins Continuous Integration Server.
Oct 07 04:28:02 ip-172-31-45-71 jenkins[19189]: 2025-10-07 04:28:02.988+0000 [id=49]           INFO      h.m.DownloadSe>
Oct 07 04:28:02 ip-172-31-45-71 jenkins[19189]: 2025-10-07 04:28:02.990+0000 [id=49]           INFO      hudson.util.Re>
lines 1-20/20 (END)

```

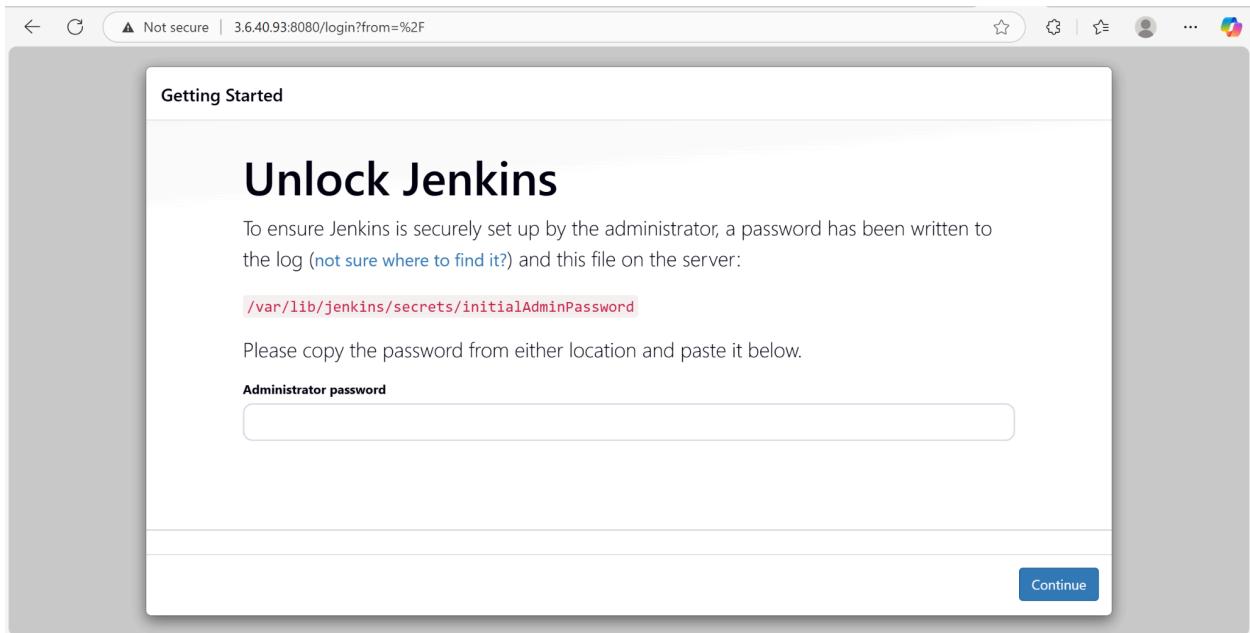
Get initial admin password:

```

ubuntu@ip-172-31-45-71:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
e0f2850e61ae4b459452f5a825e36bc3
ubuntu@ip-172-31-45-71:~$ 

```

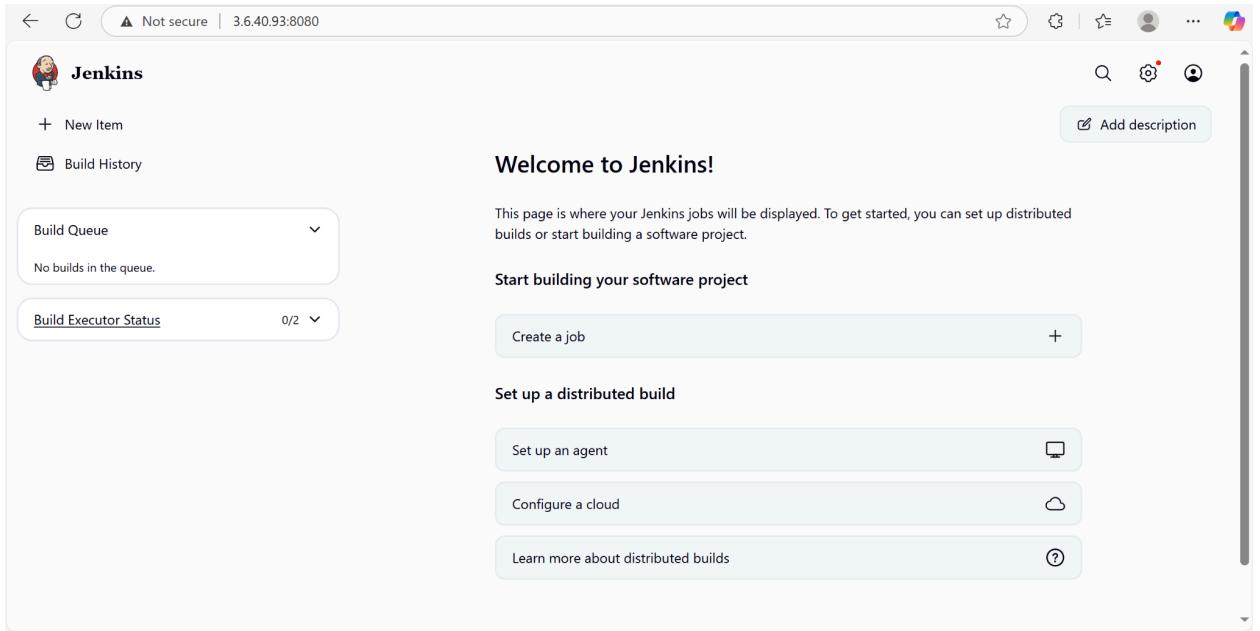
Now open ec2 and click on ipv4 address with port 8080 in browser we get this



## Install recommended Jenkins plugins (first login)

1. Visit [http://<EC2\\_PUBLIC\\_IP>:8080](http://<EC2_PUBLIC_IP>:8080) and complete initial setup with the password above.
2. When asked, select **Install suggested plugins**.
3. Ensure these are installed (Manage Jenkins → Manage Plugins → Installed):
  - Git plugin
  - Pipeline (workflow-aggregator)
  - GitHub plugin / GitHub Branch Source
  - Credentials Binding plugin
  - NodeJS plugin (optional)
  - Generic Webhook Trigger (optional but helpful)
  - Blue Ocean (optional)

(You can add others later.)



## Give Jenkins ability to deploy to `deployer`

We want Jenkins to be able to copy files and restart pm2 **as `deployer`**. Create this sudoers rule:

1. Check pm2 path for accuracy:

```
which pm2
```

```
# e.g. /usr/bin/pm2
```

2. Create sudoers entry so `jenkins` can run a few commands as `deployer` without password:

```
sudo bash -c 'echo "jenkins ALL=(deployer) NOPASSWD: /usr/bin/pm2, /usr/bin/rsync, /usr/bin/systemctl, /bin/chown, /bin/cp, /bin/mv" > /etc/sudoers.d/jenkins_deployer'
```

```
sudo chmod 0440 /etc/sudoers.d/jenkins_deployer
```

If `pm2` path differs, update `/usr/bin/pm2` accordingly.

```
ubuntu@ip-172-31-45-71:~$ which pm2  
/usr/bin/pm2
```

```
ubuntu@ip-172-31-45-71:~$ sudo bash -c 'echo "jenkins ALL=(deployer) NOPASSWD: /usr/bin/pm2, /usr/bin/rsync, /usr/bin/s  
ystemctl, /bin/chown, /bin/cp, /bin/mv" > /etc/sudoers.d/jenkins_deployer'  
ubuntu@ip-172-31-45-71:~$ sudo chmod 0440 /etc/sudoers.d/jenkins_deployer  
ubuntu@ip-172-31-45-71:~$ █
```

## Step 6 – Add deploy keys (two options) so Jenkins & deployer can access GitHub

### Option A – recommended: Use SSH keys

1. Generate SSH key for `jenkins` (on the EC2):

```
sudo -u jenkins ssh-keygen -t ed25519 -C "jenkins@$hostname" -f  
/var/lib/jenkins/.ssh/id_ed25519 -N ""  
  
sudo chown -R jenkins:jenkins /var/lib/jenkins/.ssh  
  
sudo chmod 700 /var/lib/jenkins/.ssh  
  
sudo cat /var/lib/jenkins/.ssh/id_ed25519.pub
```

2. Copy the printed public key and add it on GitHub:

- For each repo → Settings → Deploy keys → **Add deploy key** and allow **Write access** if Jenkins needs to push. For read-only pulls, write is not needed.

3. Generate SSH key for `deployer` (used if you want `deployer` to clone directly):

```
sudo -u deployer ssh-keygen -t ed25519 -C "deployer@$hostname"  
-f /home/deployer/.ssh/id_ed25519 -N ""
```

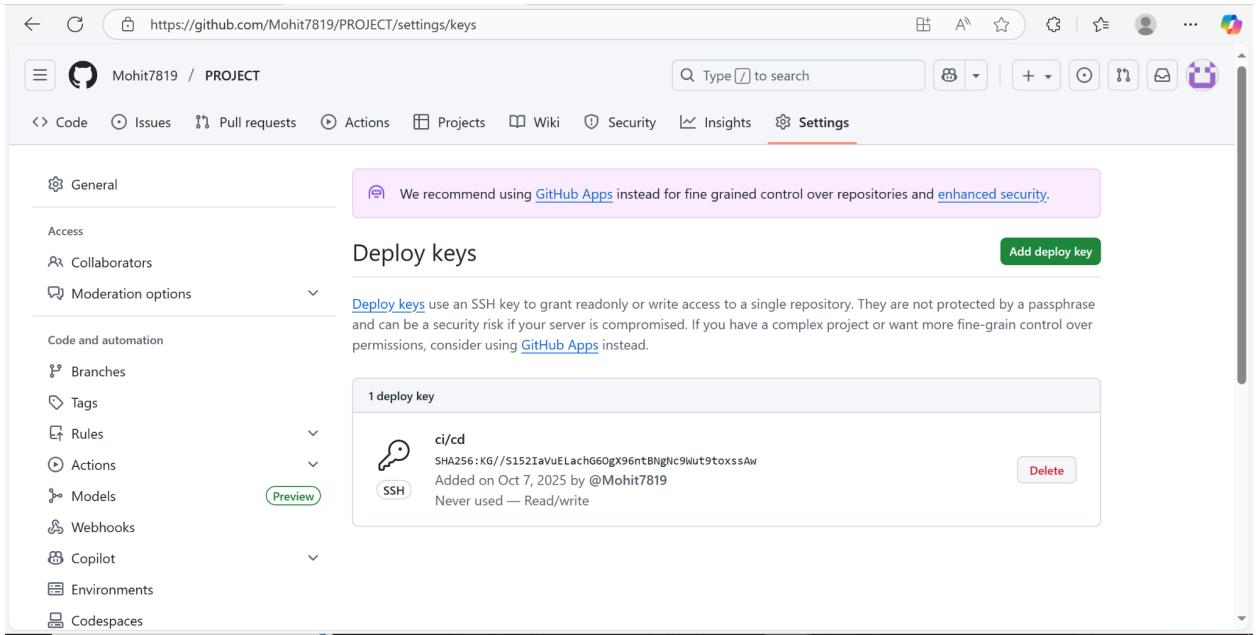
```
sudo cat /home/deployer/.ssh/id_ed25519.pub

# Add deployer's public key to GitHub repository (as deploy key
or to your GitHub account)
```

### Option B – Use GitHub Personal Access Token (PAT)

If using HTTPS with PAT, save PAT in Jenkins Credentials as "Secret text" or "Username/password" (username = your GitHub user, password = PAT). I recommend SSH keys.

```
ubuntu@ip-172-31-45-71:~$ sudo -u jenkins ssh-keygen -t ed25519 -C "jenkins@$hostname" -f /var/lib/jenkins/.ssh/id_ed25519 -N ""
Generating public/private ed25519 key pair.
Created directory '/var/lib/jenkins/.ssh'.
Your identification has been saved in /var/lib/jenkins/.ssh/id_ed25519
Your public key has been saved in /var/lib/jenkins/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:KG//S152IaVuELachG60gX96ntBNgNc9Wut9toxssAw jenkins@ip-172-31-45-71
The key's randomart image is:
+--[ED25519 256]--+
|          . o .
|         . + = + .
|        . + = * =
|       o o S B + .
|      + * E = o .
|     * + = 0 o o |
|    . =.+ * .ooo.
|   .o+.++..o o |
+---[SHA256]---+
ubuntu@ip-172-31-45-71:~$ sudo chown -R jenkins:jenkins /var/lib/jenkins/.ssh
ubuntu@ip-172-31-45-71:~$ sudo chmod 700 /var/lib/jenkins/.ssh
ubuntu@ip-172-31-45-71:~$ sudo cat /var/lib/jenkins/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGPdbxoKSA447HcQ1h/B4wXXv/dkx8mH7CEVlGxNtqAs jenkins@ip-172-31-45-71
ubuntu@ip-172-31-45-71:~$
```



## Step 7 – Prepare an initial manual deploy (one-time) for both apps under deployer

This ensures pm2 process is created and systemd startup is configured.

### Flask (example uses Gunicorn and a `wsgi.py / wsgi:app`)

```
# as deployer

sudo -u deployer -i bash <<'EOS'

cd /home/deployer

git clone git@github.com:YOU/flask-repo.git flask-app

cd flask-app

python3 -m venv venv

. venv/bin/activate

pip install -r requirements.txt

# Start via gunicorn (adjust module name if different)
```

```
pm2 start gunicorn --name flask-app --bind 0.0.0.0:5000
  "wsgi:app"

pm2 save

# Setup pm2 systemd startup for deployer

pm2 startup systemd -u deployer --hp /home/deployer

# The last command prints a sudo command – run it:

sudo env PATH=$PATH:/usr/bin pm2 startup systemd -u deployer --hp
/home/deployer

sudo systemctl enable pm2-deployer

EOS
```

- Confirm app: `curl http://localhost:5000` (on server) or `http://<EC2_PUBLIC_IP>:5000` from browser (open security group port 5000).

## Express

```
sudo -u deployer -i bash <<'EOS'
cd /home/deployer
git clone git@github.com:YOU/express-repo.git express-app
cd express-app
npm ci
pm2 start npm --name express-app -- start
pm2 save
pm2 startup systemd -u deployer --hp /home/deployer
```

```
sudo env PATH=$PATH:/usr/bin pm2 startup systemd -u deployer --hp /home/deployer
```

```
sudo systemctl enable pm2-deployer
```

EOS

- Confirm: curl http://localhost:3000 or http://<EC2\_PUBLIC\_IP>:3000.

(If you prefer **nginx** as reverse proxy: install nginx and proxy ports 80 -> 3000 and 5000. I can give nginx config if you want.)

```
(venv) deployer@ip-172-31-45-71:~/project/backend$ pm2 start "gunicorn --bind 0.0.0.0:5000 app:app" --name flask-app
[PM2] Starting /usr/bin/bash in fork_mode (1 instance)
[PM2] Done.



| id | name        | mode |    | status  | cpu | memory |
|----|-------------|------|----|---------|-----|--------|
| 1  | express-app | fork | 0  | online  | 0%  | 59.8mb |
| 0  | flask-app   | fork | 15 | errored | 0%  | 0b     |
| 2  | flask-app   | fork | 0  | online  | 0%  | 6.9mb  |



(venv) deployer@ip-172-31-45-71:~/project/backend$ pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /home/deployer/.pm2/dump.pm2
(venv) deployer@ip-172-31-45-71:~/project/backend$ 
(venv) deployer@ip-172-31-45-71:~/project/backend$ curl http://localhost:5000
{"message": "Flask backend is running!"}
(venv) deployer@ip-172-31-45-71:~/project/backend$
```

Now open the ec2\_public\_ip:3000 we get

The screenshot shows a web browser window with the following content:

Address bar: Not secure | 3.6.40.93:3000

# Send Data to Flask Backend

Name:

Email:

Fill the form and click on submit button

The screenshot shows a web browser window with the following details:

- Address bar: Not secure | 3.6.40.93:3000/submit-form
- Title: Flask Backend Response
- Content:
  - Message:** Data received successfully!
  - Name:** cicd
  - Email:** cicd@gmail.com
- Link: [Go back](#)

**Step 8 – Put Jenkinsfile in each repo (I give exact files to copy & commit)**

Flask Jenkinsfile (save to repo root as Jenkinsfile)

```
pipeline {  
    agent any  
    environment {  
        APP_DIR = '/home/deployer/flask-app'  
        PM2_NAME = 'flask-app'
```

```
        GIT_BRANCH = 'main' // change if needed

    }

    stages {

        stage('Checkout') {

            steps {

                // For "Pipeline script from SCM" Jenkinsfile will
                already be checked out.

                // If Jenkinsfile is configured separately, use:

                // git branch: "${env.GIT_BRANCH}", url:
                'git@github.com:YOU/flask-repo.git', credentialsId:
                'GIT_SSH_CREDENTIALS_ID'

                echo "source code checked out by pipeline"

            }

        }

        stage('Install deps & Test') {

            steps {

                sh '''

                    set -e

                    python3 -m venv venv

                    . venv/bin/activate

                    pip install -r requirements.txt

                    # run tests (optional)

                    pytest || true

                '''



            }

        }

    }

}
```

```
    ...
}

}

stage('Deploy to server') {

    steps {

        sh '''

            set -e

            # sync workspace to deploy directory as deployer
            (jenkins allowed to run rsync as deployer via sudo)

            sudo -u deployer rsync -av --delete ${WORKSPACE}/
            ${APP_DIR}/

            # install server-side deps and restart app under
            deployer

            sudo -u deployer bash -lc "cd ${APP_DIR} && .
            venv/bin/activate && pip install -r requirements.txt && pm2
            restart ${PM2_NAME} || pm2 start gunicorn --name ${PM2_NAME}
            --bind 0.0.0.0:5000 'wsgi:app'"

            ...
    }

}

post {

    success { echo "Flask deploy succeeded" }
}
```

```
failure { echo "Flask deploy failed" }

}

}
```

Express Jenkinsfile (save to repo root as Jenkinsfile)

```
pipeline {

    agent any

    environment {
        APP_DIR = '/home/deployer/express-app'
        PM2_NAME = 'express-app'
        GIT_BRANCH = 'main'
    }

    stages {
        stage('Checkout') {
            steps { echo "code checked out" }

        }

        stage('Install & Test') {
            steps {
                sh '''
                    set -e
                    # install node deps inside workspace
                '''
            }
        }
    }
}
```

```
    npm ci

    # optional tests

    npm test || true
    ...

}

}

stage('Deploy') {

    steps {

        sh '''

        set -e

        sudo -u deployer rsync -av --delete ${WORKSPACE}/
${APP_DIR}/

        sudo -u deployer bash -lc "cd ${APP_DIR} && npm ci &&
pm2 restart ${PM2_NAME} || pm2 start npm --name ${PM2_NAME} --
start"

        ...

    }

}

post {

    success { echo "Express deploy succeeded" }

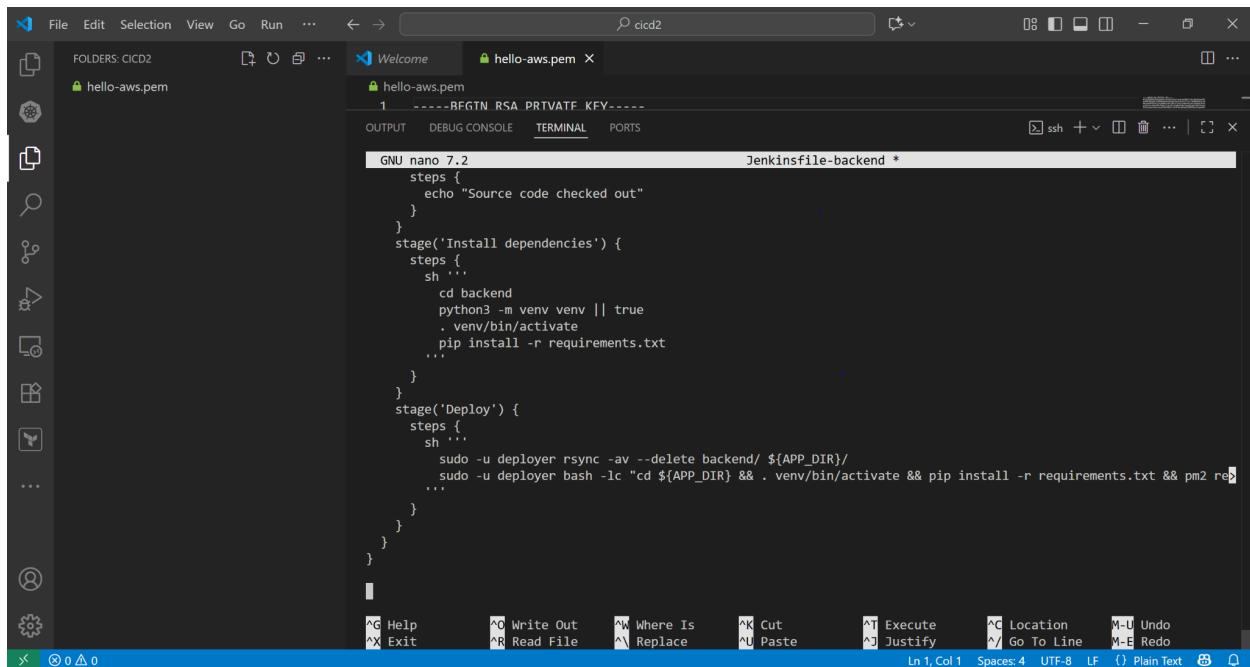
    failure { echo "Express deploy failed" }

}
```

```
}
```

## Important:

- Commit the appropriate Jenkinsfile to each repo root (same file name `Jenkinsfile`).
- Replace `YOU` and branch names as needed.
- `credentialsId` is only needed if Jenkins must authenticate to git; if you used the Jenkins SSH deploy key (plugin), checkout can use `git` step without credentials.



The screenshot shows a terminal window titled "cicd2" with a sidebar containing icons for FOLDERS, Selection, View, Go, Run, etc. The main area displays a Jenkinsfile named "Jenkinsfile-backend". The file content is as follows:

```
steps {
    echo "Source code checked out"
}
stage('Install dependencies') {
    steps {
        sh '''
            cd backend
            python3 -m venv venv || true
            . venv/bin/activate
            pip install -r requirements.txt
        '''
    }
}
stage('Deploy') {
    steps {
        sh '''
            sudo -u deployer rsync -av --delete backend/ ${APP_DIR}/
            sudo -u deployer bash -lc "cd ${APP_DIR} && . venv/bin/activate && pip install -r requirements.txt && pm2 restart all"
        '''
    }
}
```

The terminal window also shows various keyboard shortcuts at the bottom, such as Help, Write Out, Read File, Cut, Paste, Execute, Justify, Location, Undo, and Redo.

## Creating two files

```
(venv) deployer@ip-172-31-45-71:~/project$ cd ~/project
(venv) deployer@ip-172-31-45-71:~/project$ nano Jenkinsfile-backend
(venv) deployer@ip-172-31-45-71:~/project$ nano Jenkinsfile-frontend
(venv) deployer@ip-172-31-45-71:~/project$ (venv) deployer@ip-172-31-45-71:~/project$
```

```
(venv) deployer@ip-172-31-45-71:~/project$ git remote -v
origin https://github.com/Mohit7819/PROJECT.git (fetch)
origin https://github.com/Mohit7819/PROJECT.git (push)
(venv) deployer@ip-172-31-45-71:~/project$ git config --global credential.helper store
(venv) deployer@ip-172-31-45-71:~/project$ git push origin main
Username for 'https://github.com': Mohit7819
Password for 'https://Mohit7819@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 840 bytes | 840.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Mohit7819/PROJECT.git
  0621309..3805b1c main -> main
(venv) deployer@ip-172-31-45-71:~/project$
```

## Create Jenkins Pipeline job (point to the repo `Jenkinsfile`)

1. Jenkins UI → **New Item** → give name `flask-pipeline` → **Multibranch Pipeline** or **Pipeline**.
  - Easiest: choose **Pipeline**, then under **Definition** choose **Pipeline script from SCM** → SCM: **Git** → Repository URL `git@github.com:YOU/flask-repo.git` → Credentials: choose the SSH credentials you added → Branch: `main`.
  - Jenkins will pull the `Jenkinsfile` from the repo and run it.
2. Repeat for `express-pipeline`.

On job configuration, enable **GitHub hook trigger for GITScm polling** (checkbox) or use Generic Webhook Trigger plugin.

The screenshot shows the Jenkins home page. At the top, there's a header bar with a back arrow, a refresh button, and a URL indicator 'Not secure | 3.6.40.93:8080'. To the right of the URL are several icons: a star, a gear, a magnifying glass, a person, and a three-dot menu. Below the header, the Jenkins logo is displayed, followed by a 'New Item' button and a 'Build History' link. A sidebar on the left contains 'Build Queue' (with a dropdown arrow) and 'No builds in the queue.' below it, and a 'Build Executor Status' section showing '0/2' with a dropdown arrow. On the right, the main content area features a 'Welcome to Jenkins!' heading, a brief introduction about managing jobs, and a 'Start building your software project' button. Below this are sections for 'Create a job' (with a '+' icon), 'Set up a distributed build' (with three sub-options: 'Set up an agent' with a monitor icon, 'Configure a cloud' with a cloud icon, and 'Learn more about distributed builds' with a question mark icon).

Not secure | 3.6.40.93:8080

Jenkins

+ New Item

Build History

Build Queue

No builds in the queue.

Build Executor Status

0/2

Create a job +

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build

- Set up an agent
- Configure a cloud
- Learn more about distributed builds

## Click on new item

The screenshot shows the Jenkins 'Configure' screen for a GitHub project pipeline. The pipeline is defined by a Pipeline script from SCM (Git) and branches to build 'main'. The pipeline syntax is Jenkinsfile backend, and lightweight checkout is selected.

**Configure**

**General**

**Triggers**

**Pipeline**

**Advanced**

**GitHub-project**

Project url: <https://github.com/Mohit7819/PROJECT.git>

**Triggers**

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM
- Trigger builds remotely (e.g., from scripts)

**Pipeline**

Define your Pipeline using Groovy directly or pull it from source control.

**Definition**

Pipeline script from SCM

**SCM**

Git

**Repositories**

Repository URL: <https://github.com/Mohit7819/PROJECT.git>

Credentials: none

**Branches to build**

Branch Specifier (blank for 'any'): main

**Repository browser**

(Auto)

**Additional Behaviours**

+ Add

**Script Path**

Jenkinsfile backend

Lightweight checkout

**Pipeline Syntax**

**Advanced**

Advanced

**Buttons**

Save Apply

The screenshot shows a web browser window with three tabs open:

- Untitled document - Google Docs
- Launch an instance | EC2 | ap-sou
- flask-deploy #2 - Jenkins

The Jenkins tab displays the build history for "flask-deploy #2" (Oct 7, 2025, 10:19:13 AM). It shows the build was started by user MOHIT SINGH and took 11 seconds. The build details include a git commit (revision: 209342b30221140d7cd3d4efc3e189ac42d9cd45, repository: https://github.com/Mohit7819/PROJECT.git) and a change log entry: "1. Update requirements.txt with gunicorn & fix Jenkinsfile (details / githubweb)". The Jenkins version is 2.530.

```
^C
(venv) jenkins@ip-172-31-45-71:~/workspace/flask-deploy$ pm2 list
    id  name        mode   ↻  status    cpu      memory
  0   flask-app   fork    0  online   0%  23.7mb
(venv) jenkins@ip-172-31-45-71:~/workspace/flask-deploy$
```

When go to browser we see our backend is running

The screenshot shows a web browser window with three tabs open:

- Instances | EC2 | ap-south-1
- flask-deploy #3 Console - Jenkins
- 65.2.166.237:5000

The Jenkins tab displays the console output for "flask-deploy #3". The output shows the message: {"message":"Flask backend is running!"}.

## Same steps for express-deploy jenkins click on new item

The screenshot shows the Jenkins dashboard. At the top, there are three tabs: 'Instances | EC2 | ap-south-1', 'Dashboard - Jenkins', and a selected tab '65.2.166.237:5000'. Below the tabs is a header with a Jenkins logo, a search bar, and various icons. The main content area is titled 'Jenkins' and shows a 'Build History' section. A table lists one build item:

S	W	Name ↓	Last Success	Last Failure	Last Duration
Green circle	Cloud icon	express-deploy	12 min #3	45 min #1	8.7 sec

Below the table are sections for 'Build Queue' (empty) and 'Build Executor Status' (0/2). At the bottom, there are icons for 'S' (Status), 'M' (Metrics), and 'L' (Logs).

The screenshot shows the details of a specific Jenkins job. The URL is '65.2.166.237:8080/job/express-deploy/2'. The page has a left sidebar with options like Status, Changes, Console Output, and Git Build Data. The main content area shows the build summary for '#2 (Oct 7, 2025, 11:10:07 AM)'. It includes information about the user who started it (MOHIT SINGH), the duration (Started 11 sec ago, Took 6.6 sec), and the git revision (3d39f8ae9e4897f0d0db68fbabada7a639bf64e4c, Repository: https://github.com/Mohit7819/PROJECT.git, refs/remotes/origin/main). There is also a note about using npm install instead of npm ci for express deploy.

The screenshot shows the Jenkins dashboard again. The 'Build History' table now lists two items:

S	W	Name ↓	Last Success	Last Failure	Last Duration
Green circle	Cloud icon	express-deploy	59 sec #2	8 min 19 sec #1	6.6 sec
Green circle	Cloud icon	flask-deploy	29 min #3	1 hr 2 min #1	8.7 sec

The rest of the interface is identical to the first screenshot, including the 'Build Queue' and 'Build Executor Status' sections at the bottom.

Now open <http://65.2.166.237:3000> in browser we see



## Send Data to Flask Backend

Name:

Email:

Fill form and click on submit button

← → ⌛ ⚠ Not secure 65.2.166.237:3000

# Send Data to Flask Backend

Name:

Email:

← → ⌛ ⚠ Not secure 65.2.166.237:3000/submit-form

# Flask Backend Response

**Message:** Data received successfully!

**Name:** pipeline

**Email:** pipeline@gmail.com

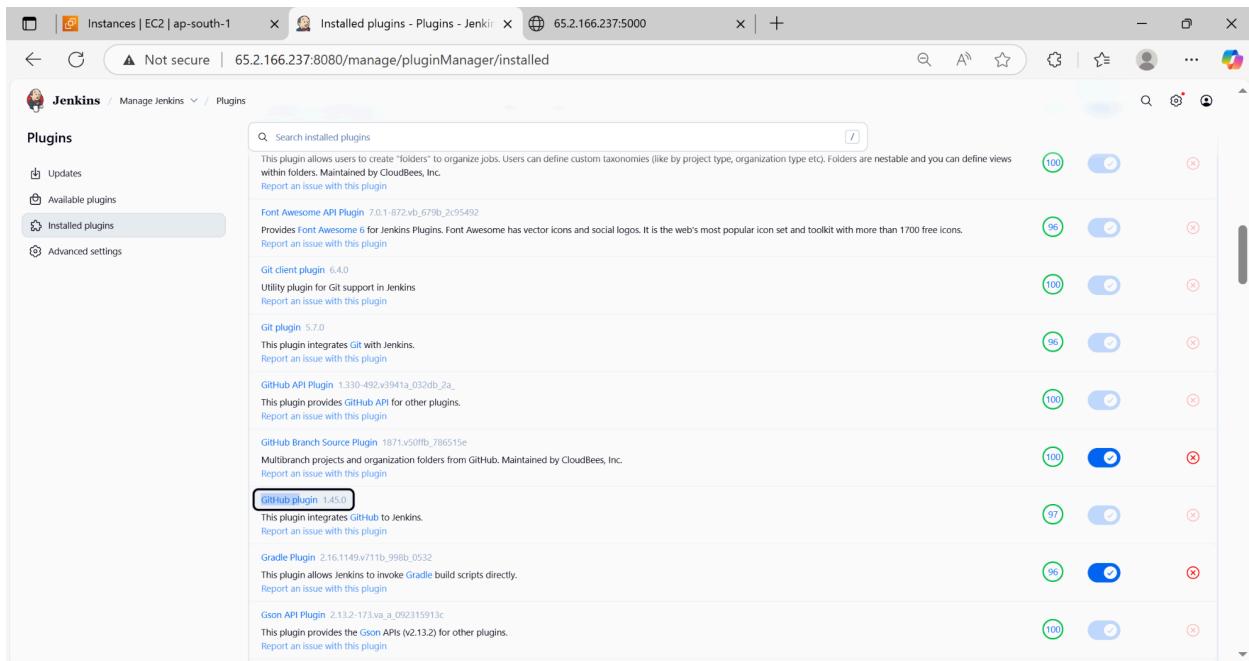
[Go back](#)

Add GitHub webhook (so push triggers Jenkins)

On each GitHub repo:

- Settings → Webhooks → Add webhook
  - Payload URL: `http://<EC2_PUBLIC_IP>:8080/github-webhook/`
  - Content type: `application/json`
  - Secret: optional (if you use it, configure the same secret in Jenkins GitHub plugin)
  - Which events: **Just the push event**
- Save.

Now a `git push` to `main` should send webhook to Jenkins and trigger the pipeline.

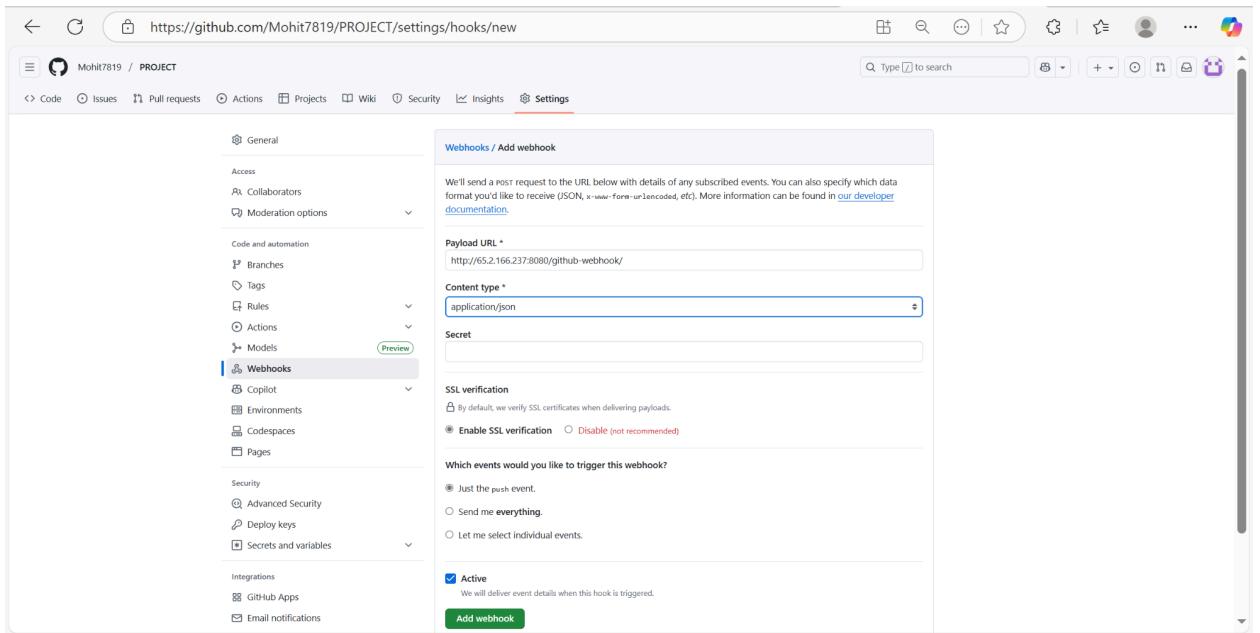


Click on config > triggers

The screenshot shows the Jenkins configuration interface for a job named 'flask-deploy'. The 'Triggers' section is selected, displaying options like 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling' (which is checked), and 'Poll SCM'. Below this is the 'Pipeline' section, which contains a 'Definition' field set to 'Pipeline script from SCM'. A dropdown menu for 'SCM' is open, showing 'Git' as the selected option. Under 'Git', there is a 'Repository URL' field containing 'https://github.com/Mohit7819/PROJECT.git'. The 'Credentials' field shows a placeholder 'Mohit7819\*\*\*\*\* (GitHub access via PAT)'. At the bottom are 'Save' and 'Apply' buttons.

Now open github repo and open setting click on webhooks

The screenshot shows the GitHub repository settings for 'PROJECT'. The 'Webhooks' tab is selected. On the right, there is a section titled 'Webhooks' with a sub-section 'Add webhook'. On the left, the sidebar navigation includes 'General', 'Access', 'Collaborators', 'Moderation options', 'Code and automation' (with 'Branches', 'Tags', 'Rules', 'Actions', 'Models', and 'Webhooks' listed), 'Security' (with 'Advanced Security', 'Deploy keys', and 'Secrets and variables'), and 'Integrations' (with 'GitHub Apps' and 'Email notifications').



## Now test webhooks

### Creating a new file and push to git repo

A screenshot of a terminal window in a code editor (VS Code). The terminal tab is active and shows a PowerShell session (PS C:\Users\dell\Downloads\cicd2). The user has run several commands to test a webhook: `echo "webhook test" >> test.txt`, `git add test.txt`, `git commit -m "Webhook test"`, and `git push origin main`. The output shows the commit message, author information (Committer: Ubuntu <ubuntu@ip-172-31-45-71.ap-south-1.compute.internal>), and the push process. The commit hash 3d39fa8... is shown at the end. The status bar at the bottom indicates the terminal has 1 line, 0 columns, and is in Plain Text mode.

The image shows two screenshots side-by-side. The left screenshot is from GitHub's 'PROJECT/settings/hooks' page, specifically the 'Webhooks' section. It displays a successful webhook configuration for the URL `http://65.2.166.237:8080/github-webhook/push`. The right screenshot is from a Jenkins instance, showing the 'Build History' page at `65.2.166.237:8080/view/all/builds`. The Jenkins interface lists several build entries, including successful builds for 'flask-deploy' (#4, #3, #2) and 'express-deploy' (#3, #2), and failed builds for 'express-deploy' (#1) and 'flask-deploy' (#1). Both screenshots are taken in a browser window.

**GitHub Webhooks Settings:**

- General
- Access
- Collaborators
- Moderation options
- Code and automation
  - Branches
  - Tags
  - Rules
  - Actions
  - Models
- Webhooks** (selected)
- Copilot
- Environments
- Codespaces
- Pages
- Security
  - Advanced Security
  - Deploy keys
  - Secrets and variables
- Integrations
  - Github Apps
  - Email notifications

**Jenkins Build History:**

S	Build	Time Since	Status
✓	flask-deploy #4	5 min 31 sec	stable
✓	express-deploy #3	5 min 31 sec	stable
✓	express-deploy #2	36 min	back to normal
✗	express-deploy #1	43 min	broken since this build
✓	flask-deploy #3	1 hr 4 min	stable
✓	flask-deploy #2	1 hr 27 min	back to normal
✗	flask-deploy #1	1 hr 37 min	broken since this build

The screenshot shows a terminal window within a code editor interface. The terminal tab is active, displaying Jenkins logs. The logs indicate the successful cloning of a repository from GitHub, the startup of the Jenkins service, and various log entries related to the Jenkins daemon and its deployment process. The terminal window includes standard Linux-style navigation keys at the top and a status bar at the bottom.

```
PS C:\Users\dell\Downloads\cicd2>
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 300 bytes | 300.00 Kib/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Mohit7819/PROJECT.git
    3d39fa8..ca0de66 main -> main
ubuntu@ip-172-31-45-71:~/PROJECT$ sudo systemctl status jenkins --no-pager
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-10-07 08:34:29 UTC; 3h 14min ago
       Main PID: 545 (java)
          Tasks: 65 (limit: 2213)
            Memory: 990.9M (peak: 1.0G)
              CPU: 3min 57.445s
             CGroup: /system.slice/jenkins.service
                     └─ 545 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache...
1603 "PM2 v6.0.13: God Daemon (/home/deployer/.pm2)"
```

Oct 07 09:26:44 ip-172-31-45-71 sudo[2408]: pam\_unix(sudo:session): session closed for user deployer  
Oct 07 09:26:44 ip-172-31-45-71 sudo[2413]: jenkins : PWD=/var/lib/jenkins/workspace/flask-pipeline ; USER=deployer  
Oct 07 09:26:44 ip-172-31-45-71 sudo[2413]: pam\_unix(sudo:session): session opened for user deployer(uid=111)  
Oct 07 09:26:48 ip-172-31-45-71 sudo[2413]: pam\_unix(sudo:session): session closed for user deployer  
Oct 07 11:34:41 ip-172-31-45-71 jenkins[545]: 2025-10-07 11:34:41.666+0000 [id=781] INFO o...JECT>!  
Oct 07 11:40:50 ip-172-31-45-71 jenkins[545]: 2025-10-07 11:40:50.537+0000 [id=781] INFO o...ebhook/  
Oct 07 11:40:50 ip-172-31-45-71 jenkins[545]: 2025-10-07 11:40:50.554+0000 [id=781] INFO o...-deploy  
Oct 07 11:40:50 ip-172-31-45-71 jenkins[545]: 2025-10-07 11:40:50.561+0000 [id=781] INFO o...-deploy  
Oct 07 11:40:51 ip-172-31-45-71 jenkins[545]: 2025-10-07 11:40:51.366+0000 [id=1634] INFO c...ring #3  
Oct 07 11:40:51 ip-172-31-45-71 jenkins[545]: 2025-10-07 11:40:51.367+0000 [id=1635] INFO c...ring #4  
Hint: Some lines were ellipsized, use -l to show in full.  
ubuntu@ip-172-31-45-71:~/PROJECT\$