

DEVOPS ASSIGNMENT 2

DAY 2

STEP1 :

The command `sudo apt update`, which updates the package lists for available updates and repositories on an Ubuntu system. The output includes information about the sources being updated, including the Jenkins repository and various components from the Ubuntu archives.

```
poojz@zZz:~/devops$ sudo apt update
[sudo] password for poojz:
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:4 http://archive.ubuntu.com/ubuntu noble InRelease
Get:5 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [364 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:11 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:12 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:13 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [20.0 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:15 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:16 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [9004 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.0 kB]
```

STEP 2:

The terminal output indicates that the user attempted to install Docker on Ubuntu using `'sudo apt install -y docker.io'`. It shows that Docker is already at the newest version, with no upgrades available for other packages.

```
poojz@zZz:~$ sudo apt install -y docker.io
[sudo] password for poojz:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (26.1.3-0ubuntu1~24.04.1).
0 upgraded, 0 newly installed, 0 to remove and 130 not upgraded.
```

STEP 3 :

The commands enabling and starting the Docker service (`systemctl enable/start docker`) and verifying the installation using `docker --version`, confirming Docker 26.1.3 on Ubuntu 24.04.

```
poojz@zZz:~$ sudo systemctl enable docker
poojz@zZz:~$ sudo systemctl start docker

poojz@zZz:~/devops$ docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
```

STEP 4 :

The terminal command using `curl` to download the latest Docker Compose binary from GitHub and save it to `/usr/local/bin/docker-compose`. The progress bar indicates a successful download of **71.4MB** at a speed of **756kB/s**.

```
poojz@zZz:~$ sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr
/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed
  0     0    0     0    0     0      0      0  --:--:-- --:--:-- --:--:--    0
  0     0    0     0    0     0      0      0  --:--:-- --:--:-- --:--:--    0
100 71.4M 100 71.4M    0     0  526k    0  0:02:18 0:02:18 --:--:-- 756k
```

STEP 5 :

The commands making Docker Compose executable (`chmod +x`), verifying its installation (`docker-compose --version`), creating a `~/devops` directory, and navigating into it.

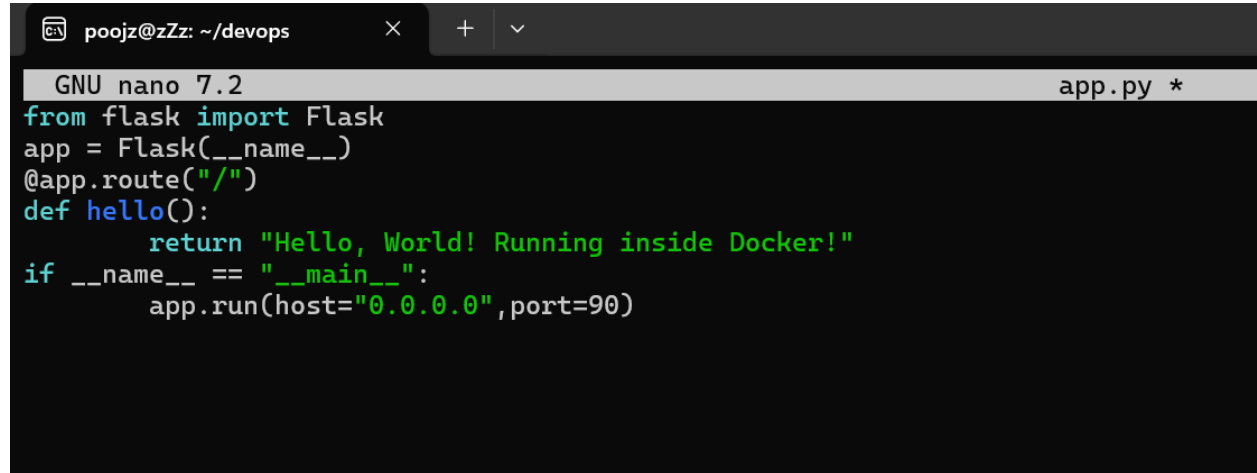
```
poojz@zZz:~$ sudo chmod +x /usr/local/bin/docker-compose
poojz@zZz:~$ docker-compose --version
Docker Compose version v2.34.0

poojz@zZz:~$ mkdir ~/devops
poojz@zZz:~$ cd ~/devops
```

STEP 6 :

The Flask application (`app.py`) being created using the `nano` editor, defining a simple web server that returns "Hello, World! Running inside Docker!" on port 90.

```
poojz@zZz:~/devops$ nano app.py
```



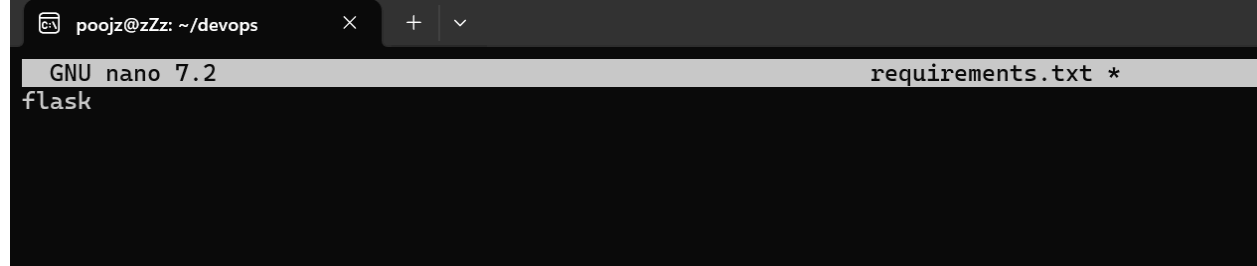
The screenshot shows a terminal window with the nano editor open. The editor's title bar indicates the file is `app.py`. The code inside the editor is as follows:

```
GNU nano 7.2 app.py *
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Hello, World! Running inside Docker!"
if __name__ == "__main__":
    app.run(host="0.0.0.0",port=90)
```

STEP 7:

The creation of a `requirements.txt` file using the `nano` editor, listing `flask` as a dependency for the Python project. Note: There's a typo in the filename (`requiremnts.txt`).

```
poojz@zZz:~/devops$ nano requiremnts.txt
```



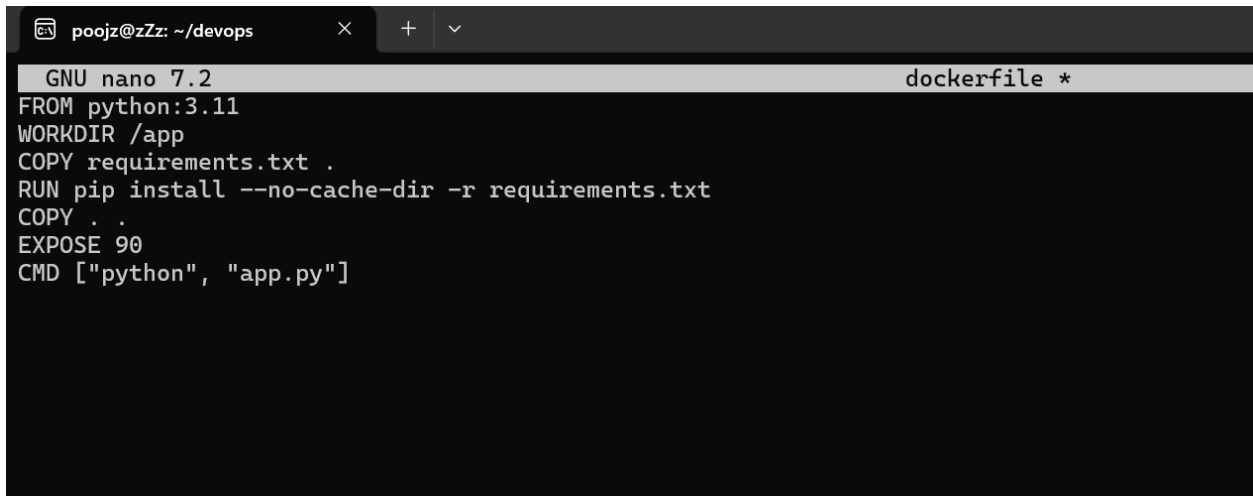
The screenshot shows a terminal window with the nano editor open. The editor's title bar indicates the file is `requirements.txt`. The code inside the editor is as follows:

```
GNU nano 7.2 requirements.txt *
flask
```

STEP 8:

The `Dockerfile` being created using `nano`, defining a containerized environment for a Python 3.11 Flask app by copying dependencies, installing them, exposing port 90, and running `app.py`.

```
poojz@zZz:~/devops$ nano dockerfile
```

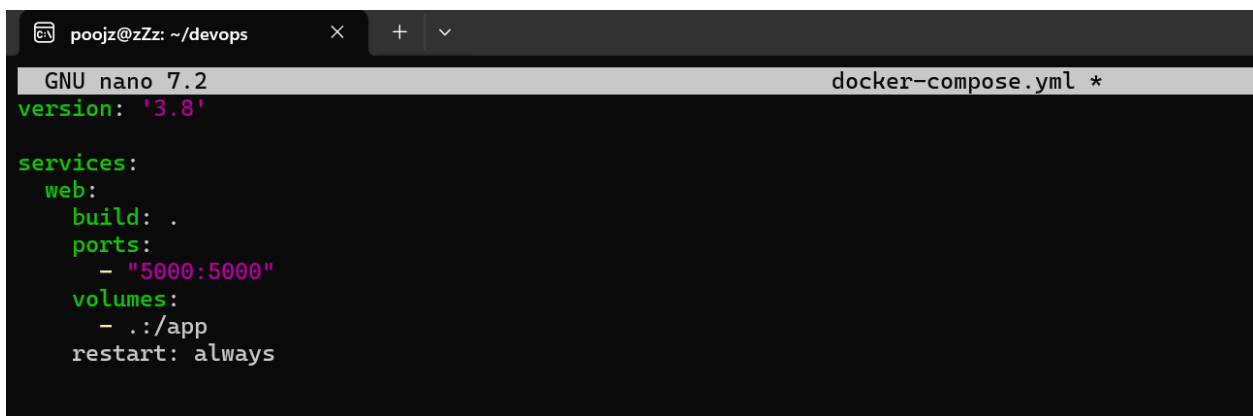


```
GNU nano 7.2 dockerfile *
FROM python:3.11
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 90
CMD ["python", "app.py"]
```

STEP 9 :

The `docker-compose.yml` file being created using `nano`, defining a service named `web` that builds from the current directory, maps port `5000:5000`, mounts a volume, and restarts automatically.

```
poojz@zZz:~/devops$ nano docker-compose.yml
```



```
GNU nano 7.2 docker-compose.yml *
version: '3.8'

services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - ./app
    restart: always
```

STEP 10 :

The image shows the output of `sudo docker images`, listing three Docker images: `test` (1.03GB, created 20 hours ago), `nginx` (192MB, 5 weeks old), and `python:3.11` (1.01GB, 3 months old)

```
poojz@zZz:~/devops$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
test          latest    b597cf24bcd9   20 hours ago   1.03GB
nginx         latest    53a18edff809   5 weeks ago    192MB
python        3.11      18c0f2265fd9   3 months ago   1.01GB
```

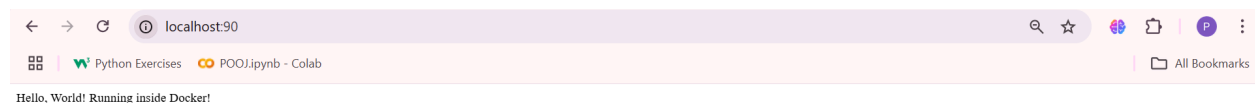
STEP 11 :

The image shows the execution of `sudo docker-compose up --build`, where Docker Compose is building an image from a `docker-compose.yml` file. A warning indicates that the `version` attribute is obsolete, and the build steps confirm that the required files and dependencies are successfully cached.

```
poojz@zZz:~/devops$ sudo docker-compose up --build
WARN[0000] /home/poojz/devops/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid
potential confusion
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 0.2s (11/11) FINISHED
=> [web internal] load build definition from dockerfile                                docker:default 0.0s
=> => transferring dockerfile: 188B                                                  0.0s
=> [web internal] load metadata for docker.io/library/python:3.11                    0.0s
=> [web internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                        0.0s
=> [web 1/5] FROM docker.io/library/python:3.11                                     0.0s
=> [web internal] load build context                                                  0.0s
=> => transferring context: 164B                                                      0.0s
=> CACHED [web 2/5] WORKDIR /app                                                      0.0s
=> CACHED [web 3/5] COPY requirements.txt .                                           0.0s
=> CACHED [web 4/5] RUN pip install --no-cache-dir -r requirements.txt                0.0s
=> CACHED [web 5/5] COPY . .                                                          0.0s
```

OUTPUT:

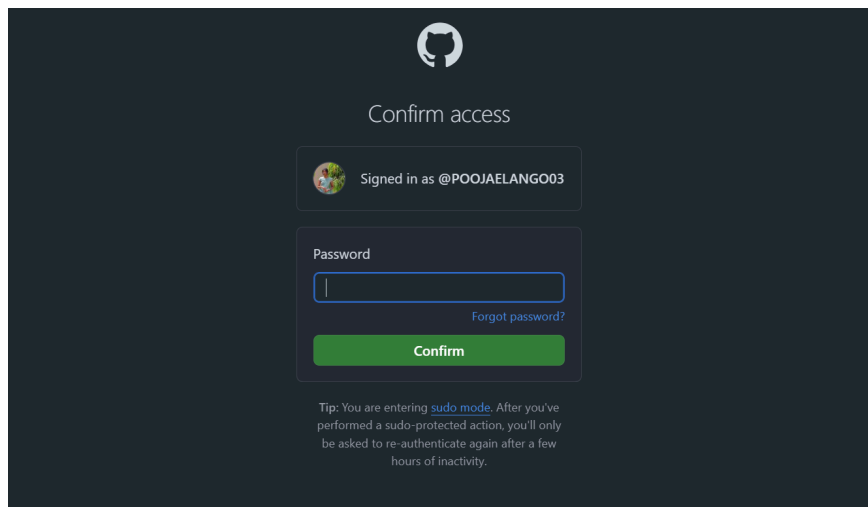
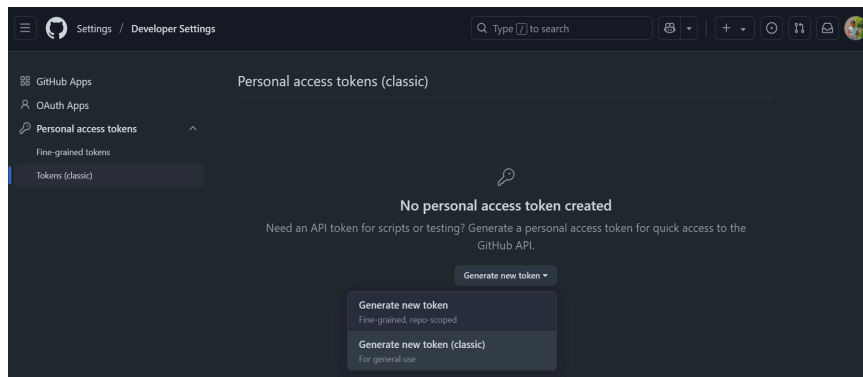
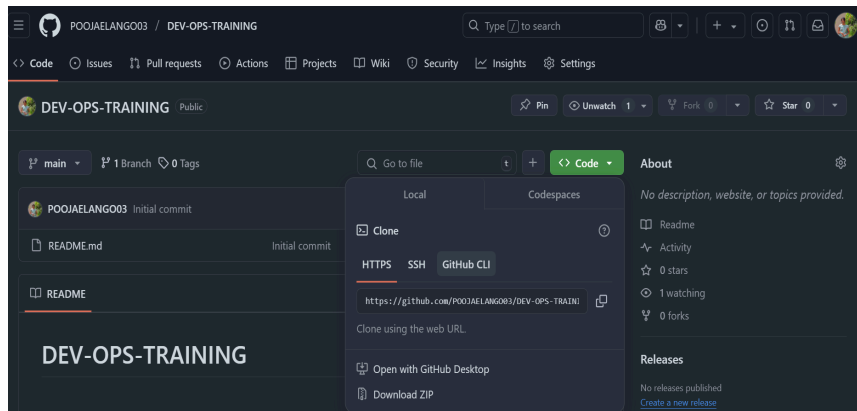
The final output shows a web browser displaying the message "Hello, World! Running inside Docker!" at `localhost:90`, confirming that the Flask application successfully runs inside a Docker container. This validates the containerization process, including building the Docker image, running the container, and exposing the correct port.



The screenshot shows a web browser window with the address bar set to `localhost:90`. The page content displays the message "Hello, World! Running inside Docker!". The browser's address bar includes navigation icons (back, forward, refresh) and a search icon. The page title is "Python Exercises" and the URL is "POOJipynb - Colab". The browser's bookmark bar is visible at the bottom, showing "All Bookmarks".

DAY 3:

STEPS TO CREATE AND SETUP:



GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

Search

Type (/) to search

Profile

+

+

+

+

+

+

Avatar

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

What's this token for?

Expiration

30 days (Apr 18, 2025)

The token will expire on the selected date.

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

☐ repo

☐ repo:status

☐ repo_deployment

☐ public_repo

☐ repo:invite

Full control of private repositories

Access commit status

Access deployment status

Access public repositories

Access repository invitations

GitHub Apps

OAuth Apps

Personal access tokens

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

Search

Type (/) to search

Profile

+

+

+

+

+

+

Avatar

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

DAY 3 DEVOPS TRAINING

What's this token for?

Expiration

30 days (Apr 18, 2025)

The token will expire on the selected date.

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

☐ repo

☐ repo:status

☐ repo_deployment

☐ public_repo

☐ repo:invite

Full control of private repositories

Access commit status

Access deployment status

Access public repositories

Access repository invitations

Settings / Developer Settings

Search

Profile

+

+

+

+

+

+

Avatar

Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

Personal access tokens (classic)

Generate new token

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

ghp_NlCA0BtlwS8qDr9bVVECKSdVlSrVPYz1F2xQf

Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

© 2025 GitHub, Inc.

Terms

Privacy

Security

Status

Docs

Contact

Manage cookies

Do not share my personal information

STARTING AND VERIFYING JENKINS SERVICE ON LINUX:

The terminal session showing the process of managing a Jenkins server on a Linux system using 'systemctl', a command-line tool for controlling the systemd system and service manager.

1. Enable Jenkins:

- Command: `sudo systemctl enable jenkins`
- This command enables the Jenkins service to start automatically at boot.

2. Start Jenkins:

- Command: `sudo systemctl start jenkins`
- This command starts the Jenkins service immediately.

3. Check Jenkins Status:

- Command: `sudo systemctl status jenkins`
- Displays the current status of the Jenkins service, indicating that it is "active (running)" and shows additional details about the service, including its main process ID (PID) and memory usage.

The Summary:

The session demonstrates the successful initialization and management of a Jenkins Continuous Integration server, indicating its readiness for use. The service was started successfully and is currently running smoothly. It also provides insight into the internal processes Jenkins follows during startup.

```
poojz@zZz:~$ sudo systemctl enable jenkins
[sudo] password for poojz:
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
poojz@zZz:~$ sudo systemctl start jenkins
poojz@zZz:~$ 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 Wed 2025-03-19 04:02:30 UTC; 24min ago
     Main PID: 173 (java)
       Tasks: 52 (limit: 4624)
      Memory: 393.9M ()
     CGroup: /system.slice/jenkins.service
             └─173 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins
Mar 19 04:02:26 zZz jenkins[173]: 2025-03-19 04:02:26.421+0000 [id=36] INFO jenkins.InitReactorRunner$1#o
Mar 19 04:02:26 zZz jenkins[173]: 2025-03-19 04:02:26.426+0000 [id=32] INFO jenkins.InitReactorRunner$1#o
Mar 19 04:02:27 zZz jenkins[173]: 2025-03-19 04:02:27.180+0000 [id=37] INFO h.p.b.g.GlobalTimeOutConfigur
Mar 19 04:02:29 zZz jenkins[173]: 2025-03-19 04:02:29.809+0000 [id=36] INFO jenkins.InitReactorRunner$1#o
Mar 19 04:02:29 zZz jenkins[173]: 2025-03-19 04:02:29.813+0000 [id=33] INFO jenkins.InitReactorRunner$1#o
Mar 19 04:02:29 zZz jenkins[173]: 2025-03-19 04:02:29.901+0000 [id=33] INFO jenkins.InitReactorRunner$1#o
Mar 19 04:02:29 zZz jenkins[173]: 2025-03-19 04:02:29.928+0000 [id=33] INFO jenkins.InitReactorRunner$1#o
Mar 19 04:02:29 zZz jenkins[173]: 2025-03-19 04:02:29.990+0000 [id=30] INFO jenkins.InitReactorRunner$1#o
Mar 19 04:02:30 zZz jenkins[173]: 2025-03-19 04:02:30.051+0000 [id=24] INFO hudson.lifecycle.Lifecycle#on
Mar 19 04:02:30 zZz systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
lines 1-19/19 (END)
```


NEXT STEPS TO SETUP:

Dashboard > DEVOPS DAY 2 > Configuration

Configure

General

Triggers

Pipeline

Advanced

Pipeline

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

Definition

Pipeline script from SCM

SCM ?

None

Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

Pipeline Syntax

Save

Apply

Jenkins

Search, Alerts, PoojaElango, log out

Dashboard >

New Item

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

0/2

All

+

S	W	Name	Last Success	Last Failure	Last Duration
🔄	☀️	DEVOPS DAY 2	N/A	N/A	N/A
✅	☀️	DEVOPS DAY1	25 min #1	N/A	9.3 sec

Icons: S M L

Add description

REST API

Jenkins 2.492.2

COMMITTING AND PUSHING CHANGES TO GITHUB IN A DEVOPS PROJECT:

The terminal session where the user is working with a Git repository named DEV-OPS-TRAINING. The steps performed include:

1. Listing the repository files, including Jenkinsfile, README.md, docker-compose.yml, dockerfile, and requirements.txt.
2. Editing the Jenkinsfile using **nano**.
3. Staging changes with **git add ..**
4. Committing the changes with the message "updated".
5. Pushing the changes to a remote GitHub repository using a personal access token for authentication.

The process successfully pushes updates to the main branch on GitHub.

```
poojz@zz:~/devops/DEV-OPS-TRAINING$ ls
Jenkinsfile README.md app.py docker-compose.yml dockerfile requirements.txt
poojz@zz:~/devops/DEV-OPS-TRAINING$ nano Jenkinsfile
poojz@zz:~/devops/DEV-OPS-TRAINING$ git add .
poojz@zz:~/devops/DEV-OPS-TRAINING$ git commit -m "updated"
[main 1da264c] updated
1 file changed, 1 insertion(+)
poojz@zz:~/devops/DEV-OPS-TRAINING$ git push https://POOJAELANGO03:ghp_NcA08Mw58qoDr9bVVECKsdVMSrvPYz1F2xQf@github.com/POOJAELANGO03/DEV-OPS-TRAINING.git
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 309 bytes | 30.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/POOJAELANGO03/DEV-OPS-TRAINING.git
5a43f41..1da264c  main -> main
```

The screenshot shows the GitHub web interface for the repository 'DEV-OPS-TRAINING' by user 'POOJAELANGO03'. The repository is public and has 7 commits. The 'main' branch is selected, showing a list of files: Jenkinsfile, README.md, app.py, docker-compose.yml, dockerfile, and requirements.txt. Each file has a commit history table. The 'About' section on the right indicates no description, website, or topics are provided, and there are 0 stars, 1 watcher, and 0 forks. The 'Releases' section shows no releases published.

File	Commit Message	Time Ago
Jenkinsfile	updated	5 hours ago
README.md	Initial commit	13 hours ago
app.py	first commit	10 hours ago
docker-compose.yml	first commit	10 hours ago
dockerfile	first commit	10 hours ago
requirements.txt	first commit	10 hours ago

GRANTING JENKINS DOCKER ACCESS AND RESTARTING SERVICE:

The terminal session where the user is managing Jenkins and Docker permissions. The following commands are executed:

1. **sudo usermod -aG docker jenkins** – Adds the Jenkins user to the Docker group, allowing it to run Docker commands without requiring sudo privileges.
2. **sudo systemctl restart jenkins** – Restarts the Jenkins service to apply the changes made to its user permissions.

These steps are typically done to enable Jenkins to interact with Docker seamlessly in a CI/CD pipeline.

```
poojz@zzz:~/devops/DEV-OPS-TRAINING$ sudo usermod -aG docker jenkins
poojz@zzz:~/devops/DEV-OPS-TRAINING$ sudo systemctl restart jenkins
```

JENKINS DASHBOARD OVERVIEW:

The screenshot displays the Jenkins Dashboard interface. The top navigation bar includes the Jenkins logo, a search icon, a notification bell, the user 'PoojaElango', and a 'log out' link. The breadcrumb trail shows 'Dashboard > DEVOPS DAY 2 > #14'. The left sidebar contains various management options: Status, Changes, Console Output (selected), Edit Build Information, Delete build '#14', Timings, Git Build Data, Pipeline Overview, Pipeline Console, Restart from Stage, Replay, Pipeline Steps, and Workspaces. The main content area is titled 'Console Output' and shows the execution log of a pipeline. The log starts with 'Started by user PoojaElango' and 'Obtained Jenkinsfile from git https://github.com/POOJAELANG003/DEV-OPS-TRAINING.git'. It details the pipeline stages: 'node', 'stage', and 'checkout'. The 'checkout' stage shows the selection of a Git installation and the fetching of changes from the remote repository. The log ends with the checkout of revision 'c243361c83758631713477d76cf90d91684cf5a1 (origin/main)'. Below the console output, there is a table summarizing the build history.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☁	DEVOPS DAY 2	4 hr 4 min #14	4 hr 13 min #13	1 min 8 sec
✓	☀	DEVOPS DAY1	12 hr #1	N/A	9.3 sec

DOCKER HUB REPOSITORY OVERVIEW FOR poojaelango/docker-app:

poojaelango

Docker Personal

Repositories

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

Repositories / docker-app / General

poojaelango/docker-app

Last pushed about 3 hours ago

Add a description

Add a category

General

Tags

Image Management

Collaborators

Webhooks

Settings

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	about 3 hours

See all

buildcloud

Build with Docker Build Cloud

Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across your team, native multi-platform support, and accelerated data transfer, all without

Using 0 of 1 private repositories. Get more

Docker commands

To push a new tag to this repository:

docker push poojaelango/docker-app:tagname

Public view