

STEP 1:

UPDATE PACKAGES

```
selya2@MAN0J:/home/selya2/docker-app/devops-day2 sudo apt update
```

STEP 2:

INSTALL DOCKER

```
selya2@MAN0J:/home/selya2/docker-app/devops-day2 sudo apt install -y docker.io
```

STEP 3:

START AND ENABLE THE JENKINS

```
selya2@MAN0J:/home/selya2/docker-app/devops-day2 sudo systemctl start jenkins  
selya2@MAN0J:/home/selya2/docker-app/devops-day2 sudo systemctl enable jenkins  
selya2@MAN0J:/home/selya2/docker-app/devops-day2 sudo systemctl status jenkins
```

STEP 4:

USE THE CURL COMMAND

```
selya2@MAN0J:/home/selya2/docker-app/devops-day2 sudo curl -L "https://github.com/docker/compose
```

STEP 5:

USE chmod COMMAND

```
selya2@MAN0J:/home/selya2/docker-app/devops-day2 sudo chmod +x /usr/local/bin/docker-compose  
selya2@MAN0J:/home/selya2/docker-app/devops-day2 docker-compose -v
```

STEP 6:

CREATE A DIRECTORY AND CHANGE INTO THE DIRECTORY

```
selya2@MAN0J:/home/selya2/docker-app/devops-day2 mkdir ~/docker-app  
selya2@MAN0J:/home/selya2/docker-app/devops-day2 cd docker-app/
```

STEP 7:

CREATE NANO FILE CALLED "app.py"

CREATE NANO FILE CALLED "requirements.txt"

```
selya2@MAN0J:/home/selya2/docker-app/devops-day2 nano app.py  
selya2@MAN0J:/home/selya2/docker-app/devops-day2 nano requirements.txt
```

STEP 8:

USE cat COMMAND FOR "app.py"

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, world!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

STEP 9

```
selya2@MANOJ:~/docker-app/devops-day2$ cat requirements.txt
flask
```

STEP 10

```
selya2@MANOJ:~/docker-app/devops-day2$ cat Dockerfile
FROM python:3.11

# Set the working directory
WORKDIR /app

# Copy the requirements file and install dependencies
COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application files
COPY . /app/

# Expose the application port
EXPOSE 5000

# Define the command to run the app
CMD ["python", "app.py"]
```

STEP 11:

```
selya2@MANOJ:~/docker-app/devops-day2$ cat docker-compose.yml
version: '3.8'

services:
  app: # 📌 Name of the service (you can change it)
    build: .
    ports:
      - "5000:5000"
    volumes:
      - ./app
    restart: always
```

STEP 12:

NOW BUILD DOCKER USING docker-compose build COMMAND

```
selya2@MANOJ:/home/selya2/docker-app/devops-day2$ sudo docker-compose build
selya2@MANOJ:/home/selya2/docker-app/devops-day2$ sudo docker-compose up -d
selya2@MANOJ:/home/selya2/docker-app/devops-day2$ sudo docker ps
```

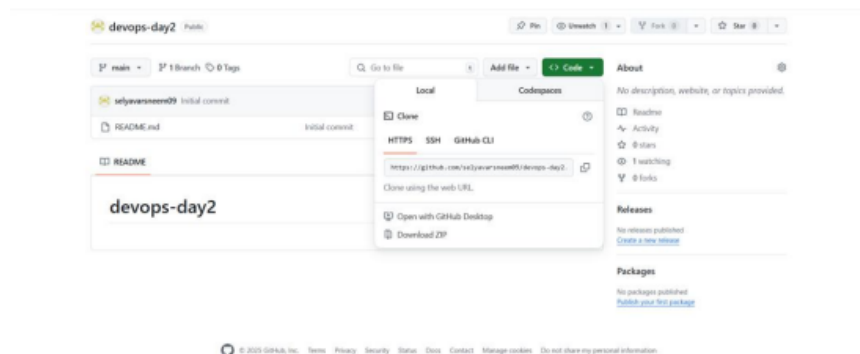
STEP 13:

NOW RUN THE LOCALHOST PORT:5000

Hello, world!

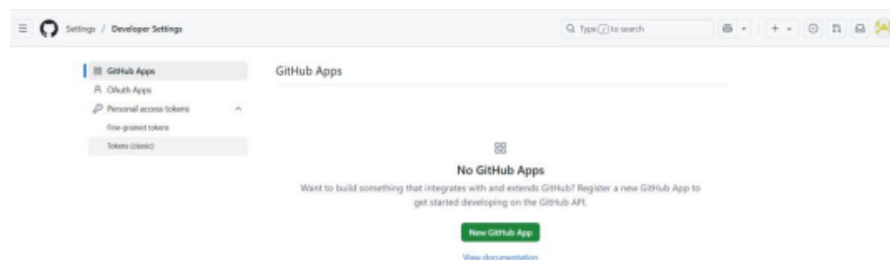
STEP 14:

CREATE A REPOSITORY AND COPY THE URL

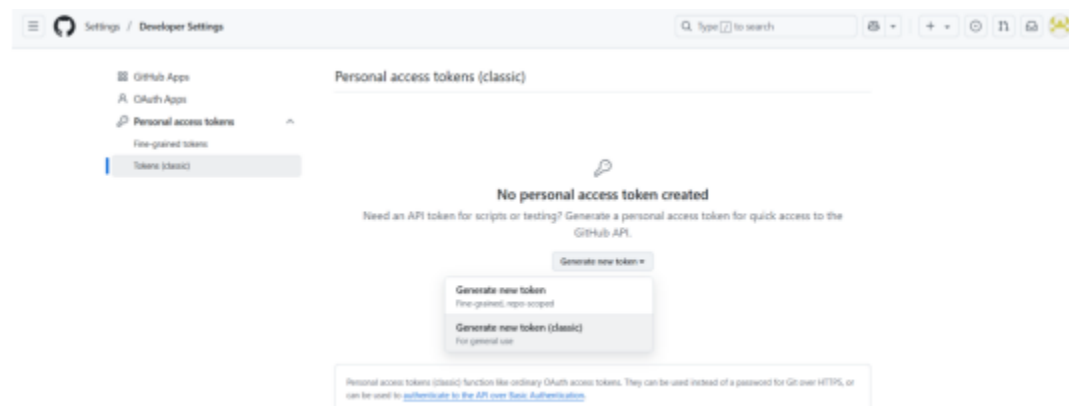


STEP 15:

GENERATE NEW TOKEN



STEP 16:



The screenshot shows the GitHub 'Personal access tokens (classic)' interface. On the left, a sidebar contains links for 'GitHub Apps', 'OAuth Apps', 'Personal access tokens' (which is highlighted), 'Fine-grained tokens', and 'Tokens (classic)'. The main content area is titled 'Personal access tokens (classic)' and includes a 'Generate new token' button. Below the title, a message states: 'Tokens you have generated that can be used to access the [GitHub API](#)'. A blue box with a warning icon says: 'Make sure to copy your personal access token now. You won't be able to see it again!'. Below this, a green box displays a token: 'ghp_1234567890123456789012345678901234567890' with a copy icon and a 'Delete' button. A final note explains that these tokens function like passwords for the GitHub API over HTTPS and can be used for authentication instead of Basic Authentication.

Configure

- General
- Triggers
- Pipeline**
- Advanced

Pipeline

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

Definition

Pipeline script

Pipeline script

Pipeline script from SCM

[try sample Pipeline...](#)

☒ Use Groovy Sandbox

[Pipeline Syntax](#)

STEP 20:

Configure

Pipeline

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

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URI

Please enter Git repository.

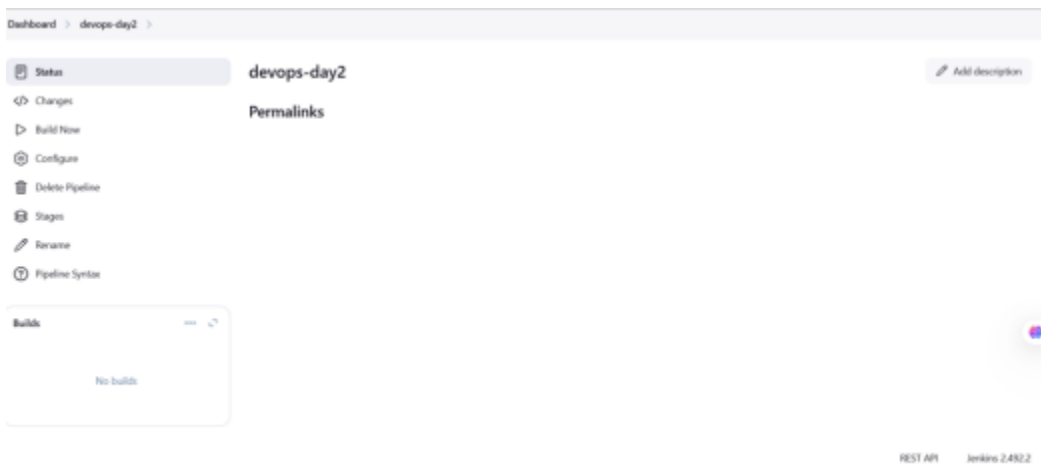
Credentials

none

+ Add

Save Apply

STEP 21:



STEP 22:

CLONE THE REMOTE REPO

```
selya11@LAPTOP-0R6B5QFD:~/devops-proj$ git clone https://github.com/selyavarsneem09/devops-day2.git
```

STEP 23:

ENTER INTO THE REPO AND LIST THE FILES

```
selya2@MANOJ:~/docker-app$ cd devops-day2/
selya2@MANOJ:~/docker-app/devops-day2$ ls
Dockerfile README.md app.py devops-day2 docker-compose.yml requirements.txt
```

STEP 24:

ADD THE FILES AND VERIFY THE STATUS

```
selya2@MANOJ:~/docker-app/devops-day2$ git add .
selya2@MANOJ:~/docker-app/devops-day2$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Dockerfile
    new file:   app.py
    new file:   devops-day2
    new file:   docker-compose.yml
    new file:   requirements.txt
```

STEP 25:

COMMIT THE CHANGES

```
selya2@MANOJ:~/docker-app/devops-day2$ git commit -m "first commit"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: empty ident name (for <selya2@MANOJ.>) not allowed
```

STEP 26:

CONFIGURE USING THE USER EMAIL AND USERNAME

```
selya2@MANOJ:~/docker-app/devops-day2$ git config --global user.email "selyadharsneemuthusamy@gmail.com"
selya2@MANOJ:~/docker-app/devops-day2$ git config --global user.name "selya911"
```

STEP 27:

PUSH THE CHANGES INTO THE REPO

```
selya2@MANOJ:~/docker-app/devops-day2$ git remote set-url origin https://selya911:ghp_QxeXksayUMIG7hw2yAqumX0084Nmp34AE0Qn@github.com/selya911/devops
selya2@MANOJ:~/docker-app/devops-day2$ git push origin main
```

STEP28:

CREATE A NANO FILE FOR JENKINS AND ADD,COMMIT IN TO THE REPOSITORY

```
selya2@MANOJ:~/docker-app/devops-day2$ nano jenkinsfile
selya2@MANOJ:~/docker-app/devops-day2$ git add jenkinsfile
selya2@MANOJ:~/docker-app/devops-day2$ git add .
selya2@MANOJ:~/docker-app/devops-day2$ git commit -m "added jenkins"
[main 447795c] added jenkins
1 file changed, 67 insertions(+)
create mode 100644 jenkinsfile
selya2@MANOJ:~/docker-app/devops-day2$ git push https://selya911:ghp_QxeXksayUMIG7hw2yAqumX0084Nmp34AE0Qn@github.com/selya911/devops-day2.git
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 947 bytes | 947.00 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/selya911/devops-day2.git
66a793f..447795c main -> main
```

STEP 29:

GO TO JENKINS AND SELECT “BUILD NOW”

```
Started by user selya varsnee muthusamy
Obtained Jenkinsfile from git https://github.com/selyavarsneem09/devops-day2.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/devops-day2
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/devops-day2/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/selyavarsneem09/devops-day2.git # timeout=10
Fetching upstream changes from https://github.com/selyavarsneem09/devops-day2.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/selyavarsneem09/devops-day2.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse origin/main^{commit} # timeout=10
Checking out Revision b526f20d7bcb73498bfb80608ce77ee107b67959 (origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f b526f20d7bcb73498bfb80608ce77ee107b67959 # timeout=10
Commit message: "commit"
> git rev-list --no-walk b526f20d7bcb73498bfb80608ce77ee107b67959 # timeout=10
[Pipeline] }
```


STEP 30:

YOU SHOULD BE NOW ABLE TO SEE THE MESSAGE

hello, world! Running inside docker!