### **DOCKER**

```
#Installing Docker on Ubuntu - go to official website (Docker docs) sudo snap install docker

#mkdir fapp2
#touch app.py

from flask import Flask app=Flask(__name__)
@app.route('/')

def run():
    return "vicky punda"
app.run('0.0.0.0',port=5000)

touch Dockerfile
```

### **Dockerfile**

FROM python:3.10
WORKDIR /app
COPY . /app
#RUN pip install -r requirements.txt
RUN pip install flask
EXPOSE 5000
CMD ["python3","app.py"]

## touch requirements.txt

## requirements.txt

Flask==2.0.1 Werkzeug==2.0.1

docker build -t fapp2.

#Docker.sock permission denied #cd .. do till you are in / #do ls -a #cd var #cd run #ls - there is docker.sock #sudo chmod 666 docker.sock

docker run -p 5000:5000 fapp2

SSH

ssh username@hostname\_or\_ip\_address

If key pair not generated already

# ssh-keygen ssh-copy-id username@hostname\_or\_ip\_address ssh username@hostname\_or\_ip\_address

```
Sabsusb=PC:-3 ssn-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sabs/.ssh/id_rsa):
/home/sabs/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sabs/.ssh/id_rsa
Your public key has been saved in /home/sabs/.ssh/id_rsa.pub
The key fingerprint is:
 SHA256:GOkOhwM05D3tc558UlPT/Fw0lb0oysqfSR7lzFaxQ/I sabs@SB-PC
 The key's randomart image is:
+---[RSA 3072]----+
    .+ .*|
0 0 . . 0 00|
0 0 + .00+ 0|
. = 0 0+00+.
           + * S +..E o
            = = ==.. .
              0 *0.=
                000=
        --[SHA256]-
 sabs@SB-PC:~$
sabs@SB-PC:~$
 sabs@SB-PC:~$
sabs@SB-PC:~$ ssh-copy-id nandha-krishna@10.17.126.26
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/sabs/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
nandha-krishna@10.17.126.26's password:
 Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'nandha-krishna@10 and check to make sure that only the key(s) you wanted were added
                                                                                 "ssh 'nandha-krishna@10.17.126.26'"
 sabs@SB-PC:~$ ssh nandha-krishna@10.17.126.26
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-48-generic x86_64)
  * Documentation: https://help.ubuntu.com

* Management: https://landscape.canonical.com

* Support: https://ubuntu.com/pro
Expanded Security Maintenance for Applications is enabled.
 0 updates can be applied immediately.
 Last login: Thu Nov 7 21:12:05 2024 from 10.17.96.124
 nandha-krishna@Nandha-Krishna-HP-Pavilion-Gaming-Laptop-15-dk1xxx:~$ ls
Desktop Documents Downloads Music nandharoopa Pictures Public pwd.txt snap Templates Videos
nandha-krishna@Nandha-Krishna-HP-Pavilion-Gaming-Laptop-15-dk1xxx:~$ 6
```

Q) Implement shell script to transfer application data from local server to remote server using SSH (use scp command inside shell script)

Here is a simple shell script to transfer application data from a local server to a remote server using `scp` (secure copy) over SSH. This script will copy a specified directory or file from your local machine to a remote server, ensuring data is transferred securely.

```
```bash
#!/bin/bash
```

#### # Define variables

LOCAL\_PATH="/path/to/local/application\_data" # Replace with the path to your local data REMOTE\_USER="remote\_username" # Replace with your remote server username

REMOTE\_HOST="remote\_server\_ip\_or\_hostname" # Replace with the remote server IP or hostname

REMOTE\_PATH="/path/to/remote/destination" # Replace with the path on the remote server where data will be copied

# Transfer data using scp echo "Starting data transfer from \$LOCAL\_PATH to \$REMOTE\_USER@\$REMOTE\_HOST:\$REMOTE\_PATH"

scp -r "\$LOCAL\_PATH" "\$REMOTE\_USER@\$REMOTE\_HOST:\$REMOTE\_PATH"

fi

#### ### Explanation of Script:

- 1. \*\*Variables\*\*: Set the paths and credentials for the local and remote servers.
- 2. \*\*`scp` Command\*\*: `scp -r` copies files and directories recursively.
- 3. \*\*Error Checking\*\*: The `if` statement checks if the `scp` command succeeded. `0` indicates success, while other codes indicate errors.

#### ### Usage:

- 1. Make sure you have SSH access to the remote server (you may need to exchange SSH keys).
- 2. Save the script, e.g., `transfer\_data.sh`.
- 3. Make the script executable:

```
```bash
sudo chmod +x transfer_data.sh
```

...

4. Run the script:

```
```bash
./transfer_data.sh
```

This script provides a secure way to transfer application data with 'scp'.

### **JENKINS**

sudo apt-get install jenkins sudo apt-get install fontconfig openjdk-17-jre (if Java doesnt exist)

sudo systemctl start jenkins sudo systemctl status jenkins

Go to <a href="https://localhost:8080">https://localhost:8080</a> - default Jenkins server

It will ask for the authentication key. Copy the path and use command

sudo cat "paste\_the\_path\_here"

Paste the key from terminal to the Jenkins server

Create an account with username and password

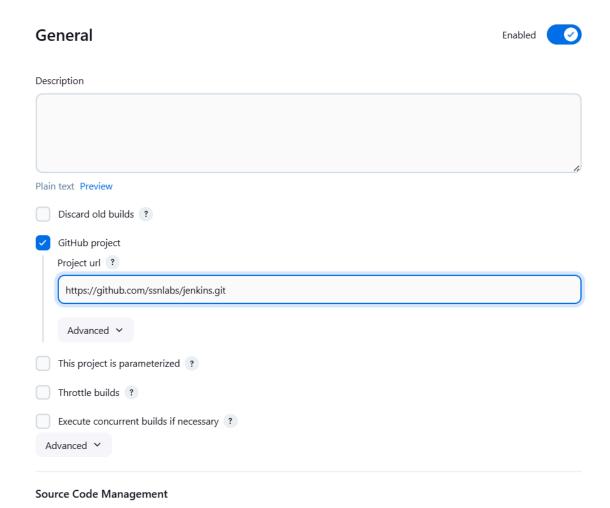
\_\_\_\_\_

## Create a repository with a Java file

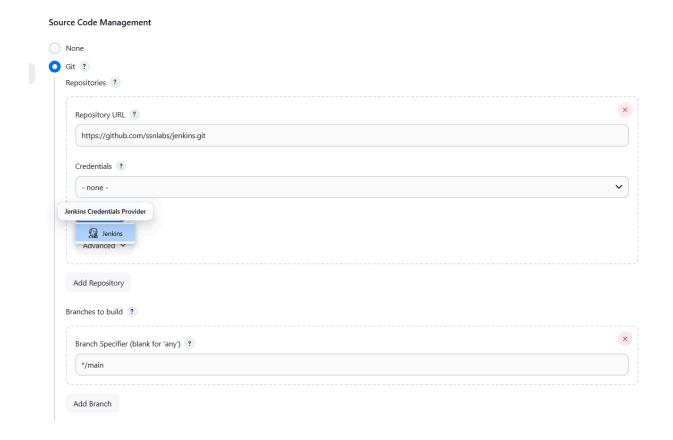
For eg: https://github.com/ssnlabs/jenkins.git

### **Jenkins**

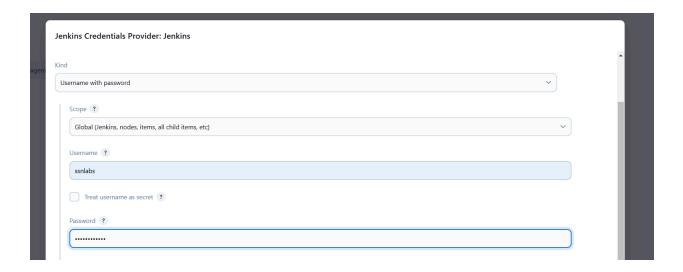
- 1. Click New Item
- 2. Give a name and select Freestyle Project
- 3. Select Github project and paste repository name



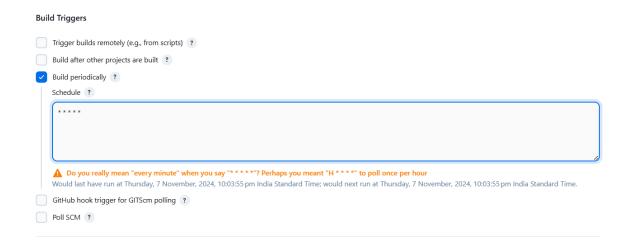
4. Select Git, Under Credentials, Click Add & Jenkins Change "Branch Specifier" to \*/main



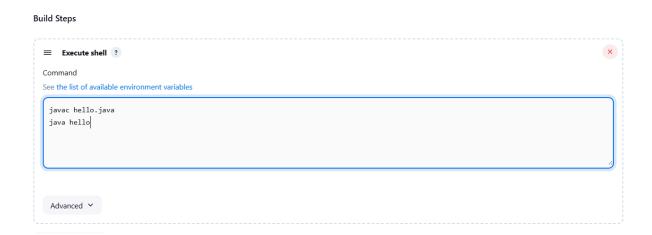
5. Enter your github username and password



6. Add \* \* \* \* \* under Build Periodically -> Schedule (Cron Job - triggers a build every minute)



7. Under Build Step -> Select Execute Shell and enter the following



## 8. Give Save and Build Now in next page

# 9. Enjoy!

# Pipeline script

```
pipeline {
       agent any
       stages {
       stage('Checkout') {
       steps {
              // Replace 'your-repository-url' with your repository URL
              git url: 'https://github.com/your-repository-url.git'
      }
      }
       stage('Build') {
       steps {
              // Commands to build the application
              // Example for a Node.js application
              sh 'npm install'
              sh 'npm run build'
      }
       }
       stage('Test') {
       steps {
              // Commands to run tests
              // Example for running Jest tests
              sh 'npm test'
       post {
              always {
```

```
// Archive test results (JUnit format for example)
              junit 'reports/junit/*.xml'
              }
      }
       stage('Results') {
       steps {
              // Display test results in Jenkins
              echo 'Displaying test results'
       }
       }
       post {
       always {
       // Clean up workspace if necessary
       cleanWs()
       }
       }
}
```