**Intellipatt**

**Project**: Capstone I

DevOps Certification Training

You have been hired as a Sr. DevOps Engineer in Abode Software. They want to implement DevOps Lifecycle in their company. You have been asked to implement this lifecycle as fast as possible. Abode Software is a product-based company and their product is available on this GitHub link.

<https://github.com/hshar/website.git>

Following are the specifications of the lifecycle:

1. Install the necessary software on the machines using a configuration management tool

2. Git workflow has to be implemented

3. Code Build should automatically be triggered once a commit is made to master branch or develop branch. a. If a commit is made to master branch, test and push to prod b. If a commit is made to develop branch, just test the product, do not push to prod

4. The code should be containerized with the help of a Dockerfile. The Dockerfile should be built every time there is a push to GitHub. Use the following pre-built container for your application: hshar/webapp the code should reside in '/var/www/html'

5. The above tasks should be defined in a Jenkins Pipeline with the following jobs:

a. Job1: build

b. Job2: test

c. Job3: prod

**\*\*Project: Intellipatt - DevOps Implementation\*\***

**\*\*Summary: \*\***

Intellipatt's Capstone I project aims to implement a DevOps lifecycle at Abode Software, a product-based company. The project involves rapid implementation of the DevOps lifecycle. The company's product repository is hosted on GitHub, and the main tasks include installing necessary software, implementing Git workflows, automating code builds, and containerizing the application using Docker. The project is defined within a Jenkins Pipeline consisting of three jobs: build, test, and production deployment. The project is executed using Oracle VirtualBox with Ubuntu VMs on a laptop for cost-effective testing.

**\*\*Specifications: \*\***

1. \*\*Installation and Configuration: \*\* Set up Ubuntu VMs with specific requirements and configurations for SRIMASTER, SRIPROD, and SRITEST.

2. \*\*Cluster and SSH: \*\* Establish a cluster configuration using SSH keys for secure communication between SRIMASTER and SRIPROD/SRITEST.

3. \*\*Git Fork: \*\* Fork the company's GitHub repository to create a personal copy for the project.

4. \*\*Jenkins Setup: \*\* Install Jenkins on SRIMASTER and configure nodes for SRITEST and SRIPROD.

5. \*\*SMEE Client: \*\* Use SMEE client for webhook functionality due to the project's private network setup.

6. \*\*Jenkins Jobs: \*\* Configure three Jenkins jobs: Job1 for build, Job2 for testing, and Job3 for production deployment.

7. \*\*Dockerization: \*\* Implement Docker containerization for the application using Dockerfile and create Docker images.

8. \*\*Git Webhooks: \*\* Set up GitHub webhooks to trigger Jenkins jobs on code commits and changes.

**\*\*Implementation Flow: \*\***

1. Install and configure VMs for SRIMASTER, SRIPROD, and SRITEST with specific resources.

2. Establish SSH key-based communication between SRIMASTER and SRIPROD/SRITEST.

3. Fork the company's GitHub repository to create a personal repository for the project.

4. Install and configure Jenkins on SRIMASTER.

5. Use the SMEE client for webhook forwarding due to private network constraints.

6. Create Jenkins jobs (Job1, Job2, Job3) to facilitate build, testing, and deployment.

7. Implement Docker containerization using Dockerfile for the application.

8. Configure GitHub webhooks to trigger Jenkins jobs on code changes.

9. Execute the project through the Jenkins Pipeline on the designated nodes.

**\*\*Results: \*\***

The DevOps lifecycle is successfully implemented at Abode Software, enabling rapid development, continuous integration, and automated deployment. Jenkins Pipeline orchestrates the entire process, from code commits to containerized deployment. With SRIMASTER managing configurations and SRITEST/SRIPROD executing tasks, the company achieves improved efficiency, reduced manual intervention, and streamlined software delivery.

Please find the project solution below with Jenkin pipeline’s job description.

* Job1: Build > SRITESTJOB1
* Job2: test > SRITESTJOB2
* Job3: prod > SRIPRODJOB3

**The project solution flow:**



**GIT Repository’s to use:**

* Clone web: <https://github.com/hshar/website.git> (for Clone)
* For Project: <https://github.com/andhukuri/Capstone.git> (Personal)

**Virtual Machine Requirements**

* 40 GB VHD for Each VM, 3 GB Ram, Select 2 processor
* Oracle Virtual box: <https://www.virtualbox.org/wiki/Downloads>
* Ubuntu 20.4: <https://www.osboxes.org/ubuntu/> (Please note: This should be only for Testing/project use not for production setup, personal use, please check the OS type based on which we need to add Jenkins or other repositories for installation of application)
* SRIMASTER we need install Ansible and write a YAMl file to be executed in Nodes (SRITEST and SRIPROD) for installing the applications remotely.

**System Requirements:**

* Windows 10 with 16 GB Memory, 512 SSD Disk, I5 Processor

**Application Requirement and other notes:**

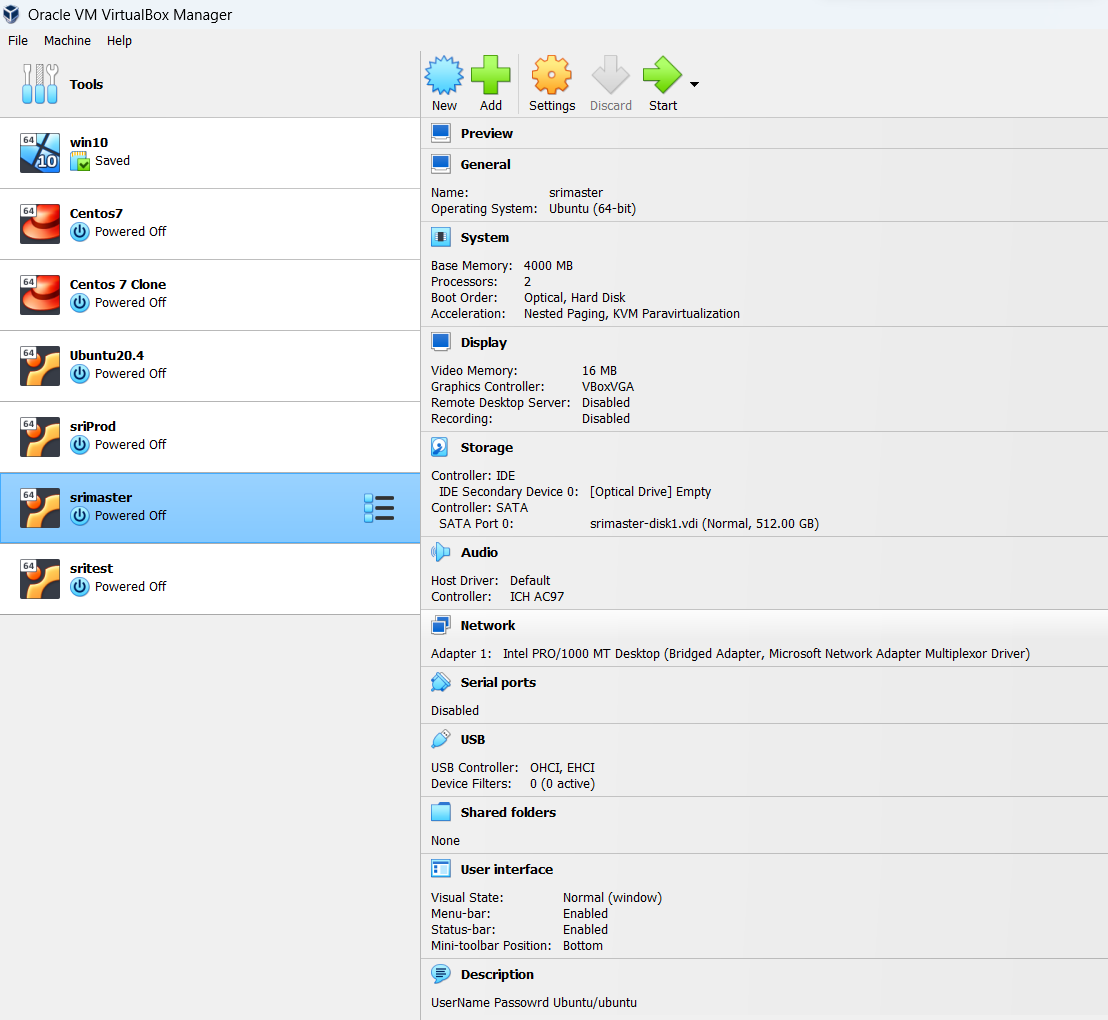
**Mobaxterm:** For SSH session and launching multiple VM session. This will also help in exporting terminal output to word file.

**SMEE.io:** It is not required if using AWS or Azure having public IP and required ports should be open for webhook. This is only used for testing purpose. <https://smee.io/> (We need to continuously give one session for smee client for proper function of webhook to Jenkins)

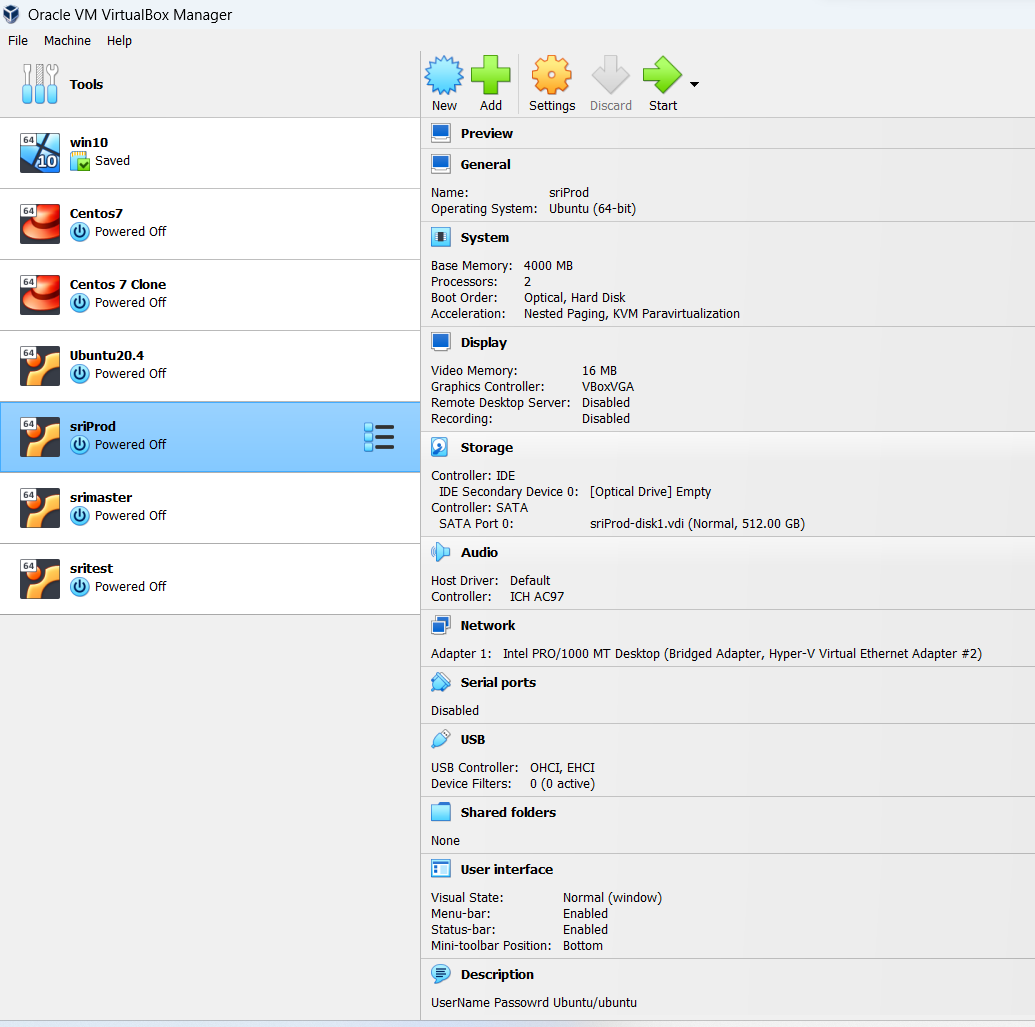
**VM Setup/Configuration**

Please note that keep one VM’s active and update with latest updates then clone VM’s generating unique MAC Address (this is to avoid network conflicts)

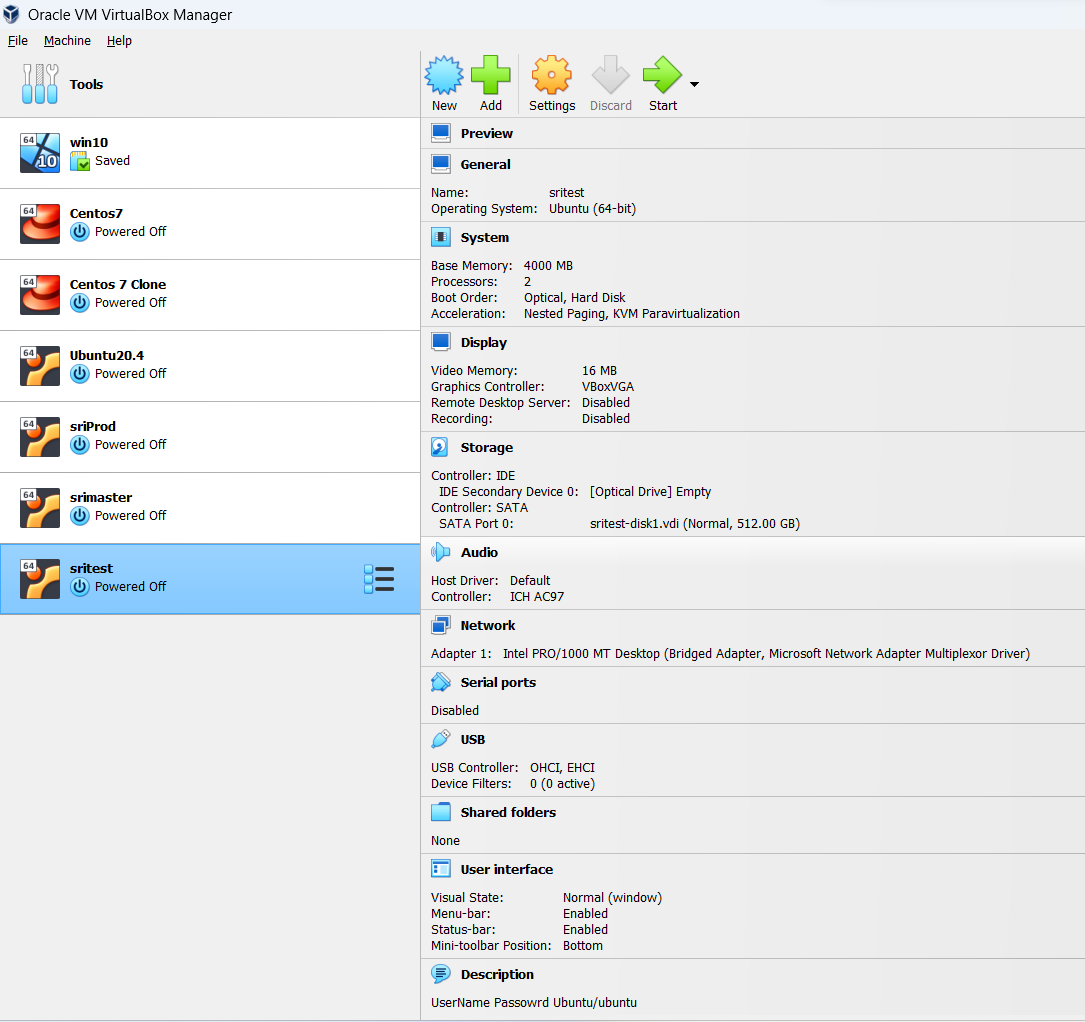
**VM Host:** SRIMASTER



**VM Host:** SRIPROD

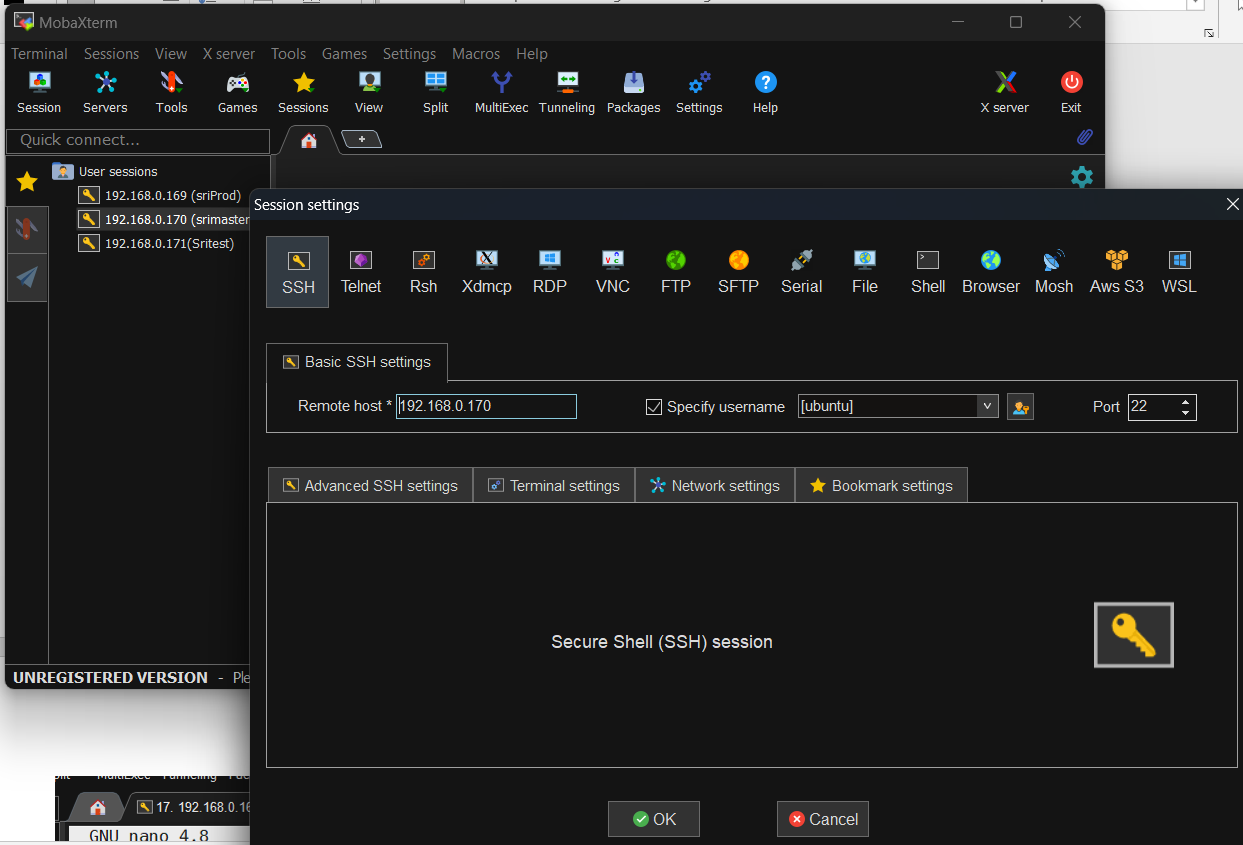


**VM Host:** SRITEST



**Host Name changes to match the project**

Connect using **Mobaxterm**:

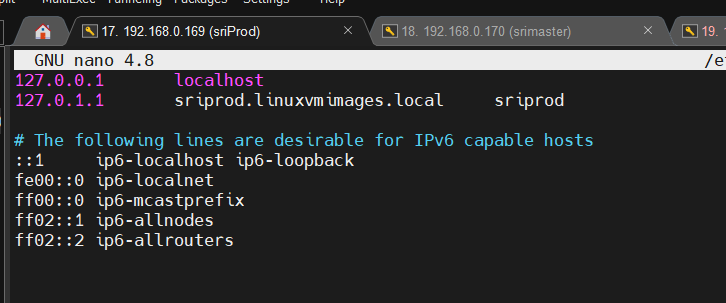


Perform below configuration in each server:

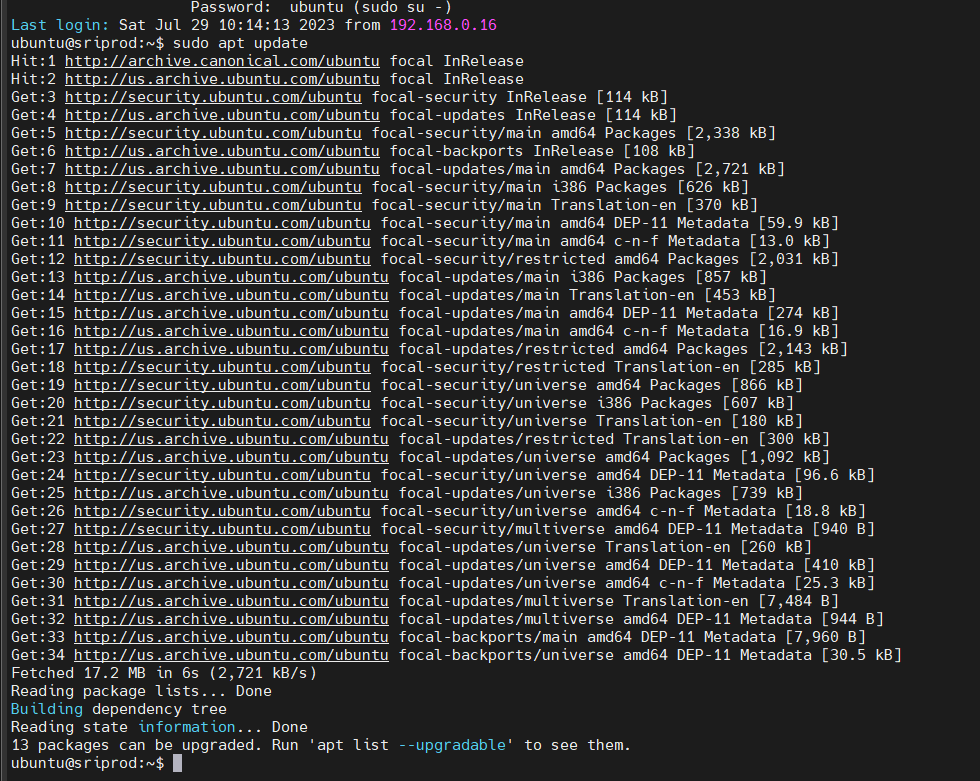
* sudo hostnamectl set-hostname sriprod



* sudo nano /etc/hosts (Modify to XXXX.linuxvmimages.local XXXX in line 2)

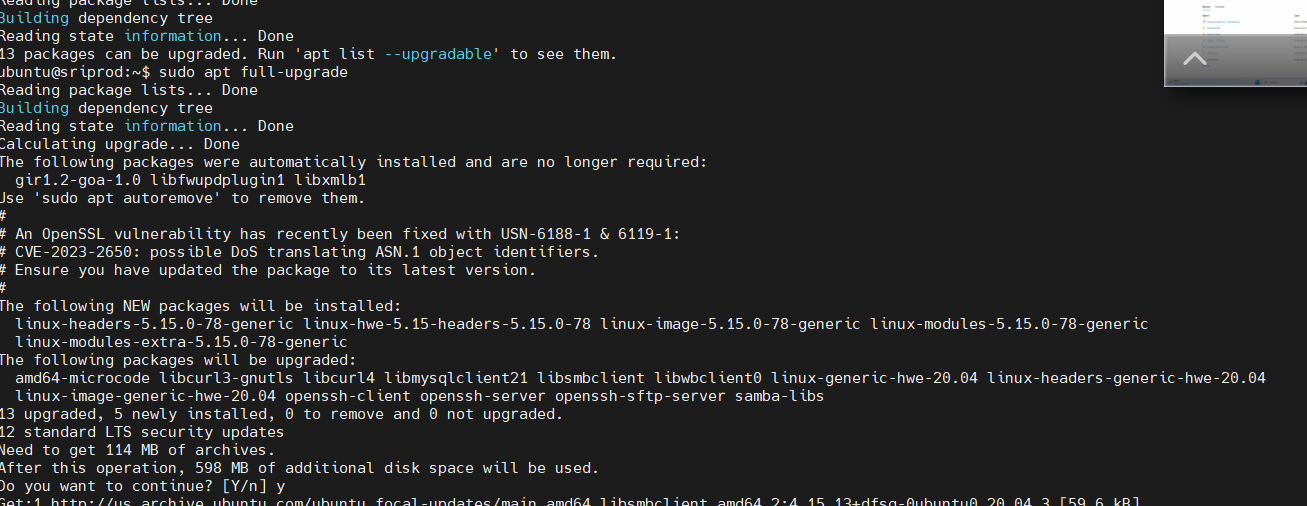


* + to save ctrl + x and Y - Yes to return to shell
* Sudo Reboot
* sudo apt upgrade : For updating the packages and say Y at do you want to continue

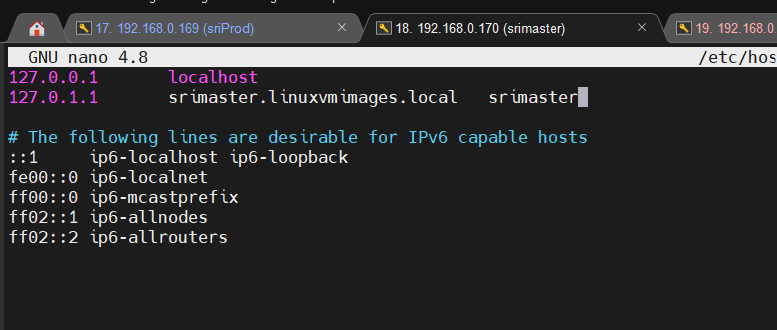


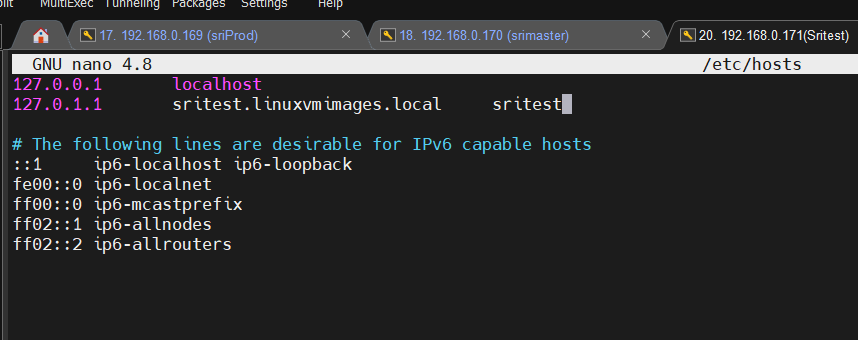
or

* sudo apt full-upgrade: for force update all packages and say Y at do you want to continue.



* Repeat for SRIMASTER and SRITEST





**Cluster using SSH key**

For configuring cluster seamlessly need SSH keygen in SRIMaster and copy public key from (/home/ubuntu/.ssh/id\_rsa.pub) in nodes SRITEST and SRIPROD (Note: We are using Ubuntu user with elevated credential and also has sudo access)

On SRIMASTER VM:

ubuntu@srimaster:~$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/home/ubuntu/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/ubuntu/.ssh/id\_rsa

Your public key has been saved in /home/ubuntu/.ssh/id\_rsa.pub

The key fingerprint is:

SHA256:+q1AKr5qAAzJMYx5zPsaJjqd4DwFbsK8oAPQJZ4HM/8 ubuntu@srimas ter

The key's randomart image is:

+---[RSA 3072]----+

|+B. |

|=oX . |

|o+ X |

|oo= o |

|\* .o .. S |

|B=o..oE. |

|@=++. o |

|=B+. o . |

|o++. o.. |

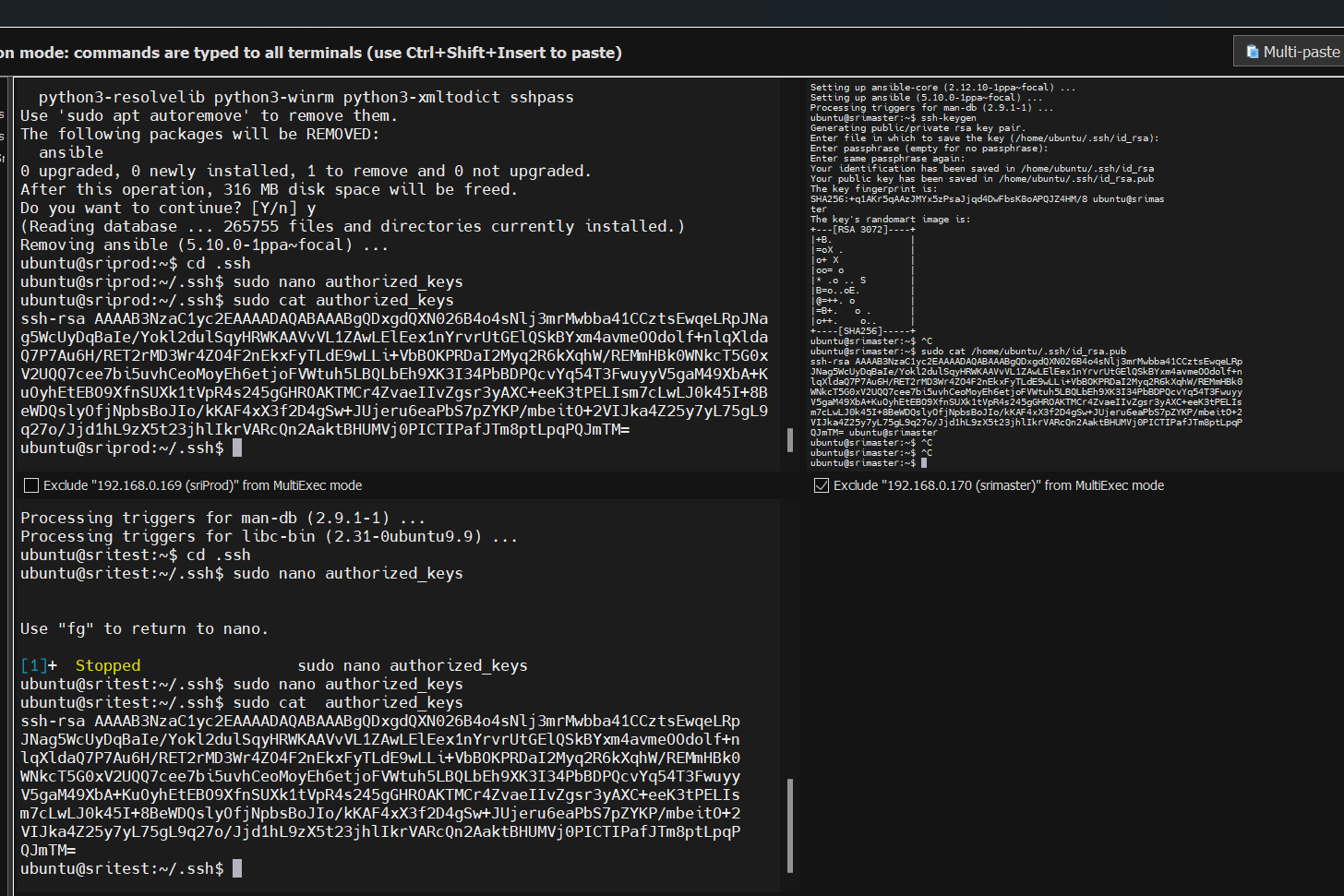
+----[SHA256]-----+

ubuntu@srimaster:~$ sudo cat /home/ubuntu/.ssh/id\_rsa.pub

ssh-rsa  ubuntu@srimaster

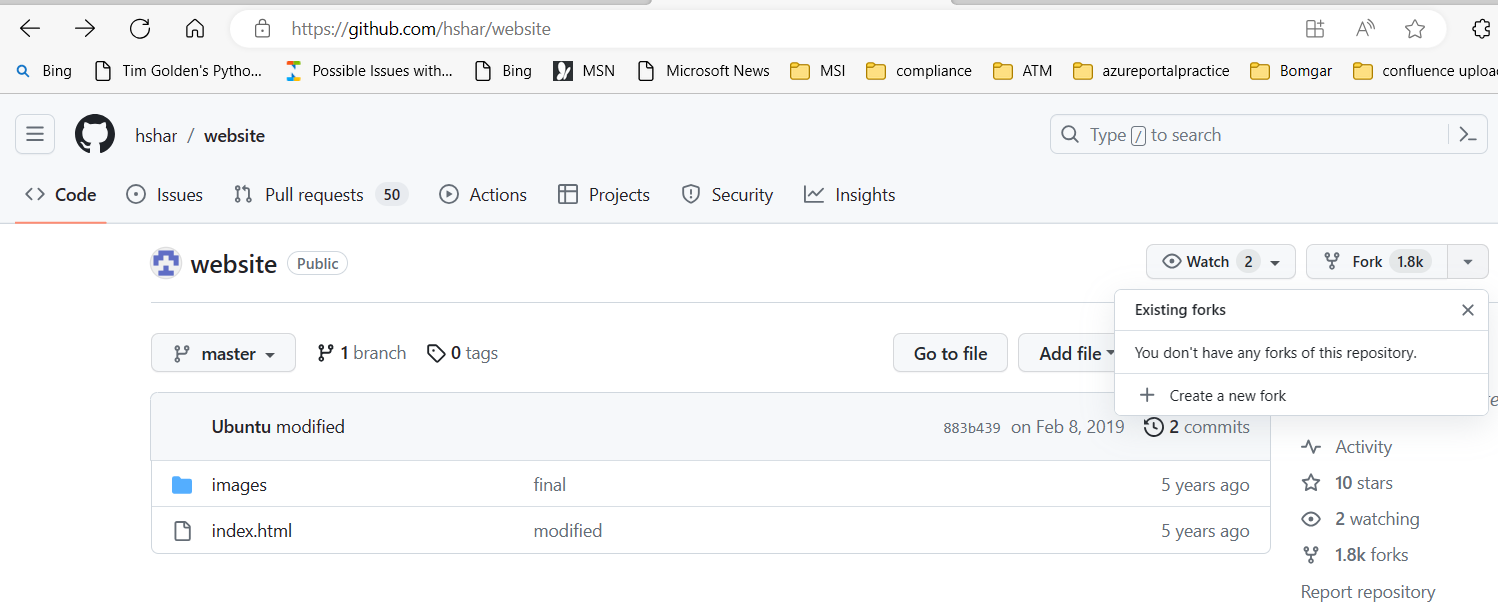
On SRITEST and SRIPROD VMs: Both in sritest and sriprod VM: copy key from srimaster “cat /home/ubuntu/.ssh/id\_rsa.pub” (Note: Authorized\_keys may not present or the file is already present with existing key press enter and input/copy your key)

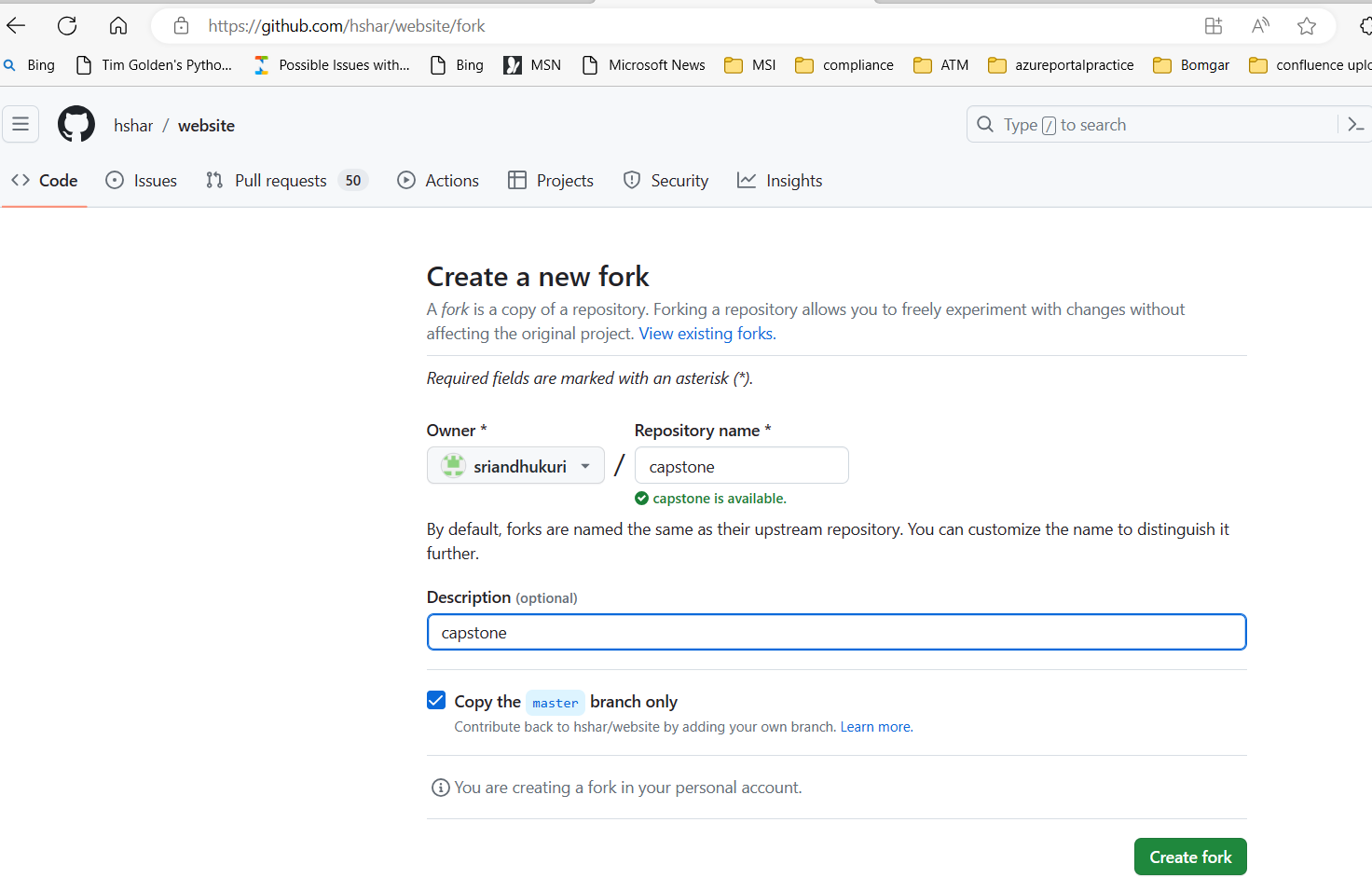
* Cd .ssh
* Sudo nano authorized\_keys (copy public key generated from srimaster from ‘ssh-rsa’ till ‘=’ do not copy ubuntu@srimaster which is end of the key to avoid issues)
* Ctrl+x and y to save file on both vms

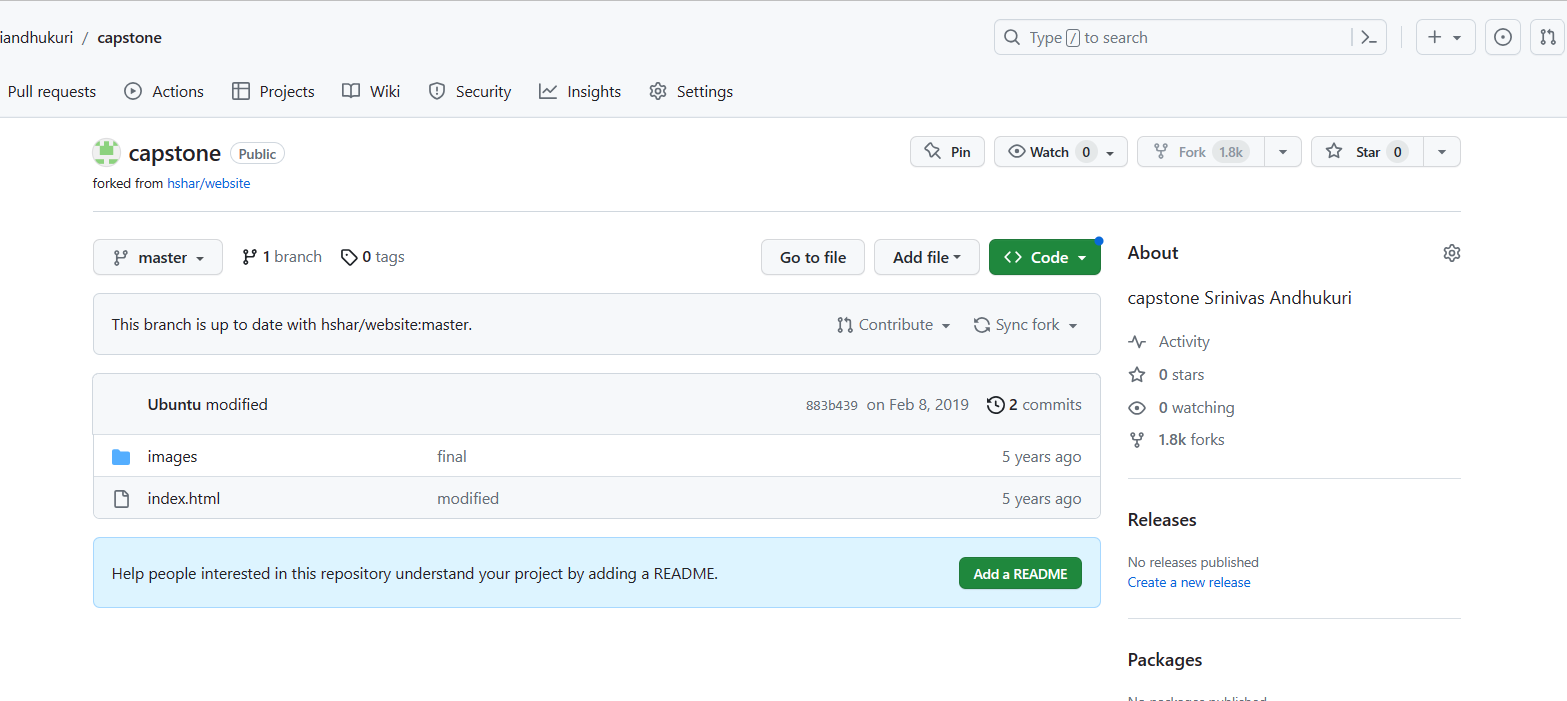


**GIT Fork to create your own repository and making the changes.**

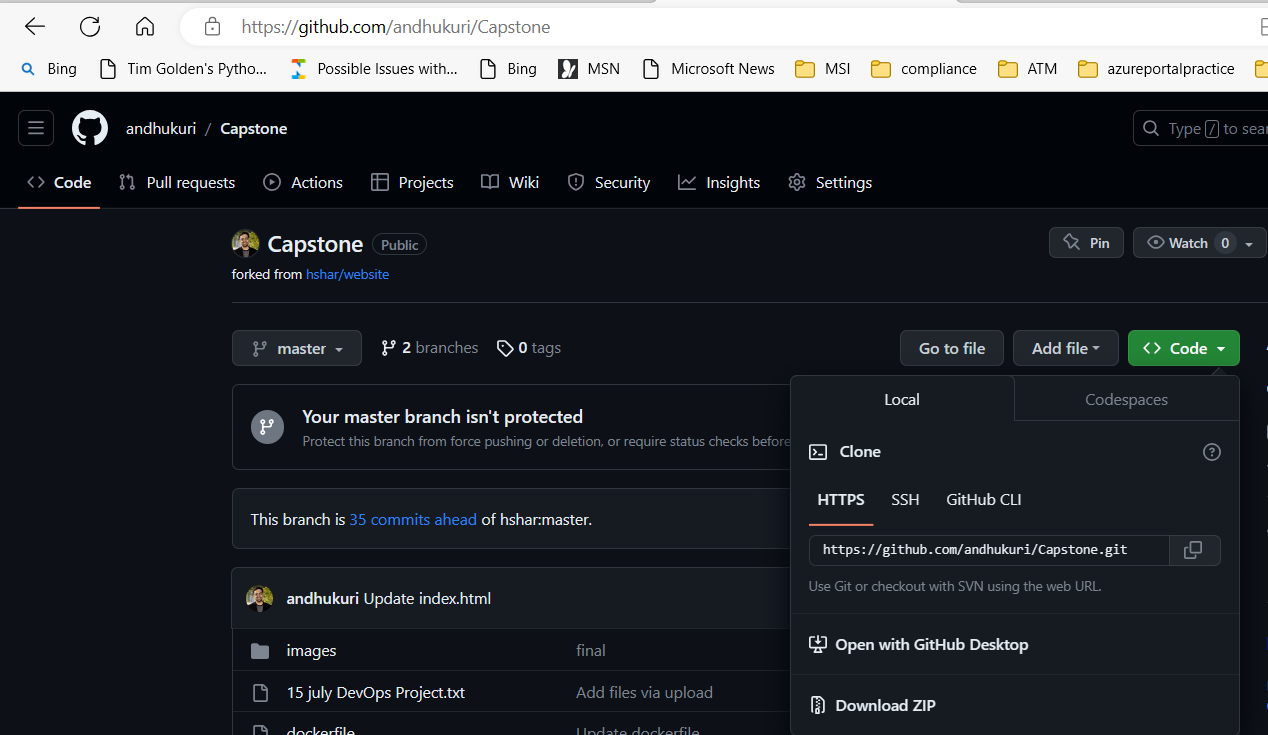
* Launch URl in the browser: <https://github.com/hshar/website.git>
* Click on fork option dropdown button and select create a new fork
* Provide the repository name capstone and add description
* Select click on Fork to create your own repository.
  + Note: You can fork and create a repository once per project, we need to cleanup existing repo present in our github to recreate again.







* Go to your own repository created, Click on code dropdown arrow.
* Copy the URL: <https://github.com/andhukuri/Capstone.git> for git clone in MasterServer



On SRIMASTER Configuration:

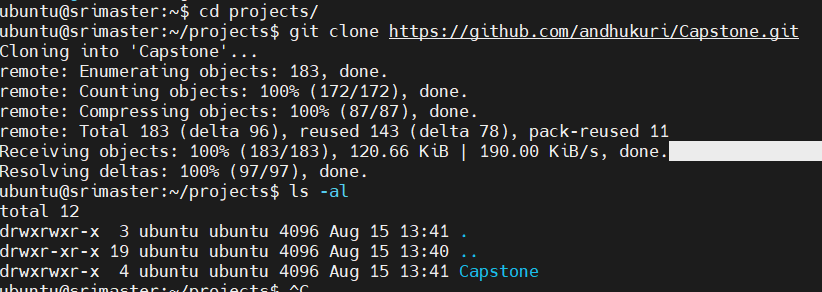
* $ sudo apt update
* $ sudo apt install software-properties-common
  + This will install the Common and dependent/recommended/security Application/ updates for Ubuntu. Below will be changed based on time to time
    - ca-certificates: Common CA certificates
    - gir1.2-glib-2.0: Introspection data for GLib, GObject, Gio and GModule
    - gir1.2-packagekitglib-1.0 (>= 1.1.0-2) : GObject introspection data for the PackageKit GLib library
    - packagekit: Provides a package management service
    - python-apt-common (>= 0.9) :Python interface to libapt-pkg (locales)
    - python3 : interactive high-level object-oriented language (default python3 version)
    - python3-dbus :simple interprocess messaging system (Python 3 interface)
    - python3-gi :Python 3 bindings for gobject-introspection libraries
    - python3-requests-unixsocket :Use requests to talk HTTP via a UNIX domain socket - Python 3.x
    - python3-software-properties (= 0.98.9.2) : manage the repositories that you install software from

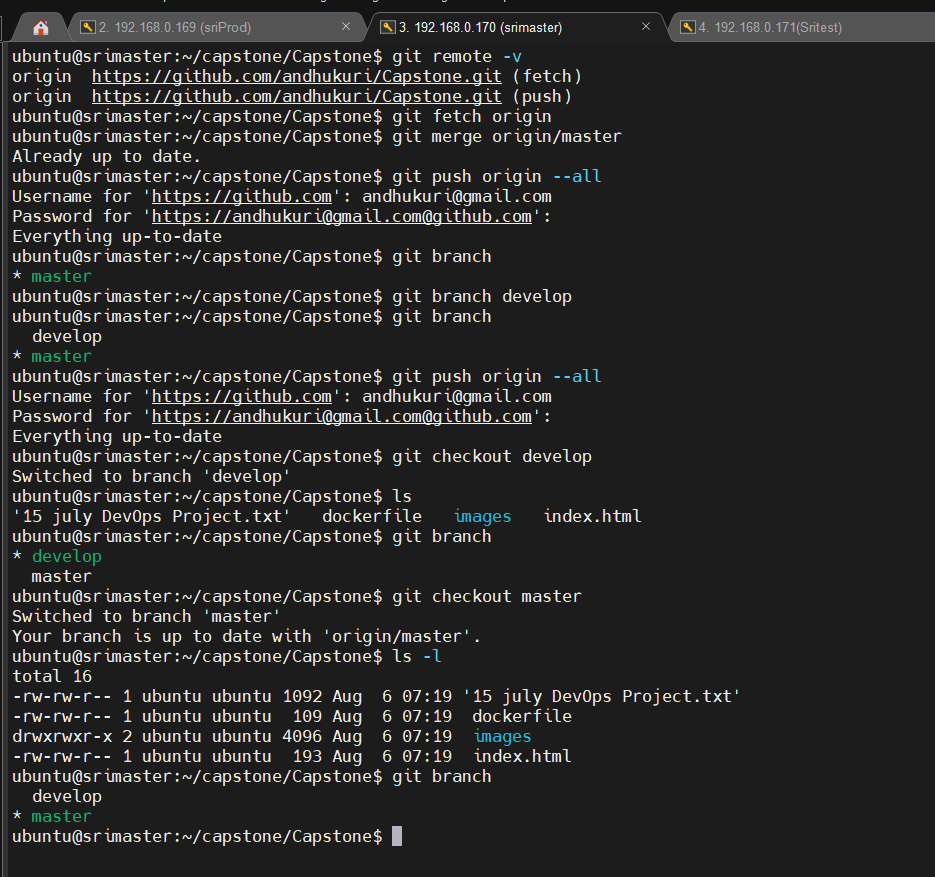
GIT Configuration: (Please note: while performing the clone you would only see two files index.html and images folder only)

* ubuntu@srimaster:~$ pwd

/home/ubuntu

* ubuntu@srimaster:~$ mkdir projects
* ubuntu@srimaster:~$ cd projects/
* ubuntu@srimaster:~/projects$ mkdir capstone
* ubuntu@srimaster:~/projects$ git clone
* ubuntu@srimaster:~$ cd projects/
* ubuntu@srimaster:~/projects$ git clone https://github.com/andhukuri/Capstone.git Cloning into 'Capstone'...
* remote: Enumerating objects: 183, done.





* ubuntu@srimaster:~/projects/Capstone$ pwd  
  /home/ubuntu/projects/Capstone
* ubuntu@srimaster:~/projects/Capstone$ git remote -v (show remote repository details)
* ubuntu@srimaster:~/projects/Capstone$git fetch origin

**Ansible Installation:**

* $ sudo add-apt-repository --yes --update ppa:ansible/ansible
* $ sudo apt install ansible
* $ sudo apt install ansible
  + For installing Ansible on SRIMASTER
  + Ref: <https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-ubuntu>
* ubuntu@srimaster:~$ cd /etc/ansible
* ubuntu@srimaster:/etc/ansible$ ls -l

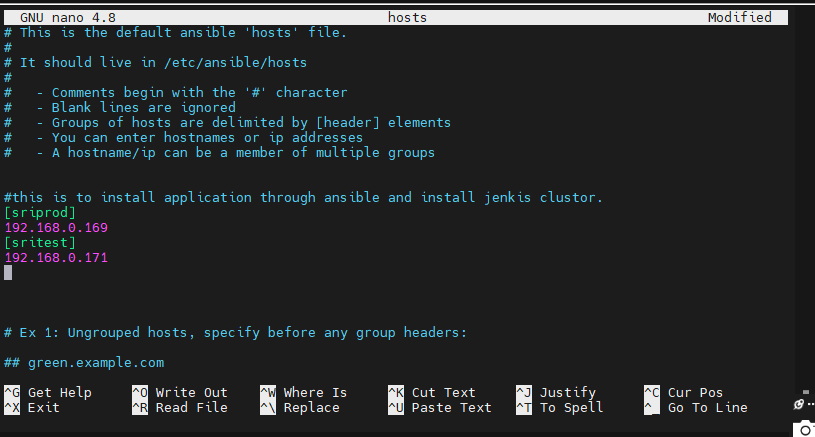
total 12

-rw-r--r-- 1 root root 614 Oct 11 2022 ansible.cfg

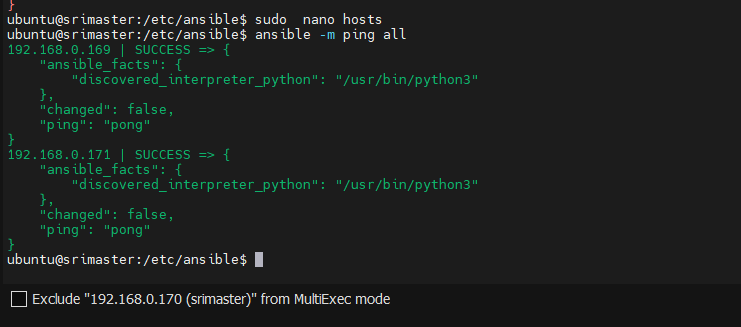
-rw-r--r-- 1 root root 1018 Oct 11 2022 hosts

drwxr-xr-x 2 root root 4096 Oct 25 2022 roles

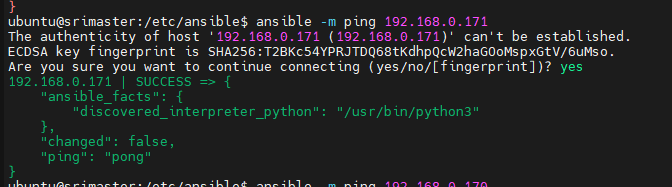
* ubuntu@srimaster:/etc/ansible$ sudo nano hosts
  + Please update/add the group for sriprod and sritest with IPaddress similar to below screen.
  + Ctrl+x and Y to save file



* ubuntu@srimaster:/etc/ansible$ ansible -m ping all



* ubuntu@srimaster:/etc/ansible$ ansible -m ping 192.168.0.171
  + Trying to ping individual IP to check the connectivity and ssh is working as expected.



For the project we require to create 3 files on srimaster. For installing required applications remotely through ansible jobs.

* capstoneproj-install.yaml (for executing the ansible job and calling the bash script for app installation the file will be executed in srimaster)
* docker.sh (The bash script will be executed in sritest & sriprod)
* jenkins.sh (The bash script will be executed in srimaster)

Application to be installed: Please create the file as below.

* SRIMASTER: Jenkins, docker, Java: Jenkins.sh (Note: Jenkins repo addition is not required if latest repo is already present)
* SRITEST: Java, Docker: docker.sh
* SRIPROD: Java, Docker: docker.sh

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(capstoneproj-install.yaml)

---

- hosts: localhost

become: true

name: install jenkins, java and Docker

tasks:

- name: srimaster task

script: jenkins.sh

- hosts: sritest

become: true

name: install java and Docker

tasks:

- name: sritest task

script: docker.sh

- hosts: sriprod

become: true

name: install java and Docker

tasks:

- name: sriprod task

script: docker.sh

----------------------------------------------------------------

(docker.sh)

sudo apt-get update

sudo apt-get install openjdk-11-jdk -y

sudo apt-get install docker.io -y

--------------------------------------------------------------

(jenkins.sh)

 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/](https://pkg.jenkins.io/debian-stable%20binary/) | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install curl -y

sudo apt-get install openjdk-11-jdk -y

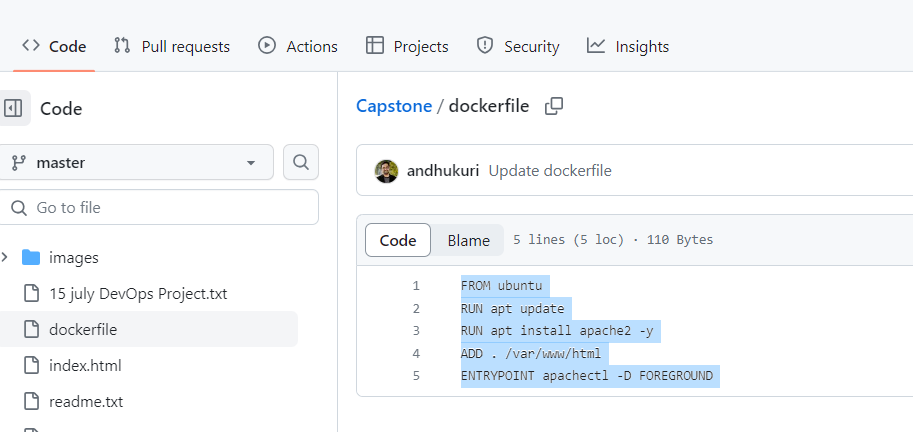
sudo apt-get install docker.io -y

sudo apt-get install jenkins -y

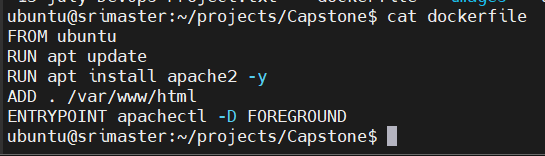
Docker File: This used for file for adding file to container. This file should be present in git repo.

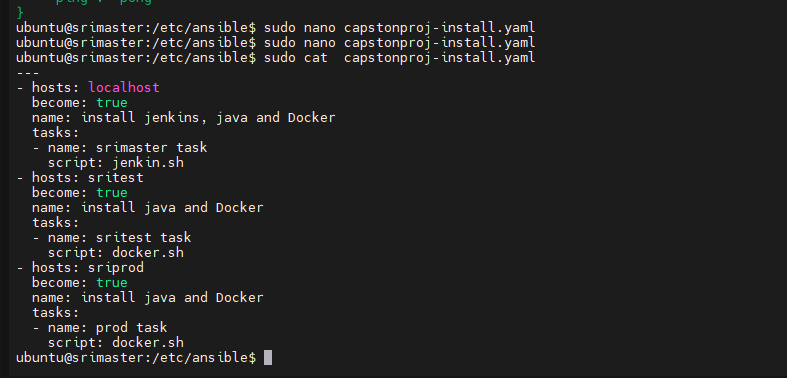
(dockerfile)

FROM ubuntu  
RUN apt update  
RUN apt install apache2 -y  
ADD . /var/www/html  
ENTRYPOINT apachectl -D FOREGROUND

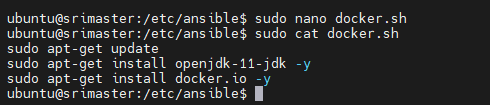


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

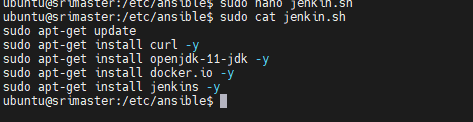
* ubuntu@srimaster:/etc/ansible$ cd /home/ubuntu/projects/Capstone (to create files in the git directory for future reference, These are not required config files in production git repo : Jenkins.sh, docker.sh, capstoneproj-install.yaml only for reference)
* ubuntu@srimaster:~/projects/Capstone$ sudo nano dockerfile
* ubuntu@srimaster:~/projects/Capstone$ cat dockerfile
* 
* ubuntu@srimaster:~/projects/Capstone$ sudo nano capstonproj-install.yaml
* ubuntu@srimaster:~/projects/Capstone$ sudo cat capstonproj-install.yaml



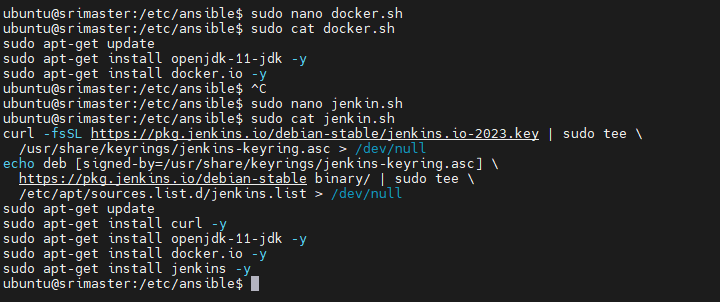
* ubuntu@srimaster:~/projects/Capstone$ sudo nano docker.sh
* ubuntu@srimaster:~/projects/Capstone$ sudo cat docker.sh



* ubuntu@srimaster:~/projects/Capstone$ sudo nano jenkin.sh
* ubuntu@srimaster:~/projects/Capstone$ sudo cat jenkin.sh

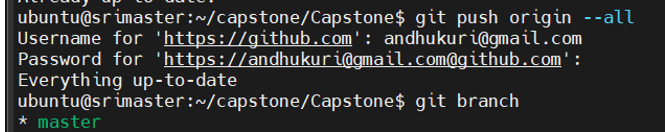


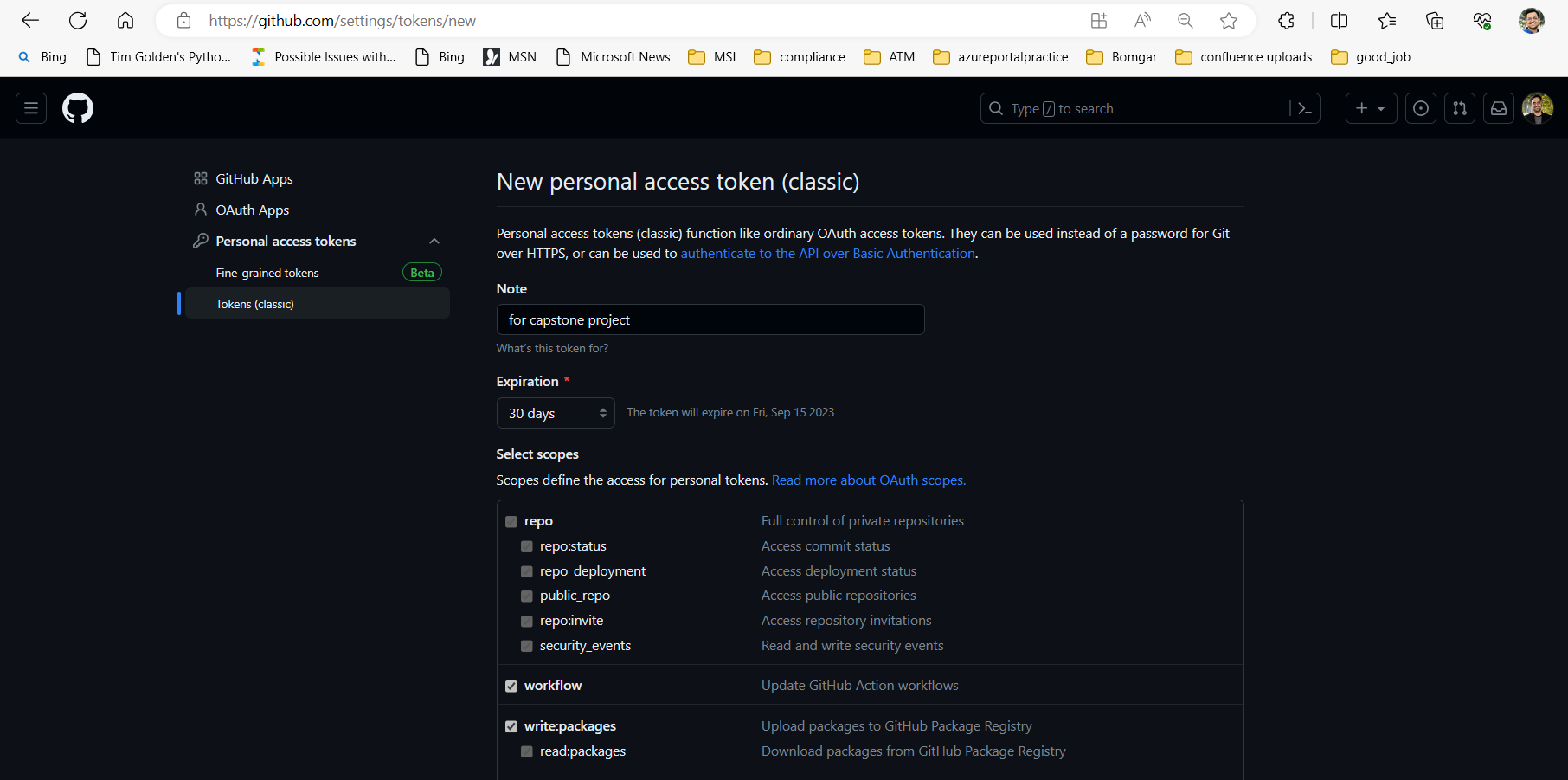
* ubuntu@srimaster:~/projects/Capstone$ sudo cat docker.sh



 For nano file to save ctrl +x , Y to save and click enter.

* ubuntu@srimaster:~/projects/Capstone$git add . && git commit -m “Uploading file yaml, sh, dockerfile files local git)
* ubuntu@srimaster:~/projects/Capstone$git branch develop (Creating development branch per req)
* ubuntu@srimaster:~/projects/Capstone$git branch (List the branches)
* ubuntu@srimaster:~/projects/Capstone$git push origin -all (to push updates to remote repository and upload the files)
  + for git push to work properly you need personal access token for password
  + go to git hub profile -> Settings->Developer Settings->Personal access tokens-> token classic
    - Click on the button generate now -> new personal access token (classic): give required permission and click on generate token. This can be used has password for uploading content to git repo.





Notes: For troubleshooting Jenkins installation. Curl is not required however installed to avoid issue.

Before installing the Jenkins, we need to find the OS Version (Long Term Support release or Weekly release) that is being used to properly configuring the repository and install the Jenkins Application. For proper function it requires Java11 while write this doc Ref: <https://www.jenkins.io/doc/book/installing/linux/>

* curl -fsSL <https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key> | sudo tee \

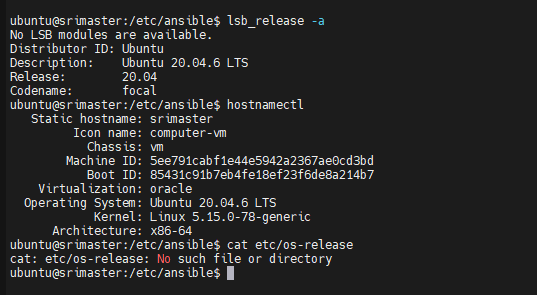
/usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

[https://pkg.jenkins.io/debian-stable binary/](https://pkg.jenkins.io/debian-stable%20binary/) | sudo tee \

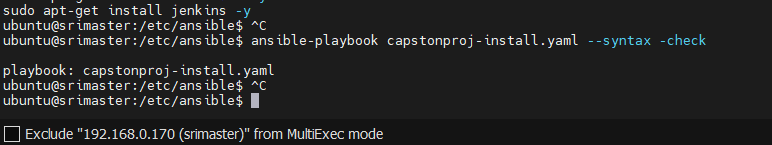
/etc/apt/sources.list.d/jenkins.list > /dev/null

* sudo apt-get update
* sudo apt-get install jenkins
* lsb\_release -a **or** cat etc/os-release **or** Hostnamectl (Note in the description and operating system with **LTS**)

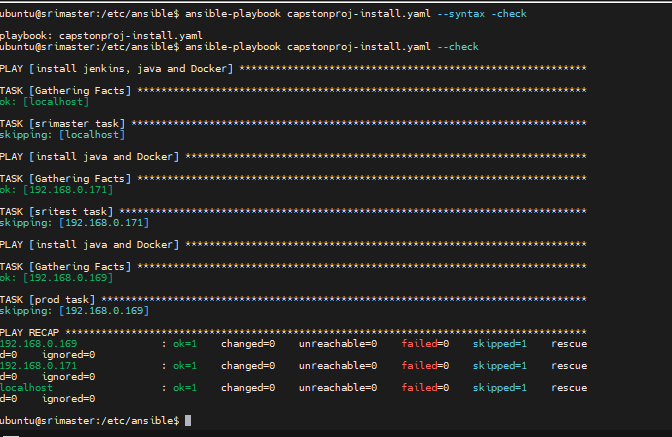


* ubuntu@srimaster:~/projects/Capstone$ ansible-playbook capstonproj-install.yaml --syntax -check
  + Validating Yaml file through Ansible

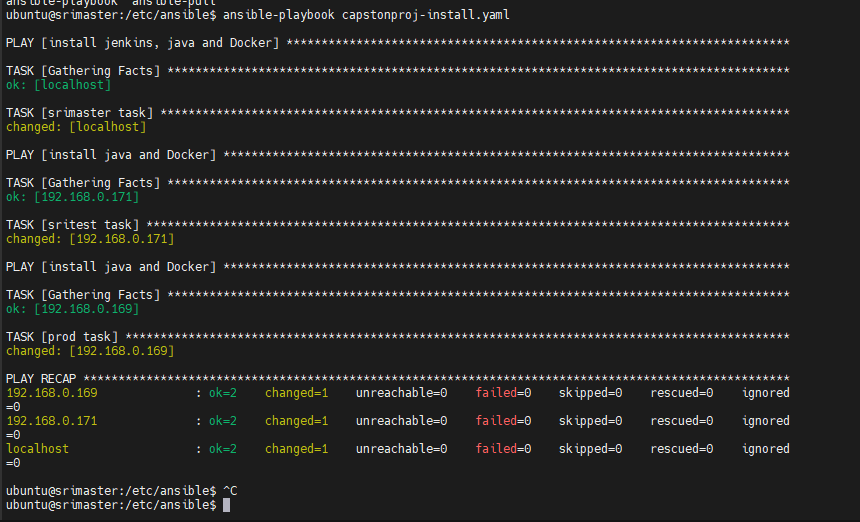
**playbook**: capstonproj-install.yaml



* ubuntu@srimaster:~/projects/Capstone$ ansible-playbook capstonproj-install.yaml –check
  + Validating connectivity of yaml playbook



* ubuntu@srimaster:~/projects/Capstone$ ansible-playbook capstonproj-install.yaml
  + Final execution which will install the required application in the cluster nodes remotely.



Post successful installation verification of application version using moboextreme multiple execute model.

ubuntu@srimaster:/etc/ansible$ jenkins --version

2.401.3

ubuntu@srimaster:/etc/ansible$ docker --version

Docker version 20.10.25, build 20.10.25-0ubuntu1~20.04.1

ubuntu@srimaster:/etc/ansible$ java --version

openjdk 11.0.20 2023-07-18

OpenJDK Runtime Environment (build 11.0.20+8-post-Ubuntu-1ubuntu120.04)

OpenJDK 64-Bit Server VM (build 11.0.20+8-post-Ubuntu-1ubuntu120.04, mixed mode, sharing)

 #################sritest

ubuntu@sritest:~/.ssh$ jenkins --version

jenkins: command not found

ubuntu@sritest:~/.ssh$ docker --version

Docker version 20.10.25, build 20.10.25-0ubuntu1~20.04.1

ubuntu@sritest:~/.ssh$ java --version

openjdk 11.0.20 2023-07-18

OpenJDK Runtime Environment (build 11.0.20+8-post-Ubuntu-1ubuntu120.04)

OpenJDK 64-Bit Server VM (build 11.0.20+8-post-Ubuntu-1ubuntu120.04, mixed mode, sharing)

ubuntu@sritest:~/.ssh$ ^C

ubuntu@sritest:~/.ssh$

##################sriprod

ubuntu@sriprod:~/.ssh$ jenkins --version

jenkins: command not found

ubuntu@sriprod:~/.ssh$ docker --version

Docker version 20.10.25, build 20.10.25-0ubuntu1~20.04.1

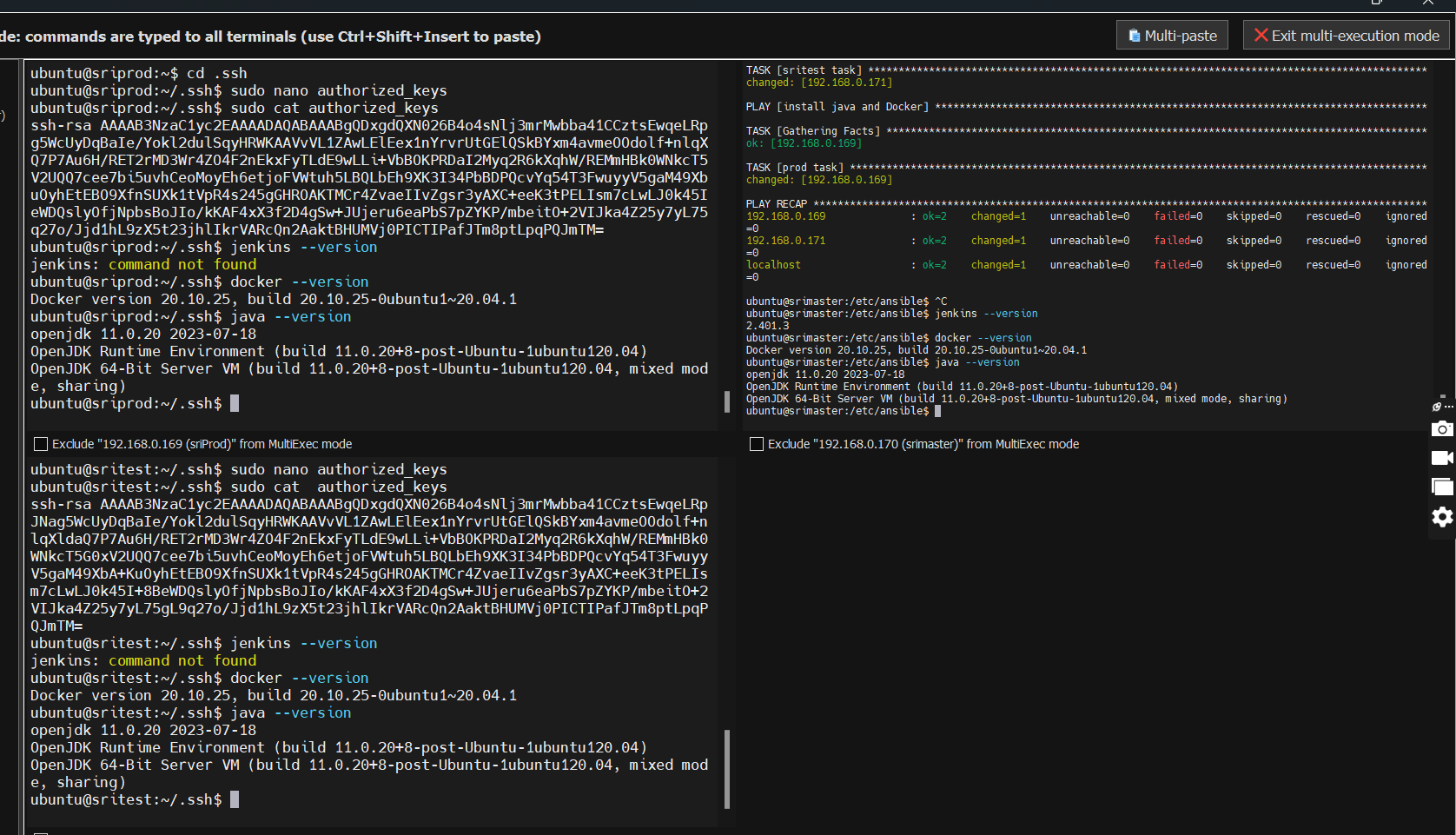
ubuntu@sriprod:~/.ssh$ java --version

openjdk 11.0.20 2023-07-18

OpenJDK Runtime Environment (build 11.0.20+8-post-Ubuntu-1ubuntu120.04)

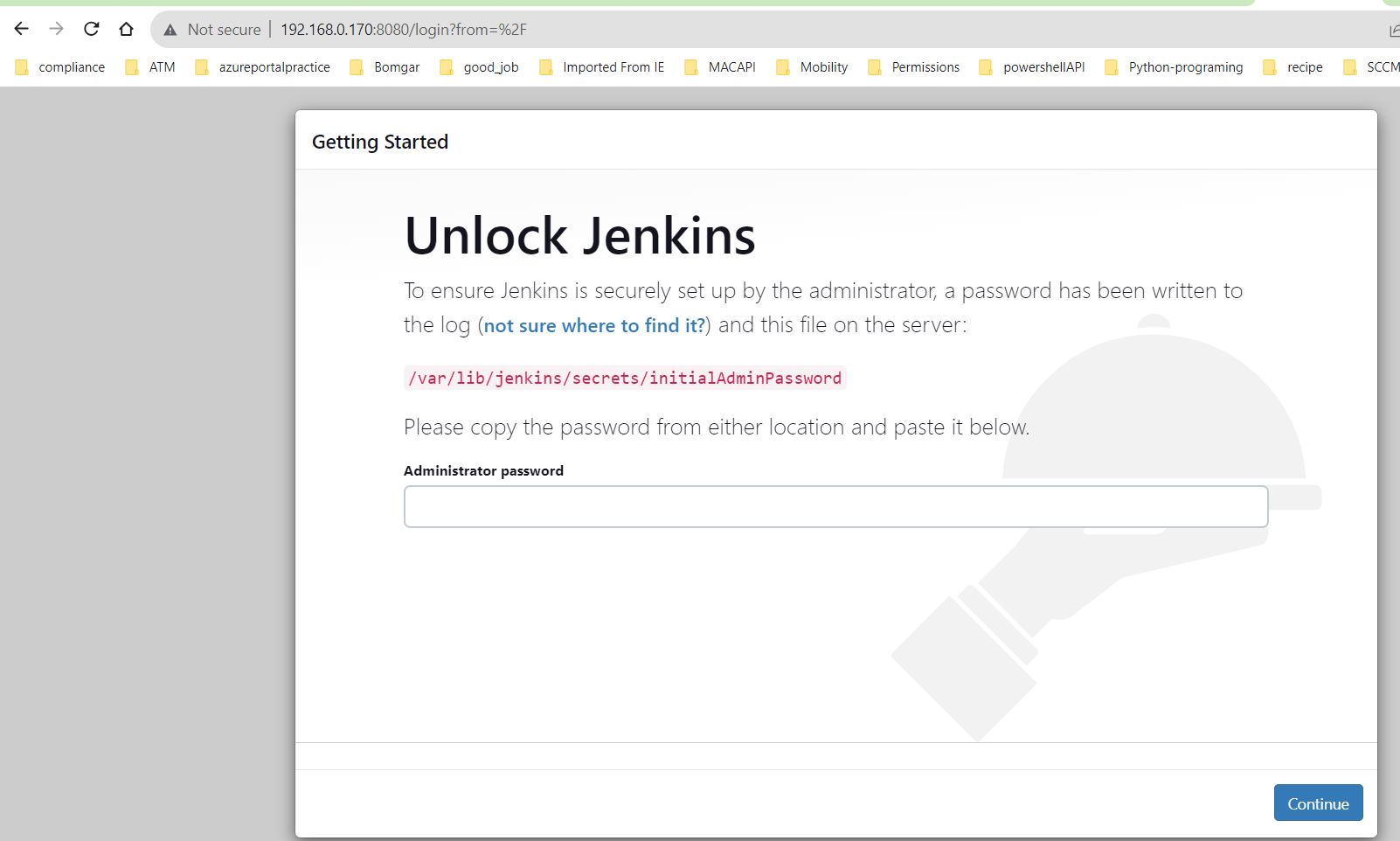
OpenJDK 64-Bit Server VM (build 11.0.20+8-post-Ubuntu-1ubuntu120.04, mixed mode, sharing)

ubuntu@sriprod:~/.ssh$



**Jenkins Configuration on SRIMASTER: IP 192.168.0.170:8080**

* Launch web URL : 192.168.0.170:8080

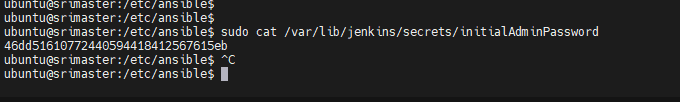




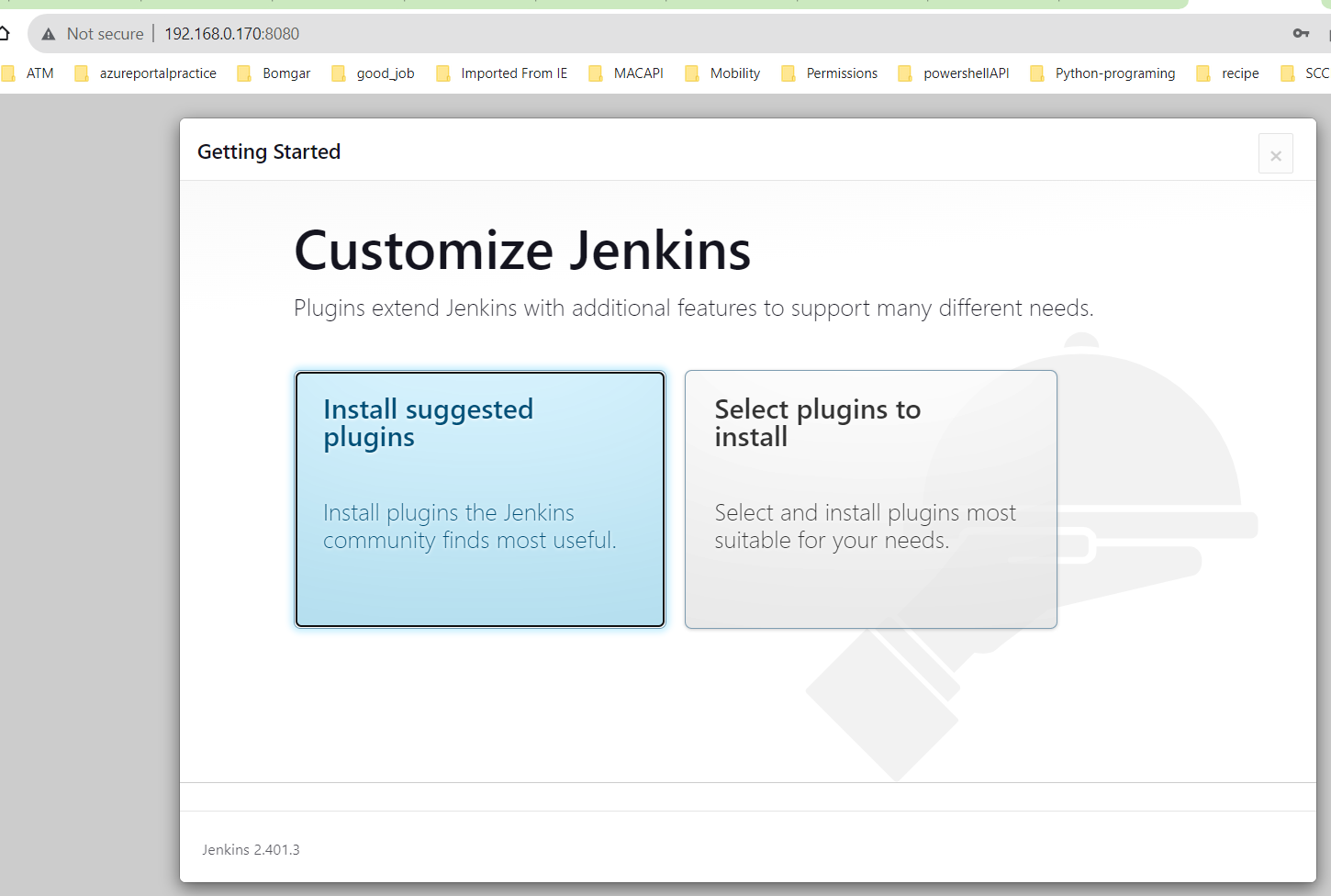
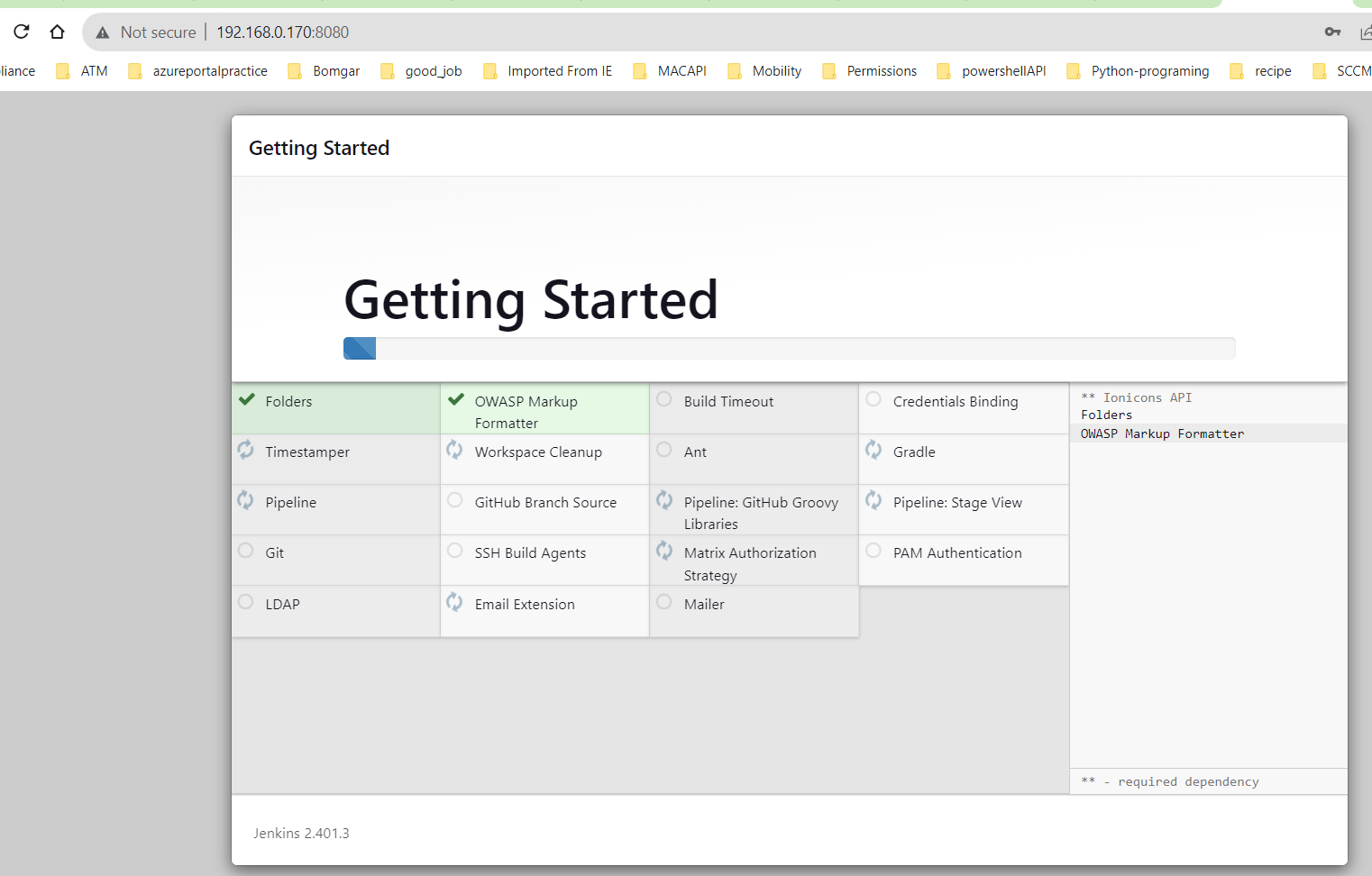
On Jenkins server run command to get admin password

* ubuntu@srimaster:/etc/ansible$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

46dd51610772440594418412567615eb

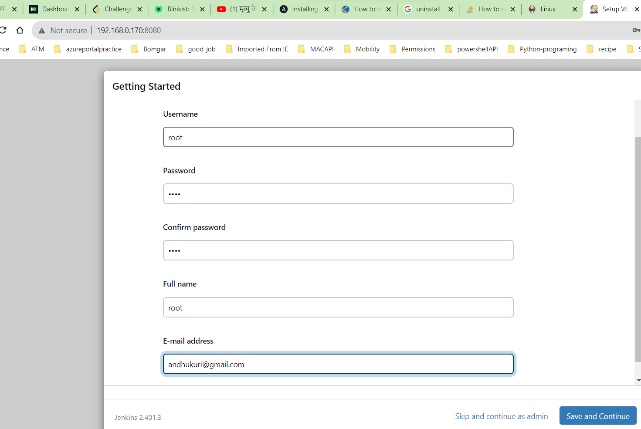
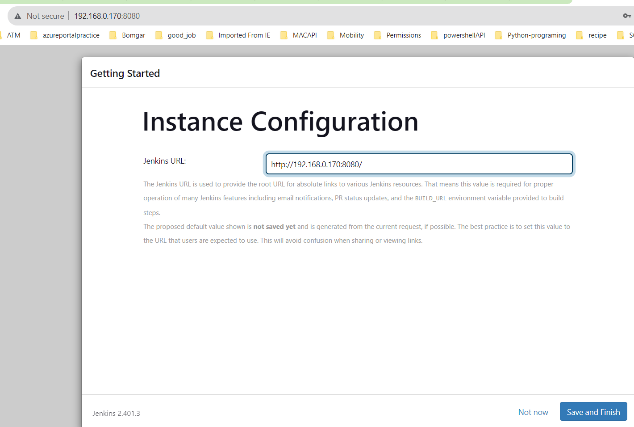


* Click “Installed suggested Plugins

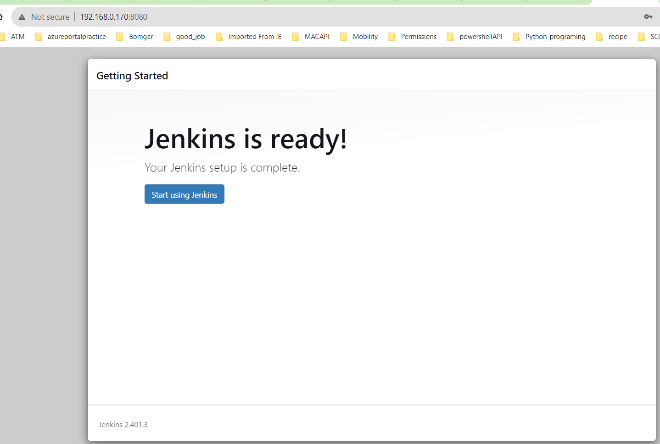


* In the following screen give user name and password root. (This is for test environment only and should not use in production) and in next screen save and finish.



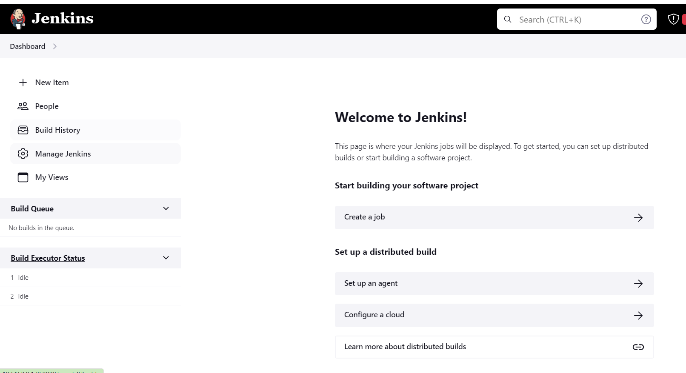
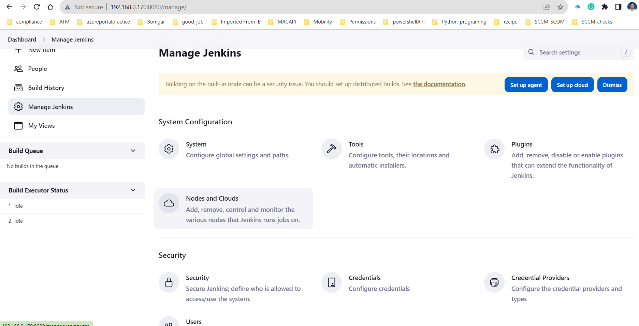
* Click on start using Jenkis





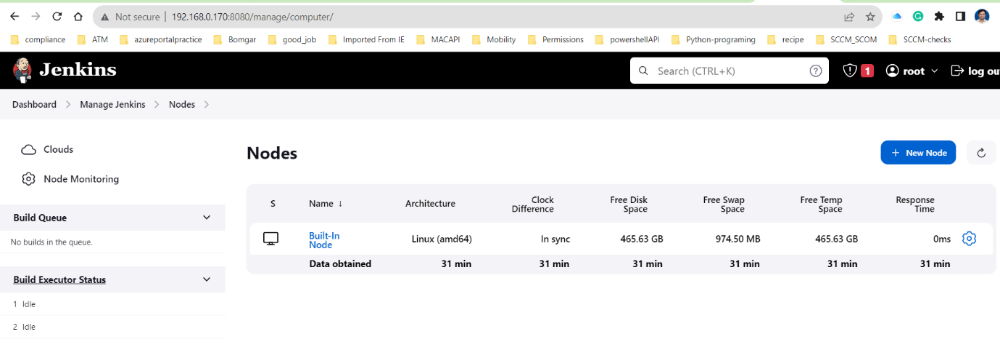
**Add nodes sritest and sriprod for Jenkins job execution.**

* Click on Manage Jenkins > nodes and cloud

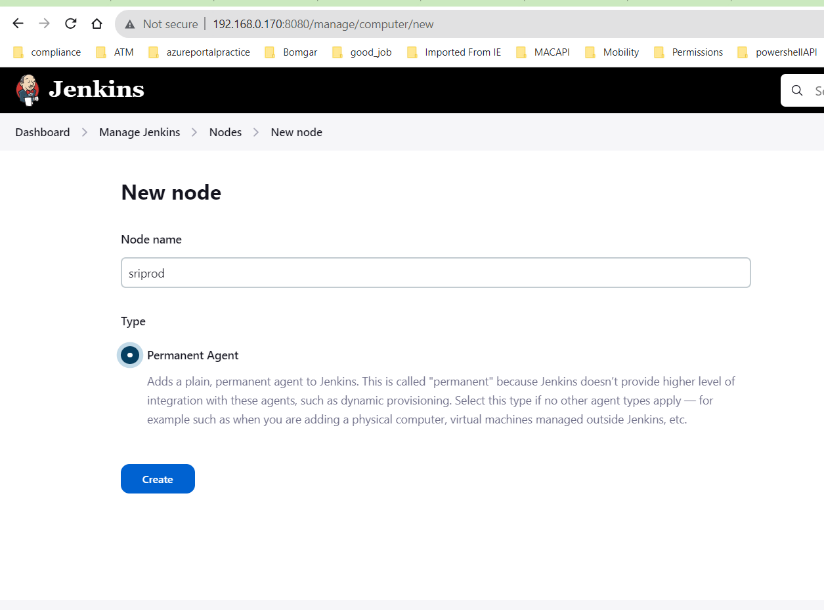


* Click on Manage Jenkins > nodes and cloud





* Click new node: Sritest and sriprod with permanent agent and update label for default node.
  + Please ensure to have same hostname to avoid any confusion



**Name :** Sriprod

**Description:** This is a Sriprod node to run the specific jobs on sriprod vm

**Number of executors:** 1 ( this is for jenkis can execute number of concert task on this node)

**Remote root directory:** /home/ubuntu/jenkins (this is where jenkins creates folder and runs the jobs)

**Labels:** sriprod ( this is important and it is a logical grouping of jenkins nodes, where we specific where to execute the job) a label can be associated with multiple nodes.

Labels (or tags) are used to group multiple agents into one logical group.

For example, if you have multiple Windows agents and you have a job that must run on Windows, then you could configure all your Windows agents to have the label windows, and then tie that job to this label.

This would ensure that your job runs on one of your Windows agents, but not on any agents without this label.

Labels do not necessarily have to represent the operating system on the agent; you can also use labels to note the CPU architecture, or that a certain tool is installed on the agent.

Multiple labels must be separated by a space. For example, windows docker would assign two labels to the agent: windows and docker.

Labels may contain any non-space characters, but you should avoid special characters such as any of these: !&|<>(), as other Jenkins features allow for defining label expressions, where these characters may be used.

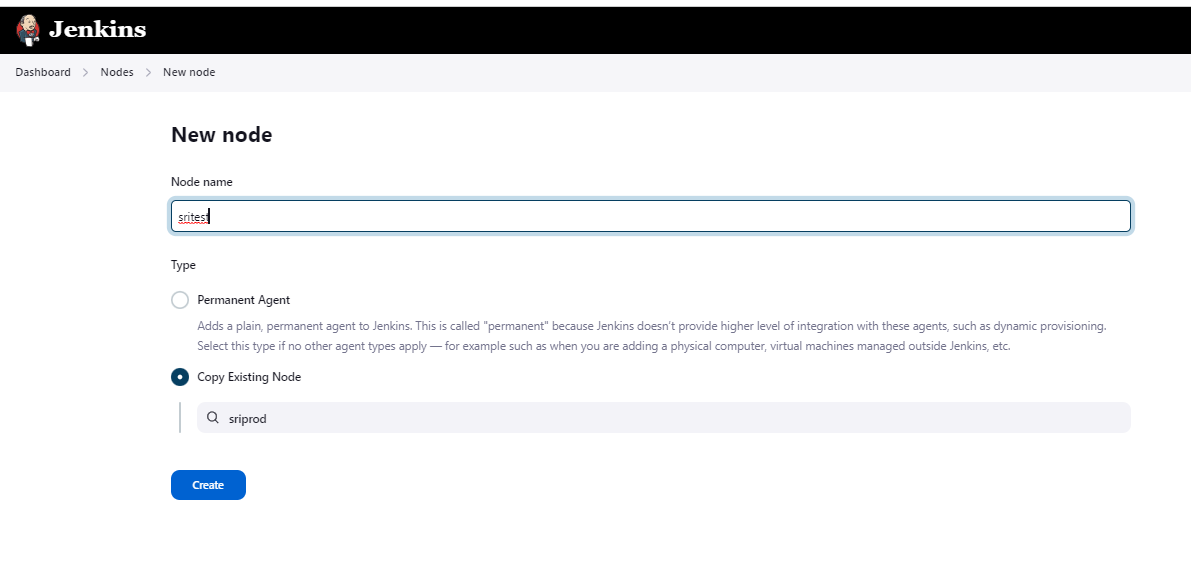
**Launch method:** launch via SSH

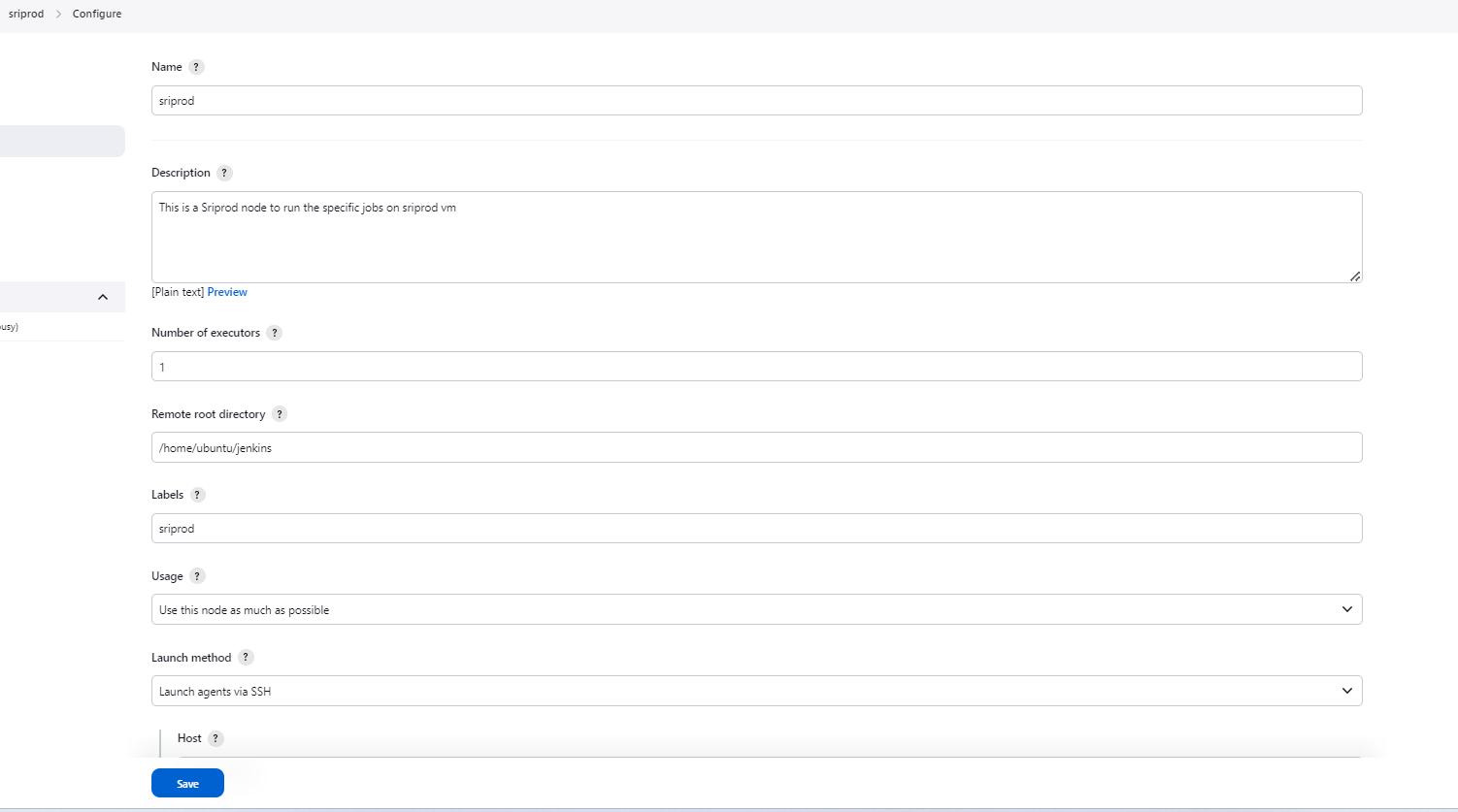
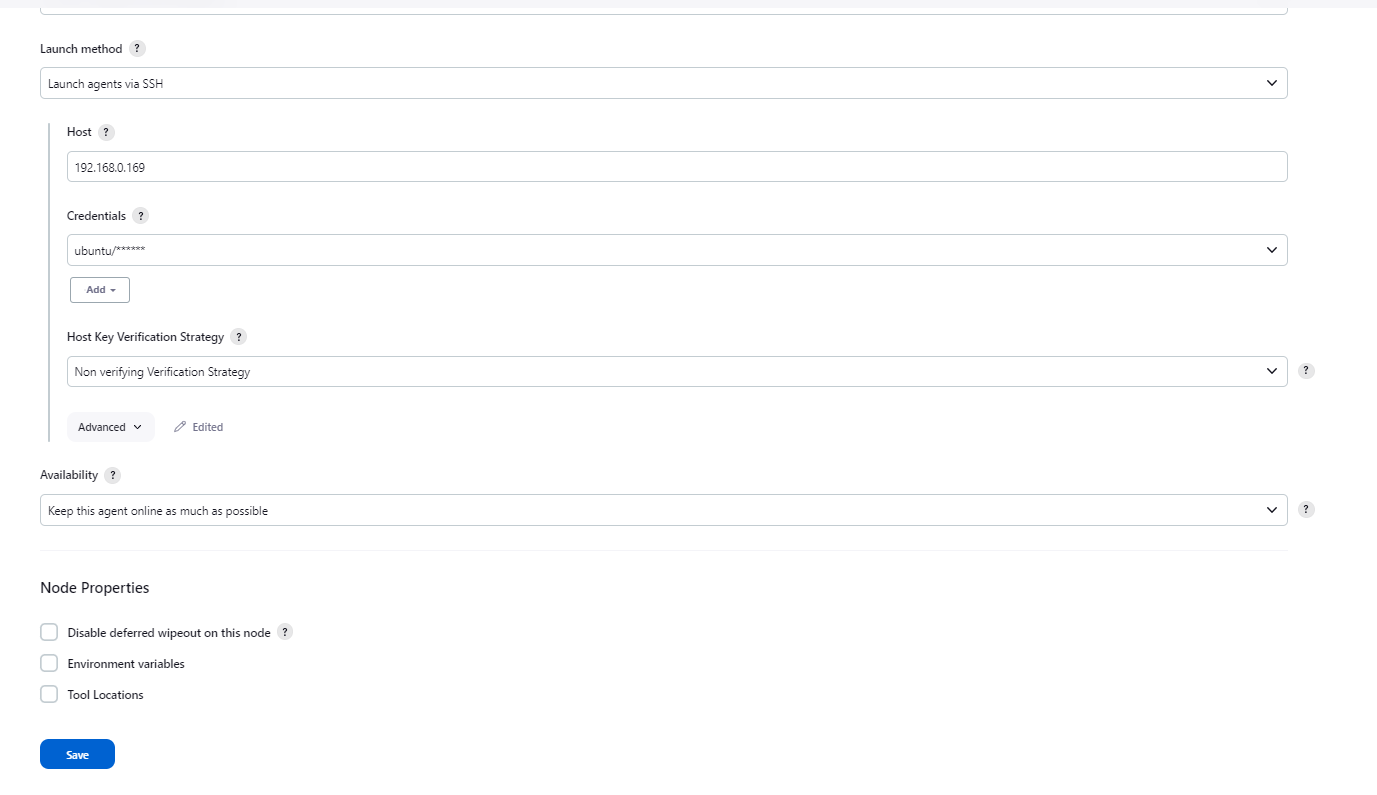
**Host :**192.168.0.171 (sriprod ip)

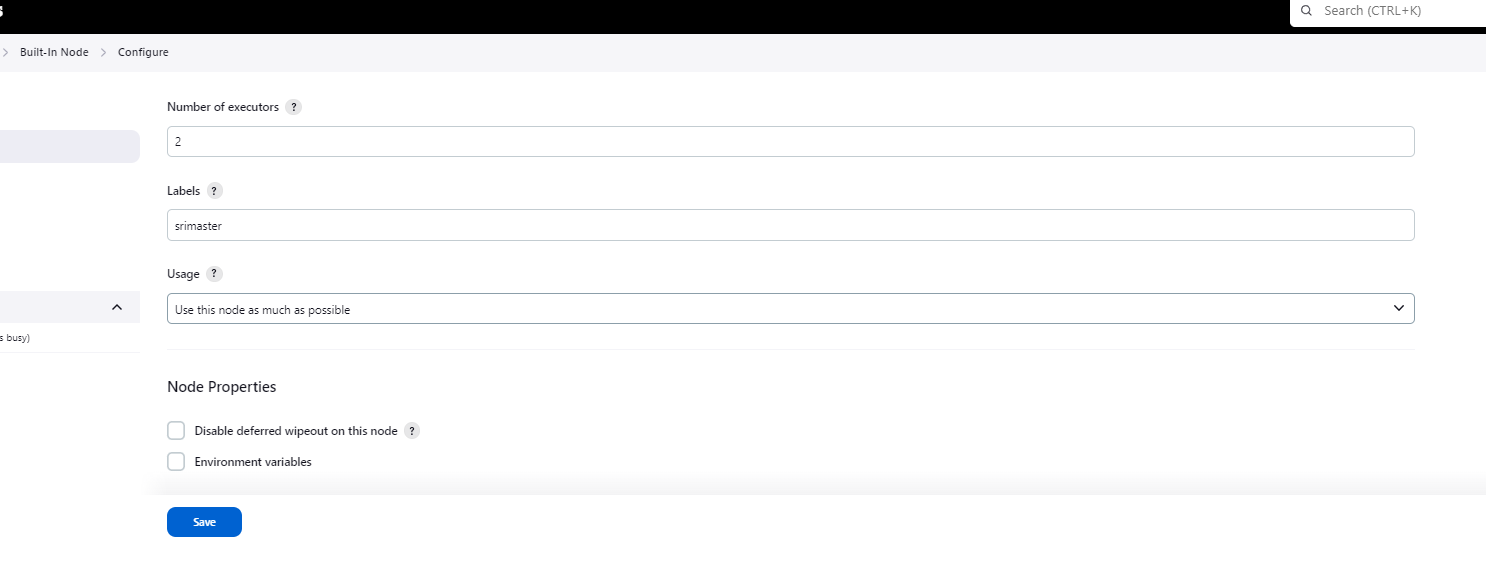
**Credintials :** select add and provide username and password “ubuntu” and select ubuntu from list

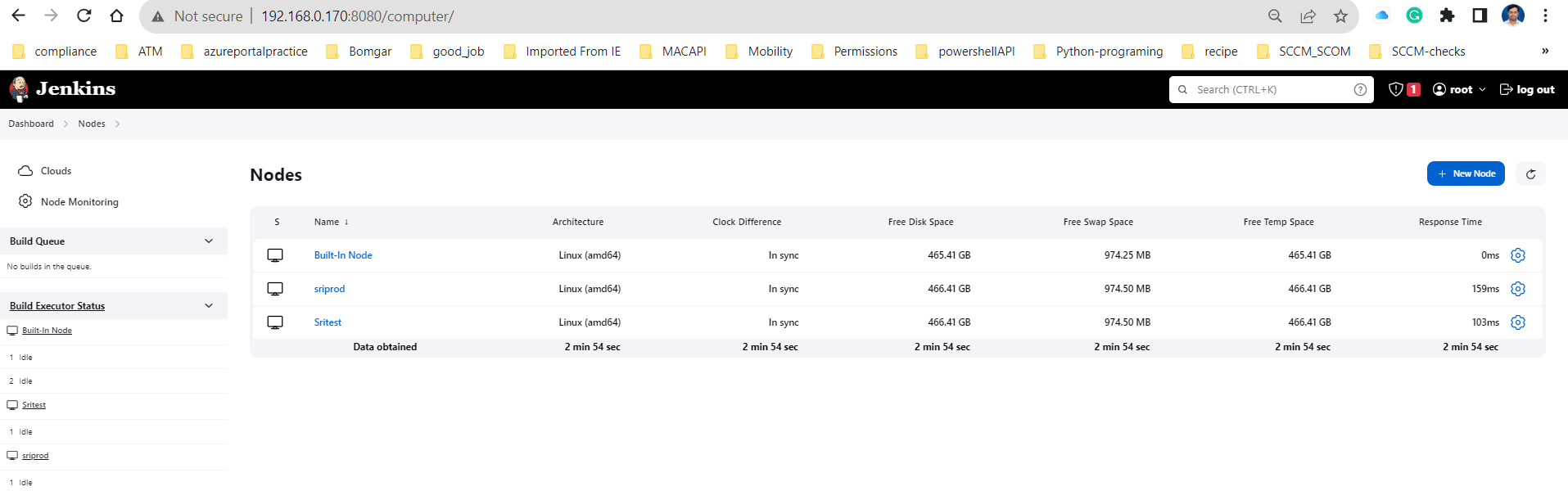
**Host key verifying strategies:** select no verifying verification strategy

* Click save. (Repeat same for sritest copy the file and change the IP Address, description, label to match)





All nodes active sriprod, sritest, srimaster(built-in node) labels. 



**SMEE Client and Githook configuration**

Since the project is running on private network which doesn’t have the public IP it is recommended to have smee client which will take care of port forwarding. If the server has public IP and ports are open, we can directly configure webhooks for Jenkins.

Please note to run the SMEE client is separate session and closing session will not run Jenkins jobs on git pushes. When a git push executed or change observed you would see post in session.

<https://smee.io/>

ubuntu@srimaster:~$ sudo snap install smee

smee 1.1.0 from Kyle Fazzari (kyrofa) installed

ubuntu@srimaster:~$ smee --url https://smee.io/9iOG0sG7OMp4sa --path /github-webhook/ --port 8080

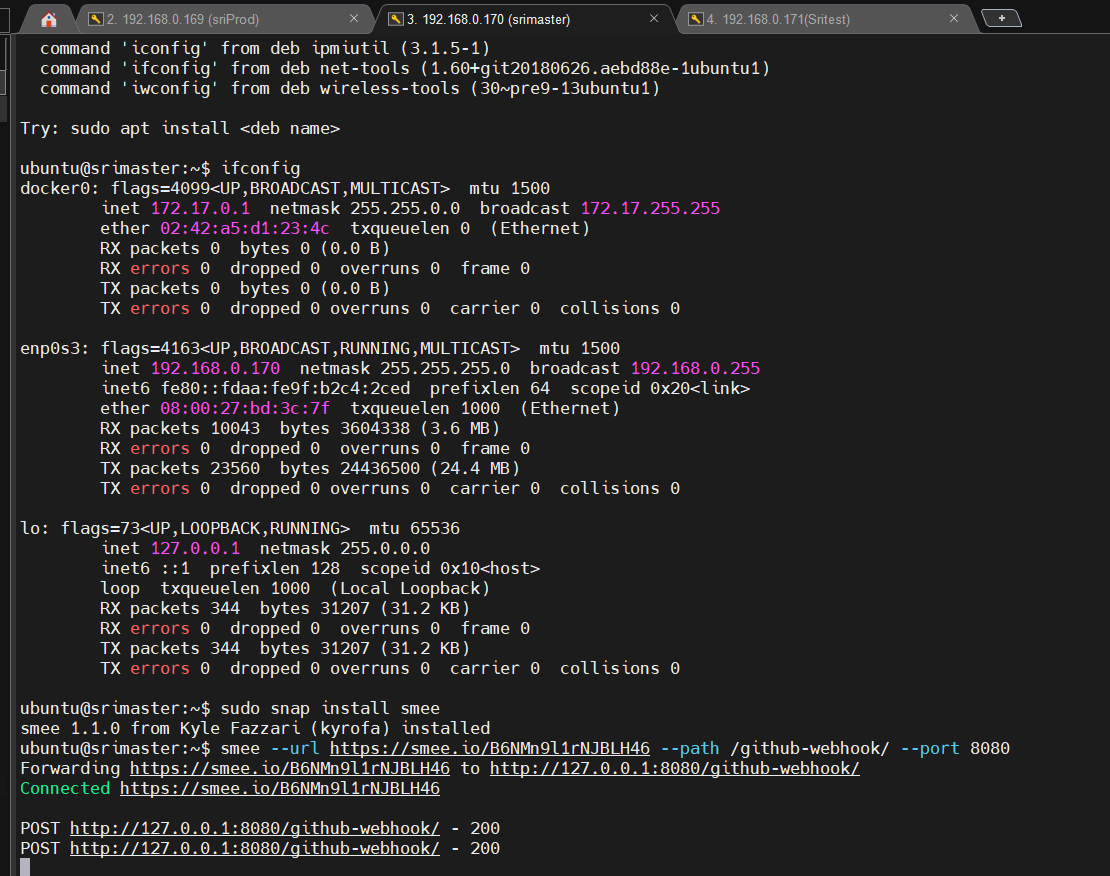
Forwarding https://smee.io/9iOG0sG7OMp4sa to <http://127.0.0.1:8080/github-webhook/>

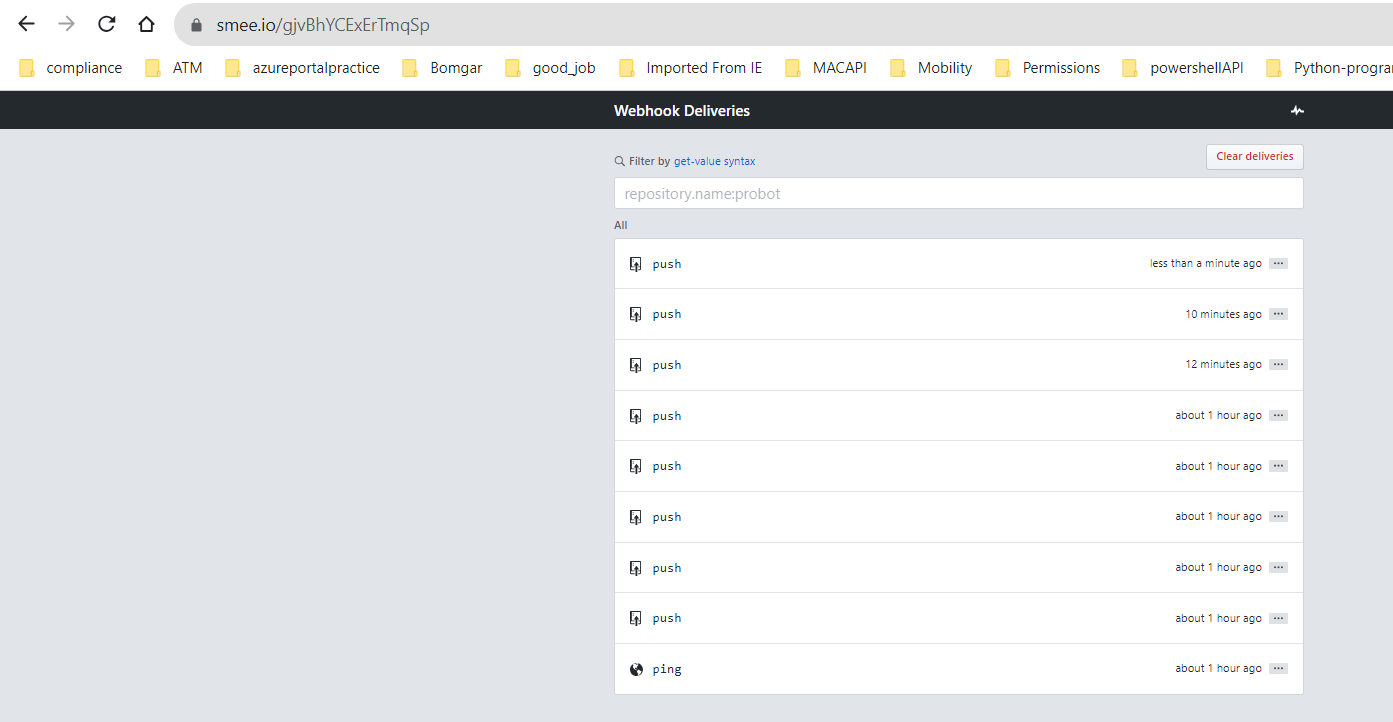
Connected https://smee.io/9iOG0sG7OMp4sa

POST <http://127.0.0.1:8080/github-webhook/> - 200

POST <http://127.0.0.1:8080/github-webhook/> - 200

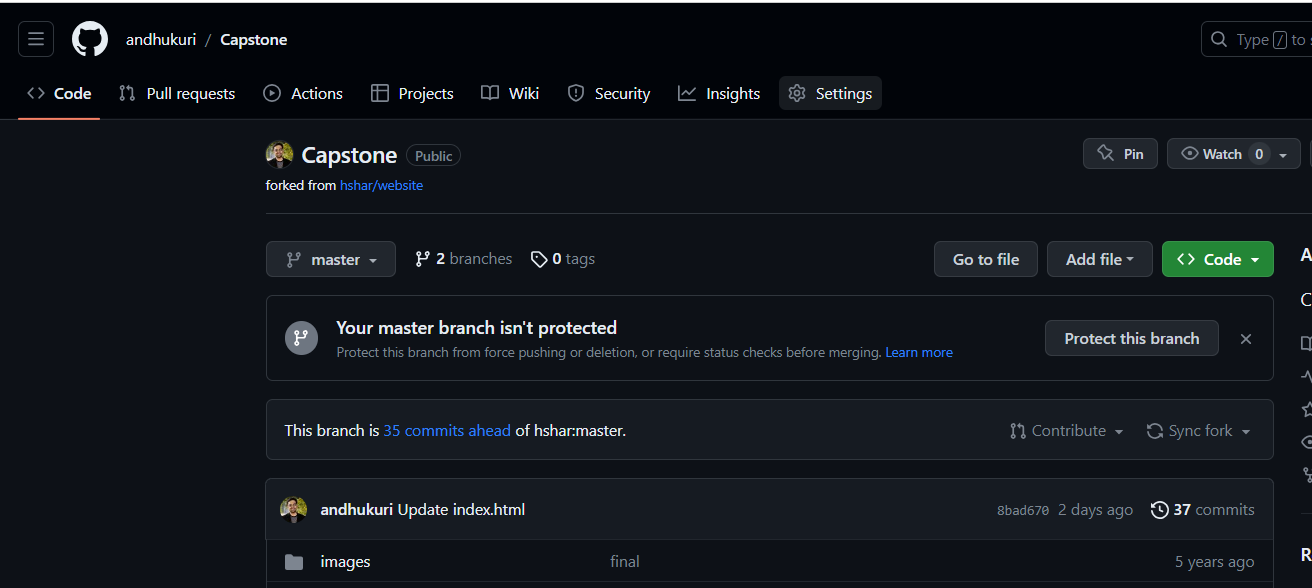
**Note:** URL in the screenshot is different and if the smee client session is closed in the srimaster, we need to generate and update new url

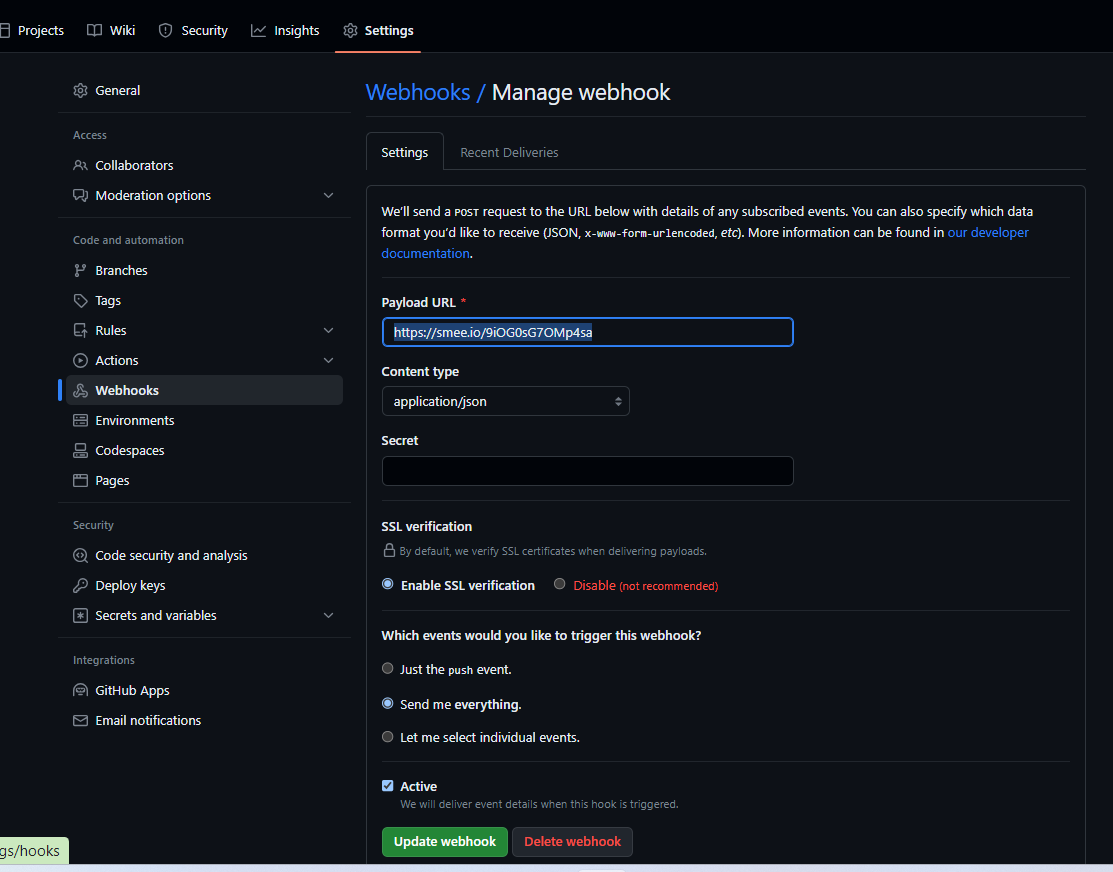


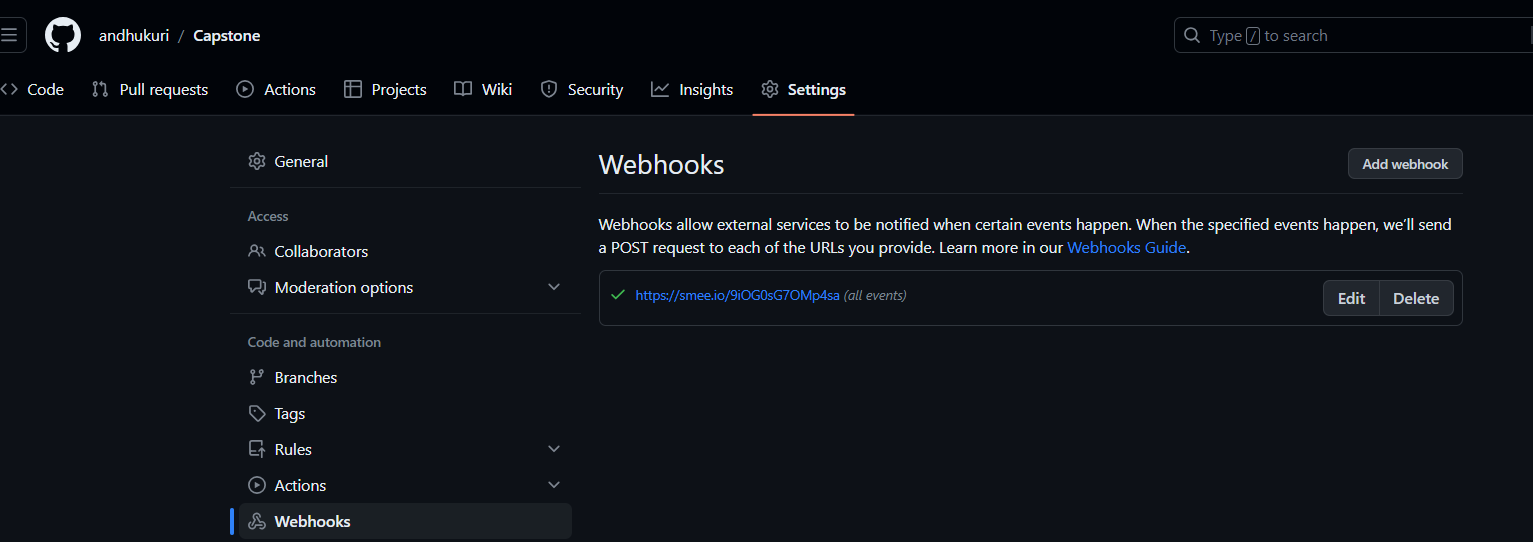


Configuring Githook for the capstone project

* Go to project settings > Webhook > add webhook

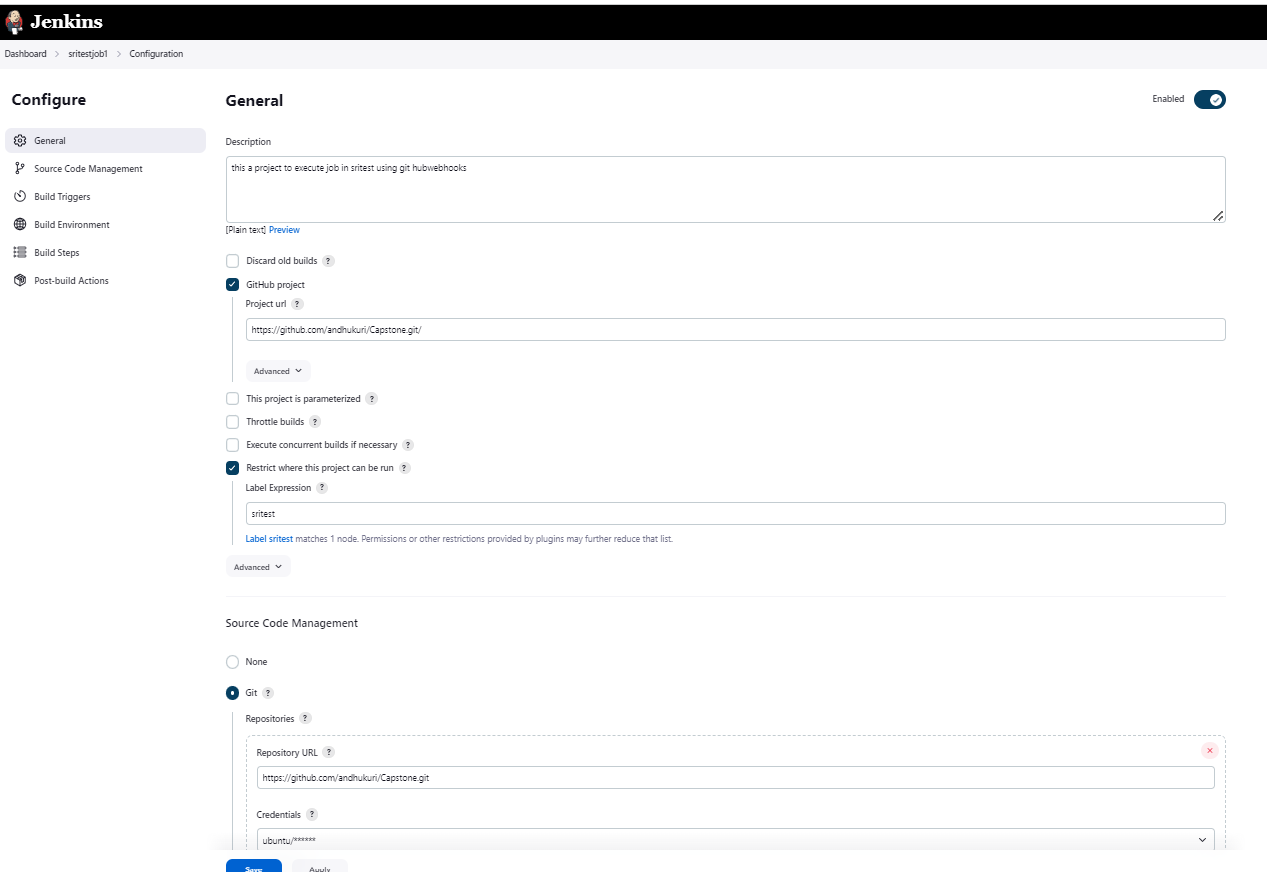


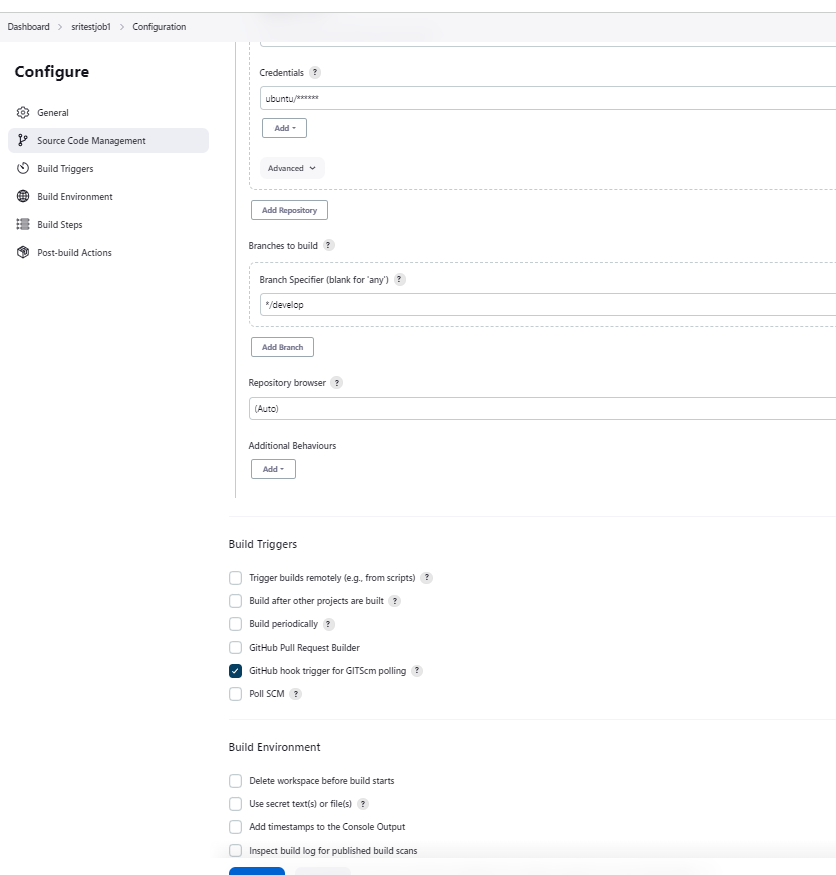


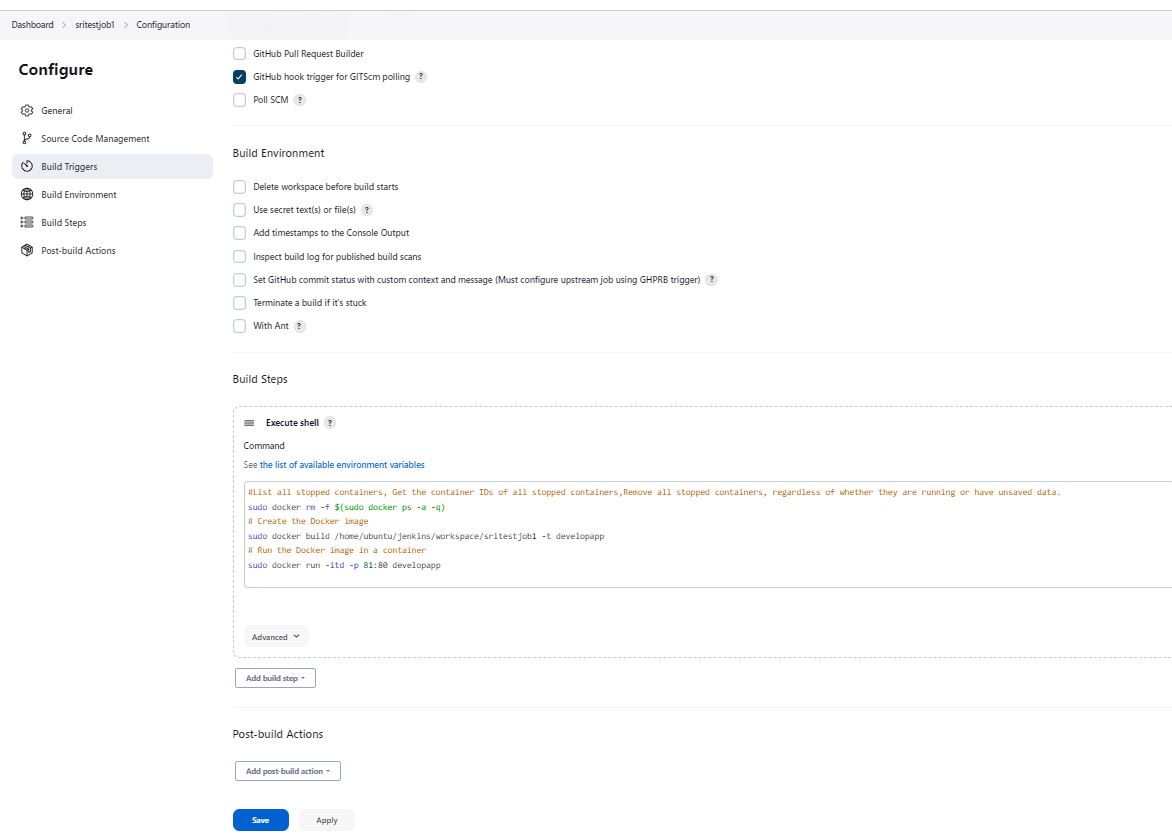


**Configuring Jenkins Jobs:**

**Job1:**

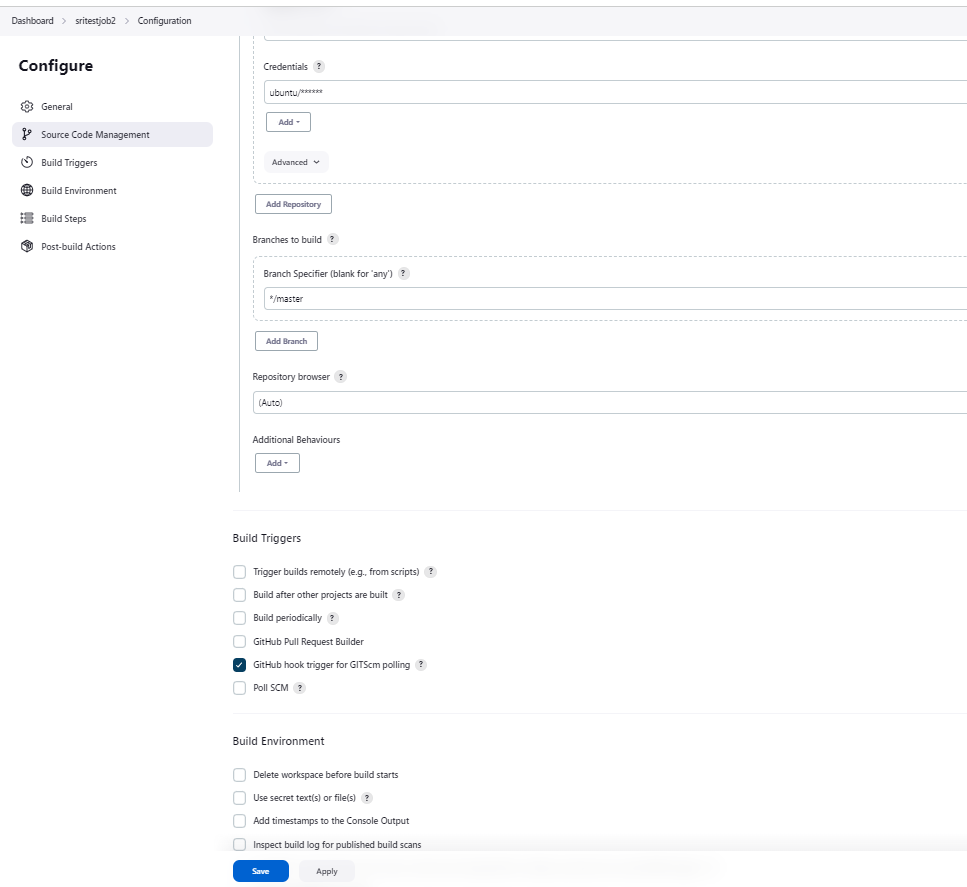


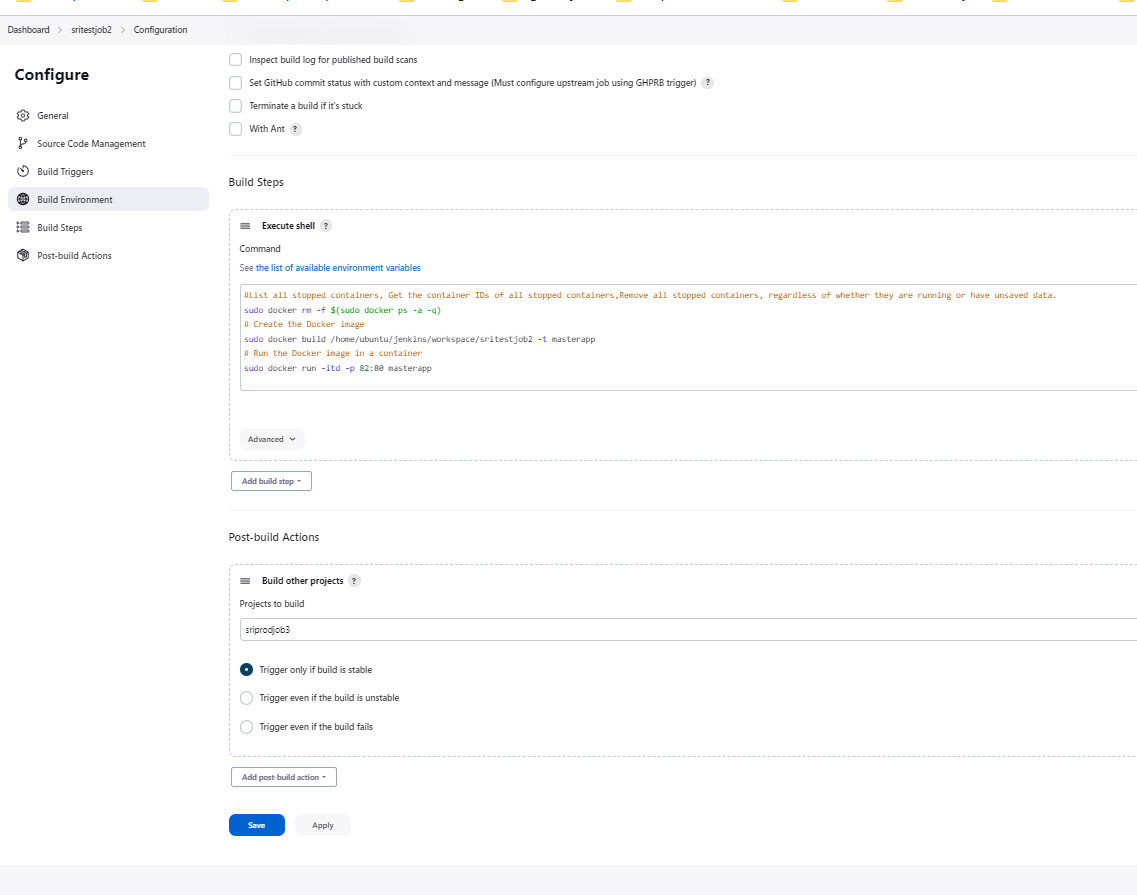




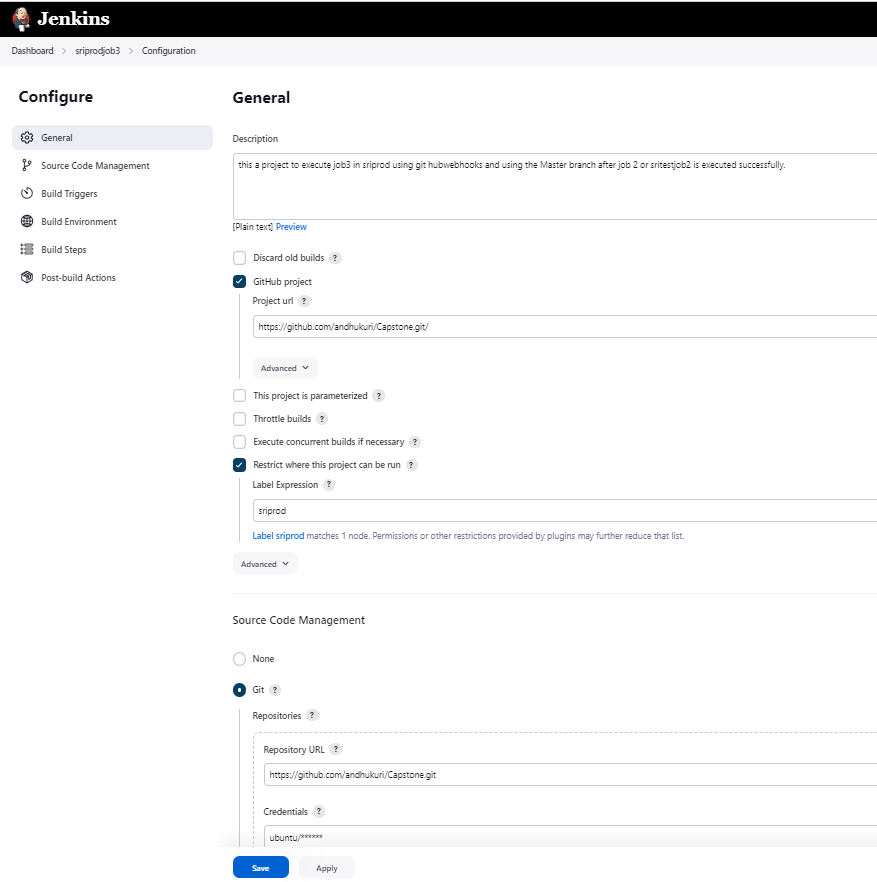
**Job2:**

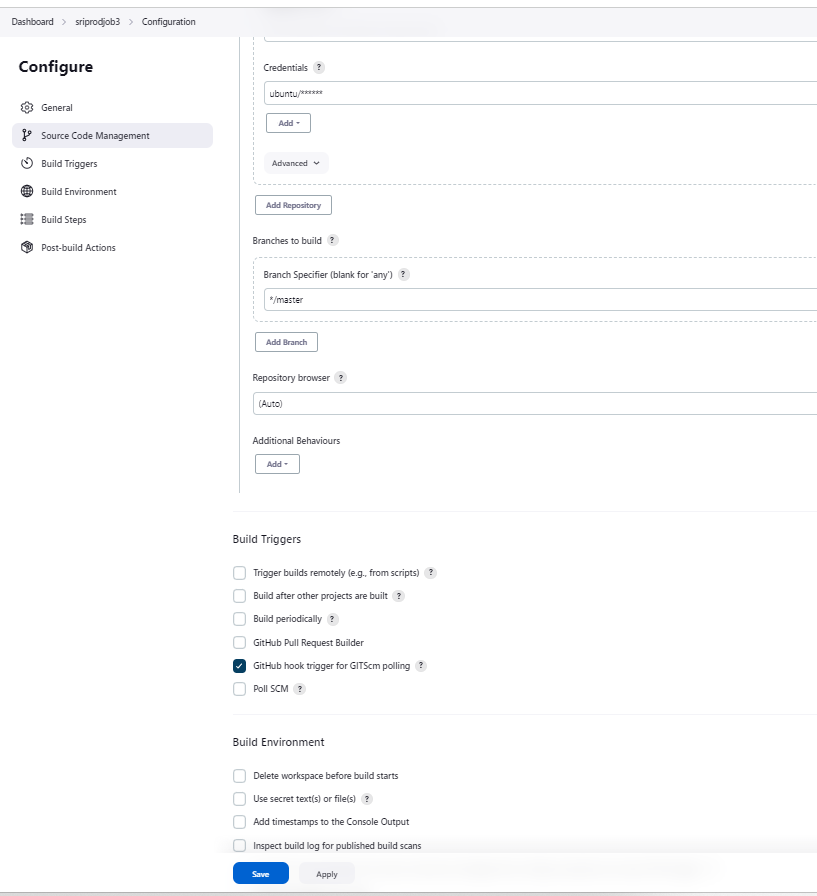


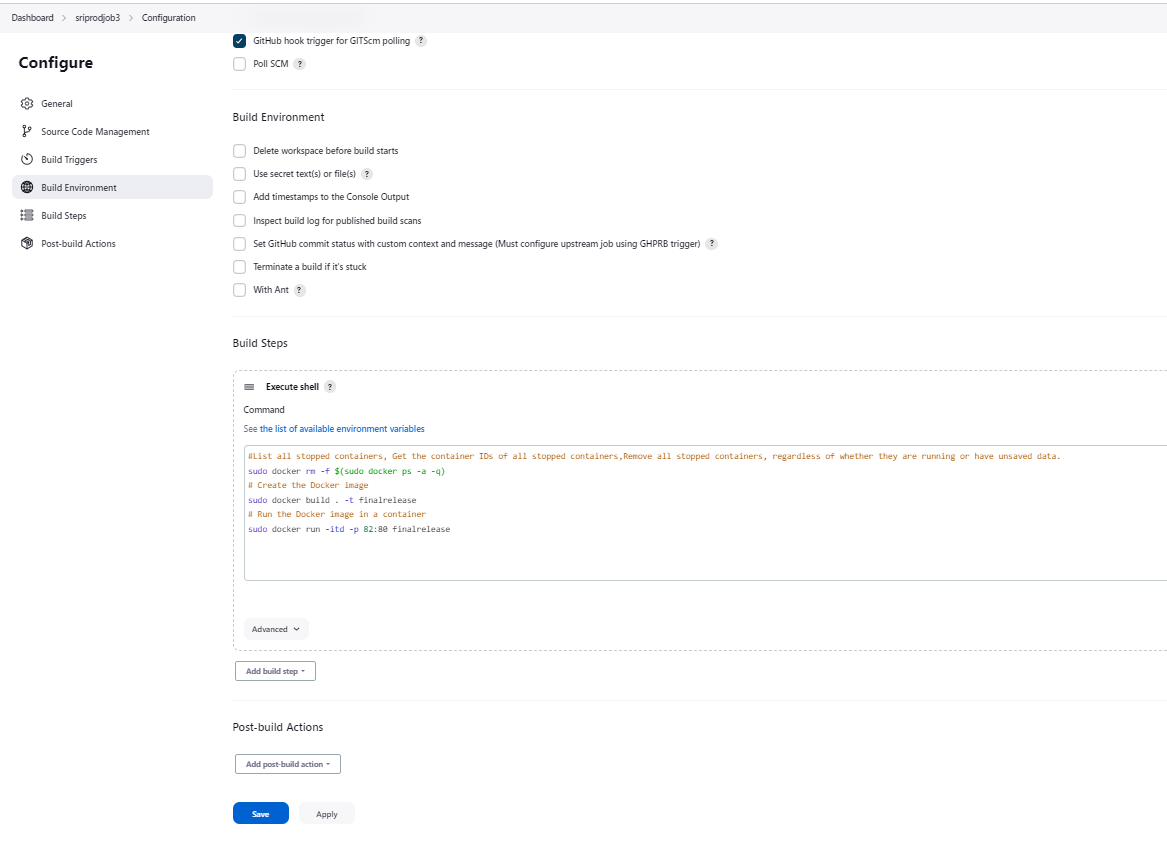




**Job3:**

****

****

****

**Configuration steps for executing the Jobs and validation.**

These are related to sritestjob1 or Job1. Try below different scenarios to check the status:  
#description: This a project to execute the job2 in Sritest using git hubwebhooks and using Master branch  
#git hub project: https://github.com/andhukuri/Capstone.git/  
#rescrict where this project can run: sritest  
#Branchspecifier: \*/ develop

**Senario:1:** Fail  
Executing the build step in Sritest for docker  
  
#Below is the build command (For Build Steps:)  
# Create the Docker image  
sudo docker build /home/ubuntu/jenkins/workspace/sritestjob1 -t developapp  
sudo docker run -it -d 81:80 developapp  
 **Senario:2** Success however port will be used by the container sritestjob1  
In the previous setup docker image creation error occurred due to -d changing the build setup  
using -p instead of -d and also the workspace directory for job one was sritestjob**-1**  
(-1) is creating problems due to Jenkins switches, creating new job sritestjob1 copying existing  
Also, ensure that the dockerfile follows standards with caps and normal letters for commands.  
#Below is the build command and dockerfile will be used for Apache installation and file copy.

# Create the Docker image  
sudo docker build /home/ubuntu/jenkins/workspace/sritestjob1 -t developapp  
# Run the Docker image in a container  
sudo docker run -itd -p 81:80 developapp

**Senario3**: Deleting the allocated port and re-assessing to docker build  
#Below is the build command and dockerfile will be used for Apache installation and file copy.

#List all stopped containers, Get the container IDs of all stopped containers,Remove all stopped containers, regardless of whether they are running or have unsaved data.

sudo docker rm -f $(sudo docker ps -a -q)  
# Create the Docker image  
sudo docker build /home/ubuntu/jenkins/workspace/sritestjob1 -t developapp  
# Run the Docker image in a container  
sudo docker run -itd -p 81:80 developapp

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Checking to execute sritestjob2 or job2 on master branch after commit

#Create Jenkins **sritestjob2** (you can copy from sritestjob1 and make changes for git branch specifier and description, docker build port is 82 to avoid conflict with test environment)

#description: This a project to execute the job2 in Sritest using git hubwebhooks and using Master branch  
#git hub project: https://github.com/andhukuri/Capstone.git/  
#rescrict where this project can run: sritest  
#Branchspecifier: \*/master

#Below is the build command and dockerfile will be used for Apache installation and file copy.

#List all stopped containers, Get the container IDs of all stopped containers, Remove all stopped containers, regardless of whether they are running or have unsaved data.

sudo docker rm -f $(sudo docker ps -a -q)  
# Create the Docker image  
sudo docker build /home/ubuntu/jenkins/workspace/sritestjob2 -t masterapp  
# Run the Docker image in a container  
sudo docker run -itd -p 82:80 masterapp

#After successful run of job2 add post build steps to run job3 for final release

#Select build another project and in project to build select sriprodjob3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#sriprodjob3 or Job 3

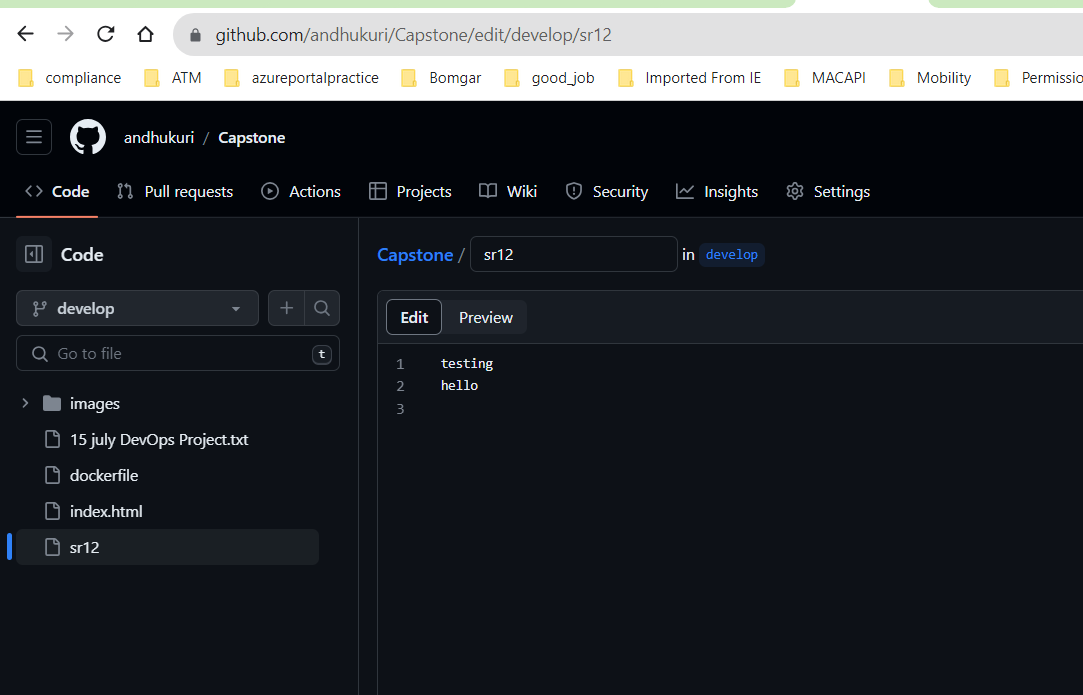
#Description: this a project to execute job3 in sriprod using git hubwebhooks and using the Master branch after job 2 or sritestjob2 is executed successfully.  
#git hub project : <https://github.com/andhukuri/Capstone.git/> or your own repository.  
#rescrict where this project can run: sriprod  
#Branchspecifier: \*/master

#Below is the build command and dockerfile will be used for Apache installation and file copy.

#List all stopped containers, Get the container IDs of all stopped containers, Remove all stopped containers, regardless of whether they are running or have unsaved data.  
sudo docker rm -f $(sudo docker ps -a -q)  
# Create the Docker image  
sudo docker build . -t finalrelease  
# Run the Docker image in a container  
sudo docker run -itd -p 82:80 finalrelease

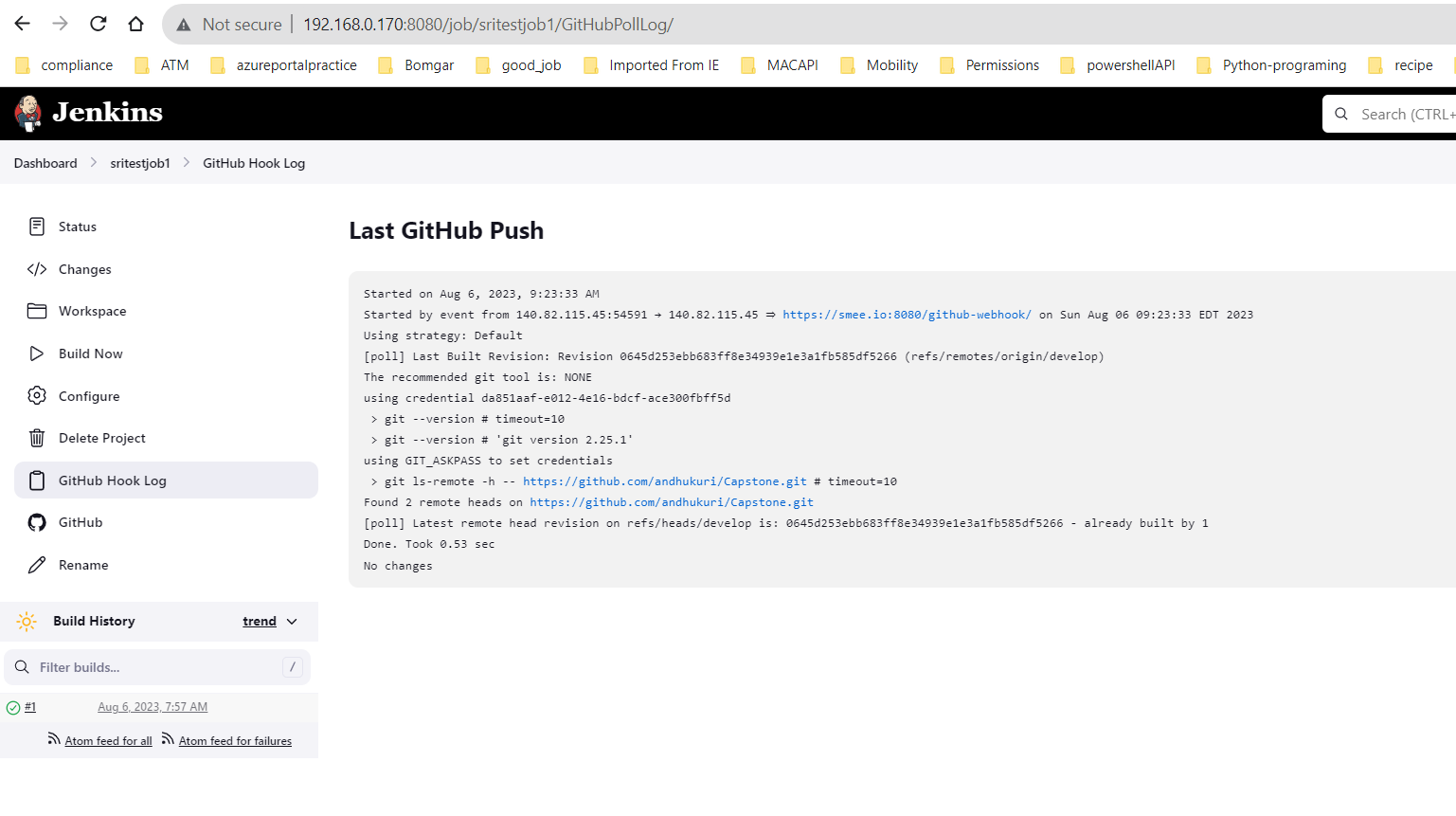
The below file is already created this is for reference only.

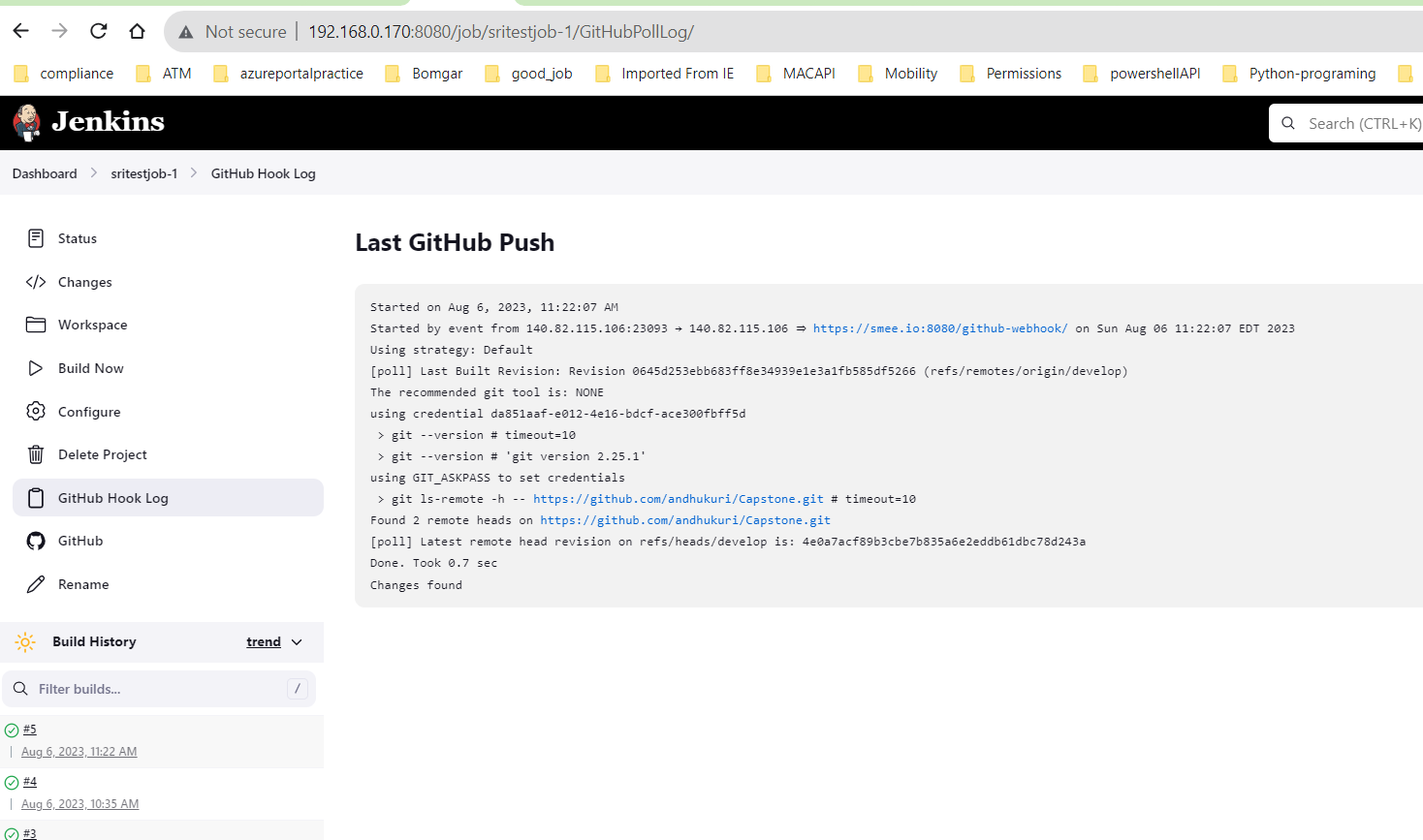
Dockefile   
FROM ubuntu  
RUN apt update  
RUN apt install apache2 -y  
ADD . /var/www/html  
ENTRYPOINT apachectl -D FOREGROUND  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Once the git hook is connected it will show in github hook log in Jenkins.

Sritestjob1 is for develop branch. It only triggers build when there is a change in develop Branch.

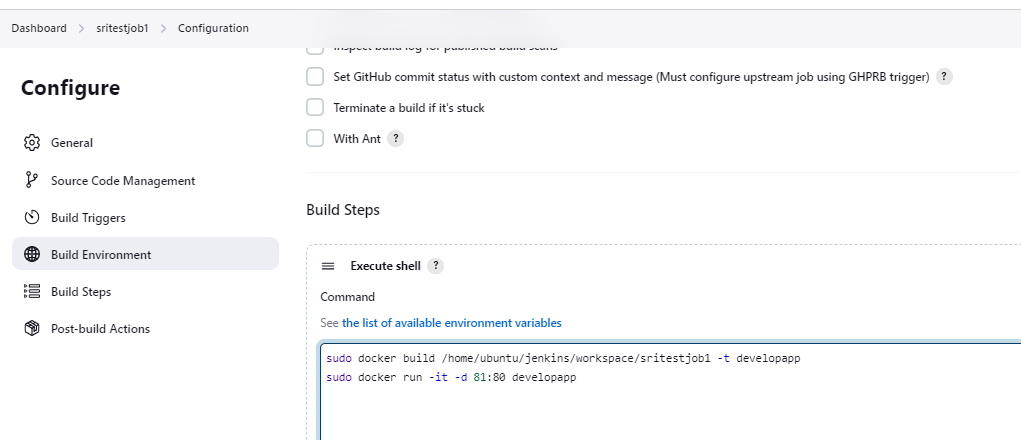


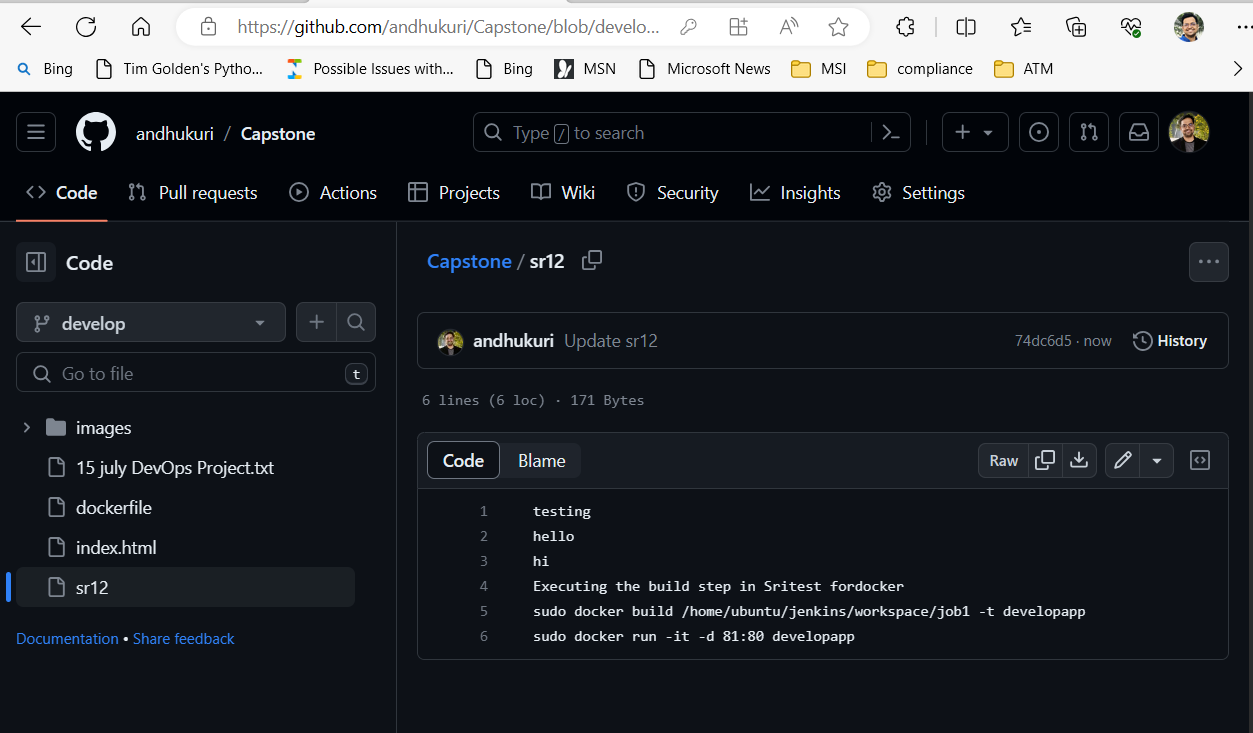


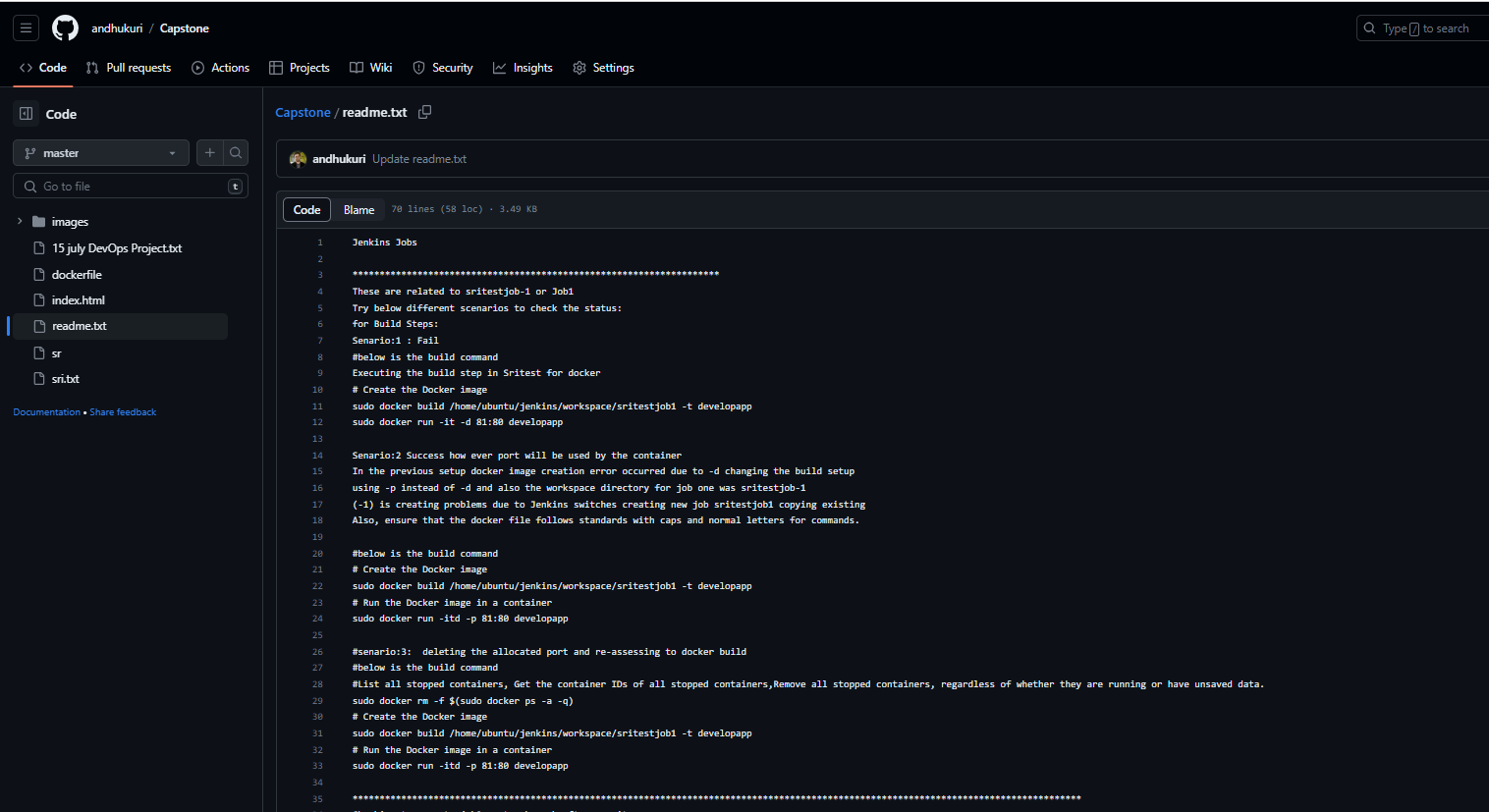
**SRITESTJOB1 OR JOB1**

**Senario:1:** Fail  
Executing the build step in Sritest for docker  
  
#Below is the build command (For Build Steps:)  
# Create the Docker image  
sudo docker build /home/ubuntu/jenkins/workspace/sritestjob1 -t developapp  
sudo docker run -it -d 81:80 developapp

Click on save and build now

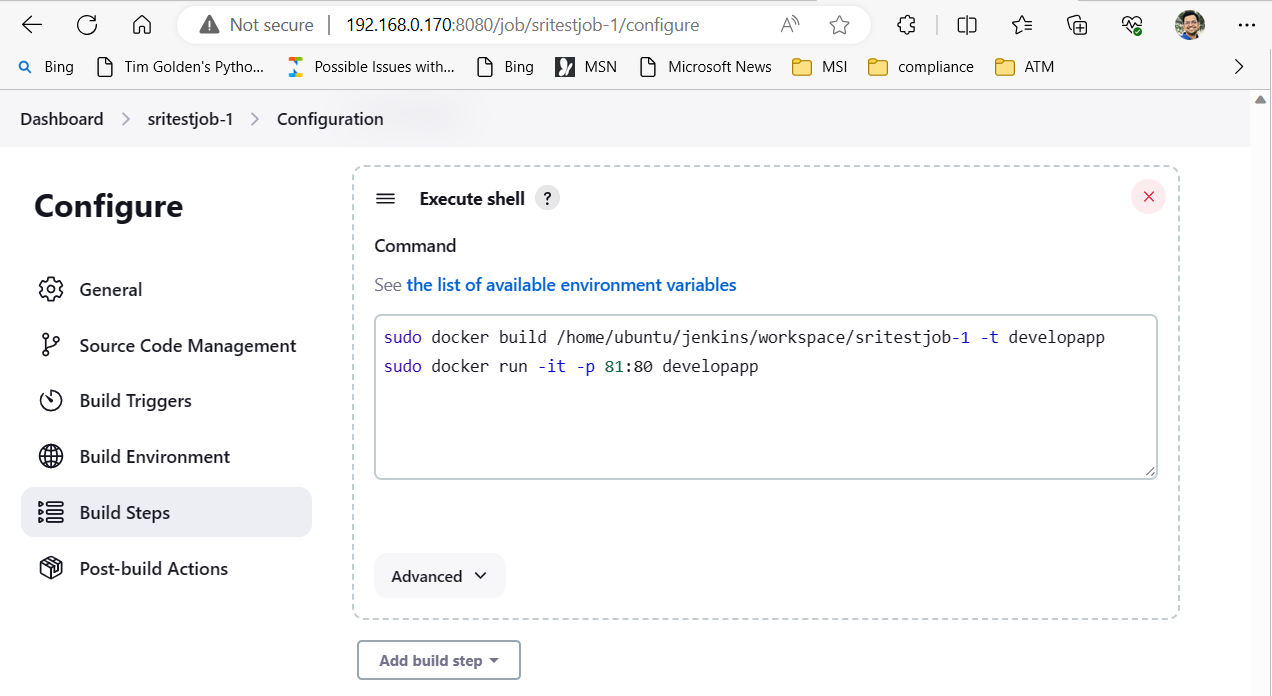


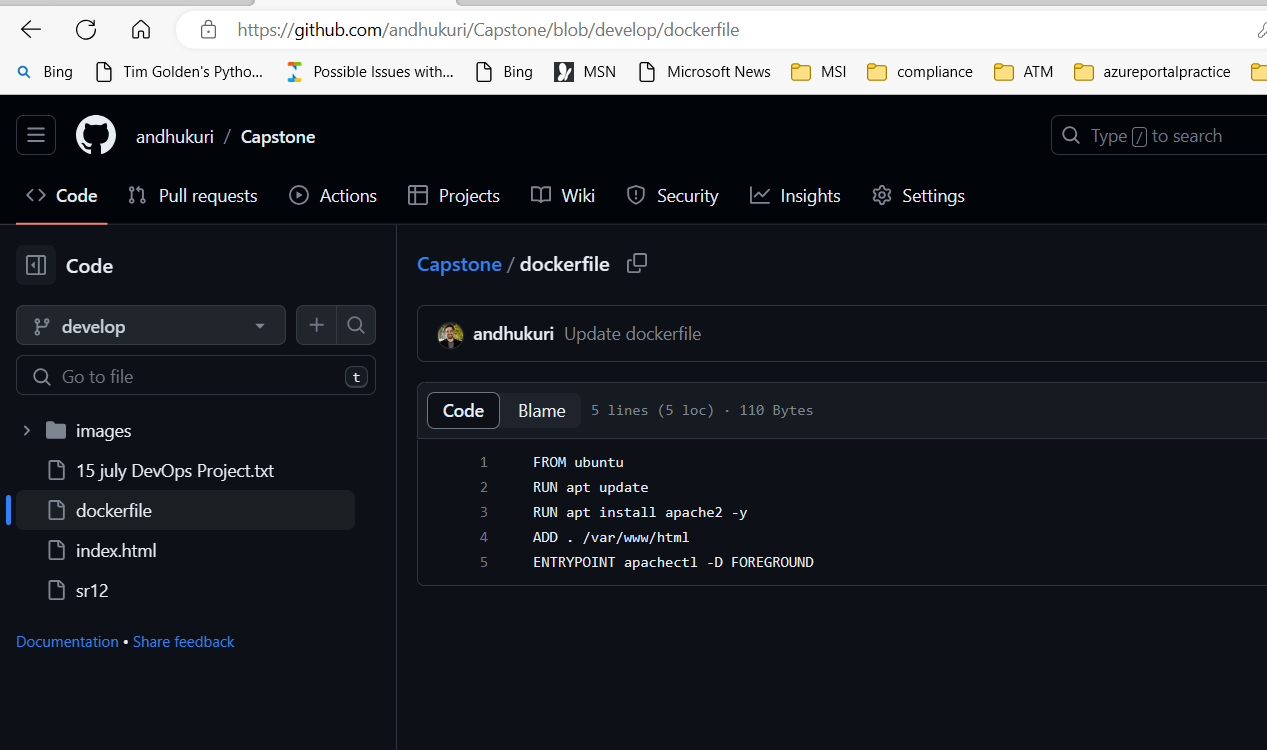




**Senario:2** Success however port will be used by the container sritestjob1  
In the previous setup docker image creation error occurred due to -d changing the build setup  
using -p instead of -d and also the workspace directory for job one was sritestjob**-1**  
(-1) is creating problems due to Jenkins switches, creating new job sritestjob1 copying existing  
Also, ensure that the dockerfile follows standards with caps and normal letters for commands.  
#Below is the build command and dockerfile will be used for Apache installation and file copy.

# Create the Docker image  
sudo docker build /home/ubuntu/jenkins/workspace/sritestjob1 -t developapp  
# Run the Docker image in a container  
sudo docker run -itd -p 81:80 developapp

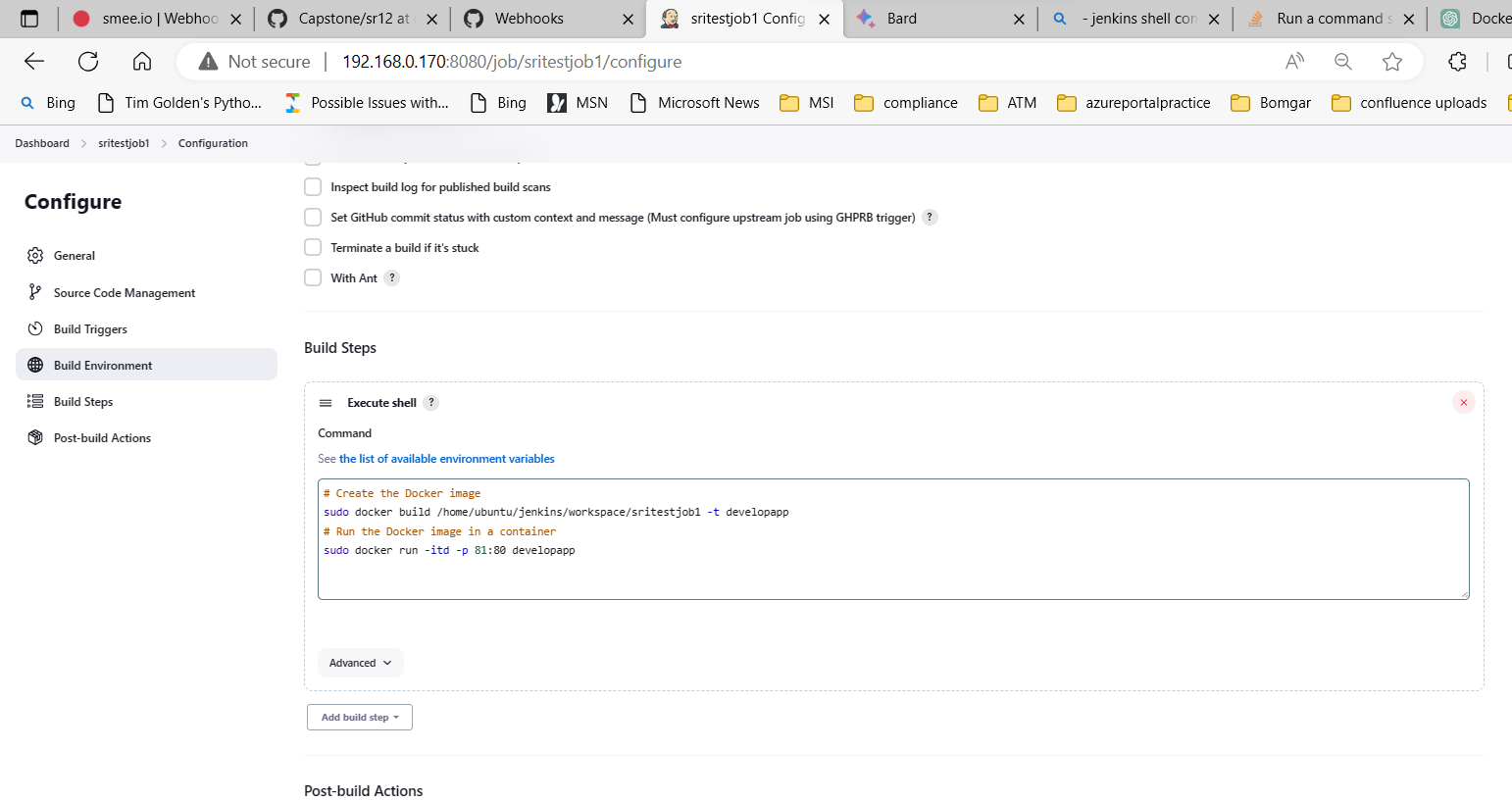


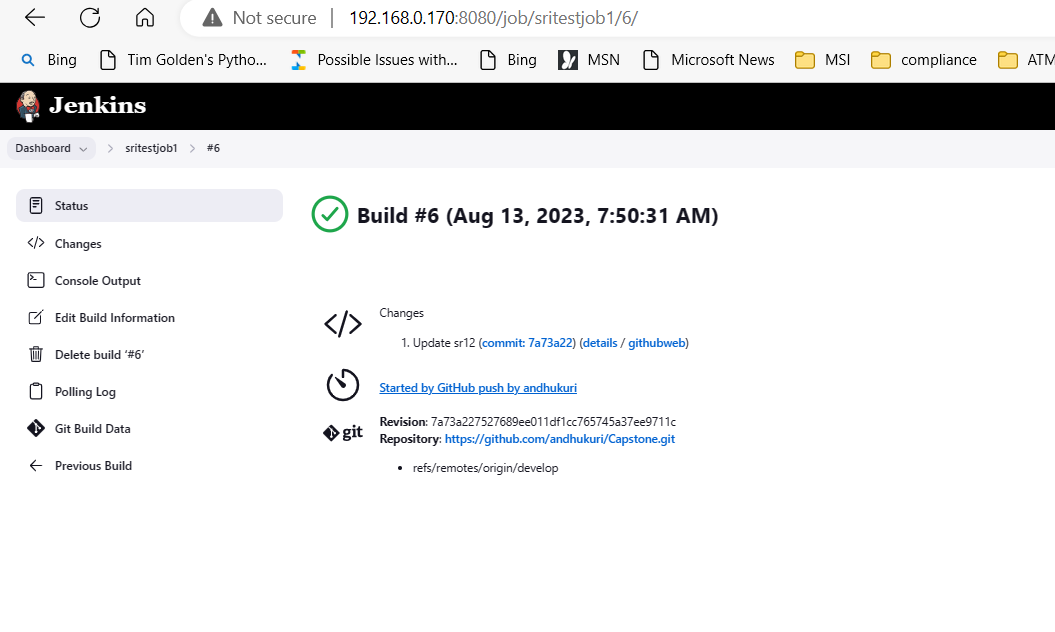


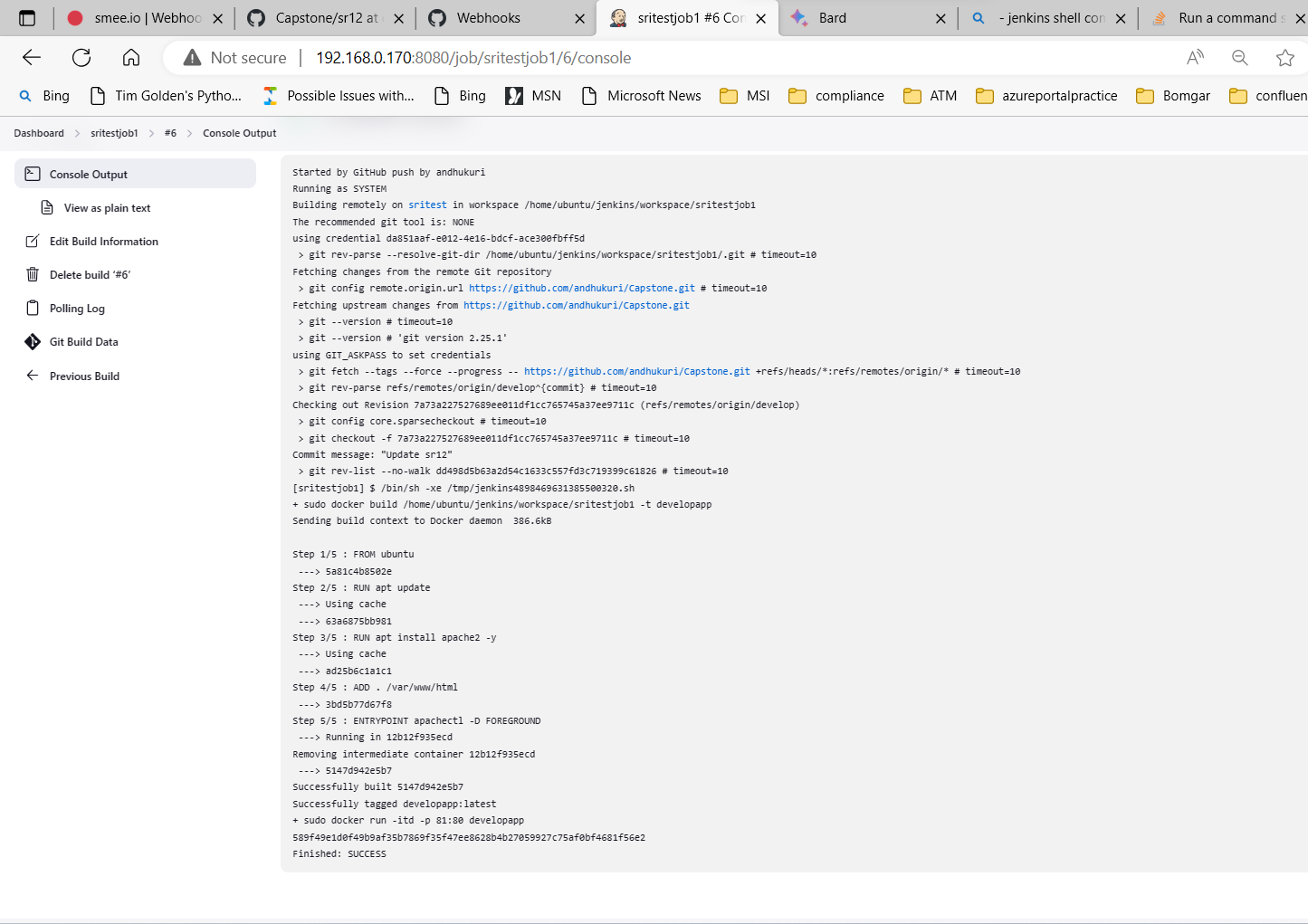
**Senario3**: Deleting the allocated port and re-assessing to docker build  
#Below is the build command and dockerfile will be used for Apache installation and file copy.

#List all stopped containers, Get the container IDs of all stopped containers,Remove all stopped containers, regardless of whether they are running or have unsaved data.

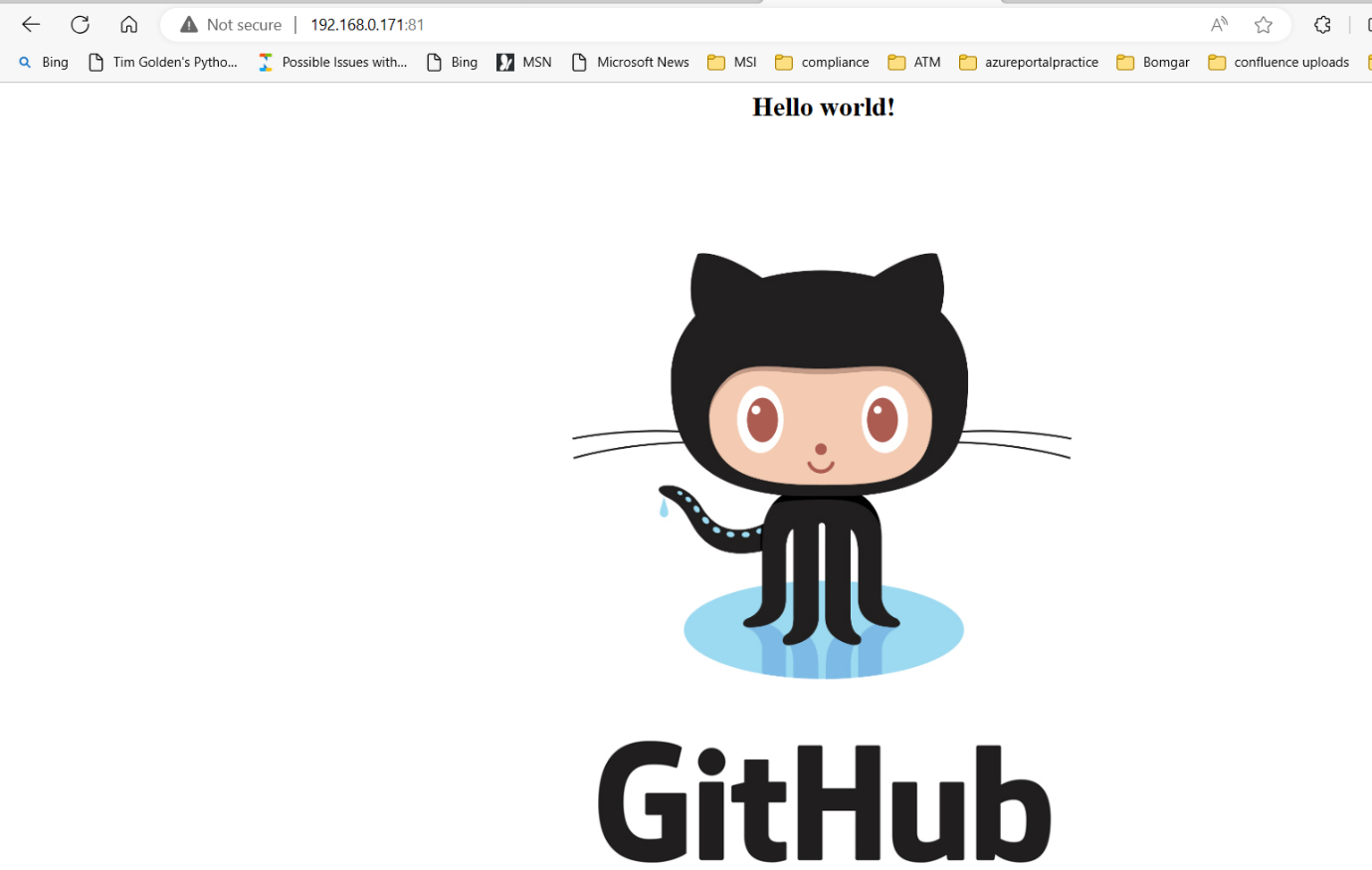
sudo docker rm -f $(sudo docker ps -a -q)  
# Create the Docker image  
sudo docker build /home/ubuntu/jenkins/workspace/sritestjob1 -t developapp  
# Run the Docker image in a container  
sudo docker run -itd -p 81:80 developapp

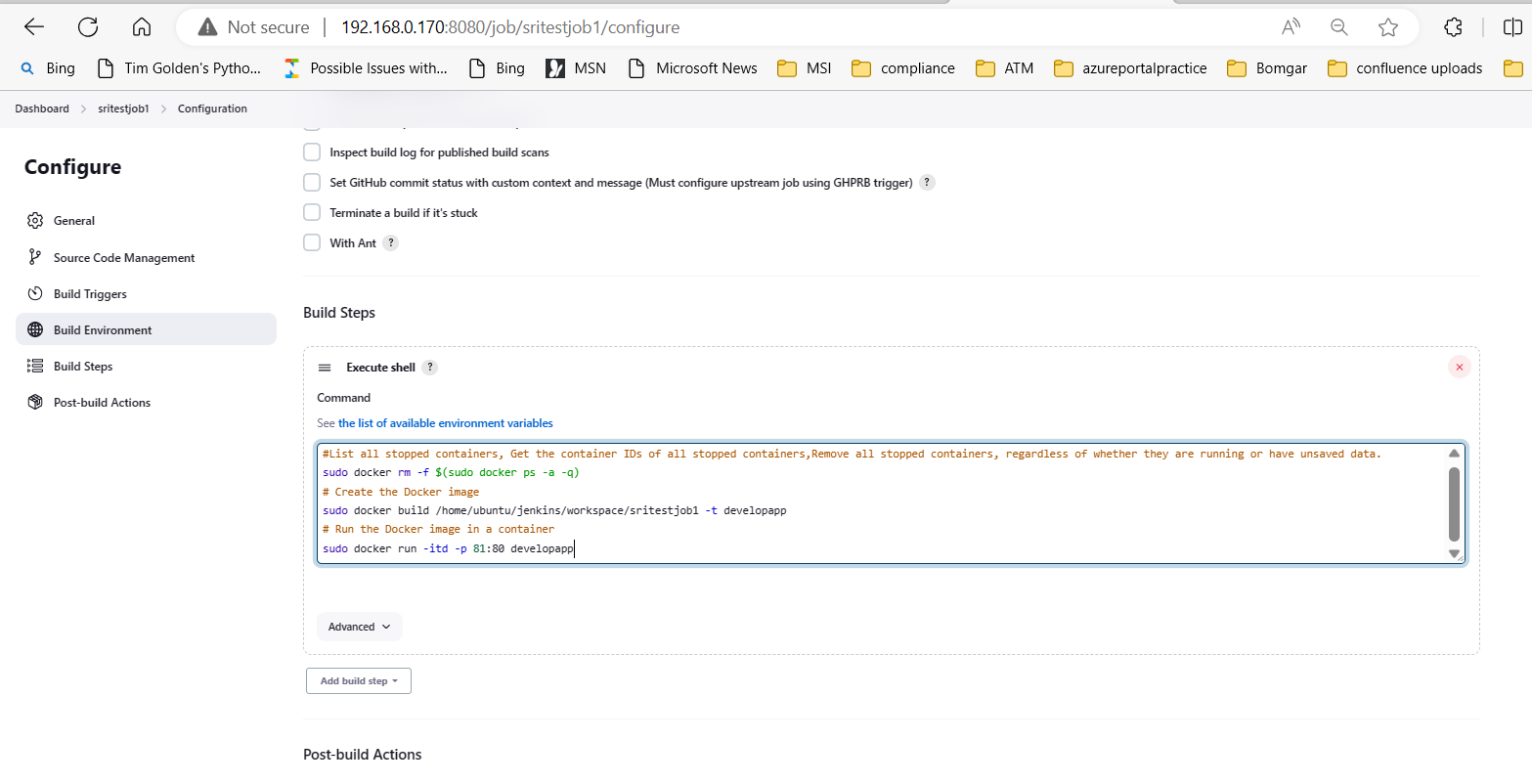


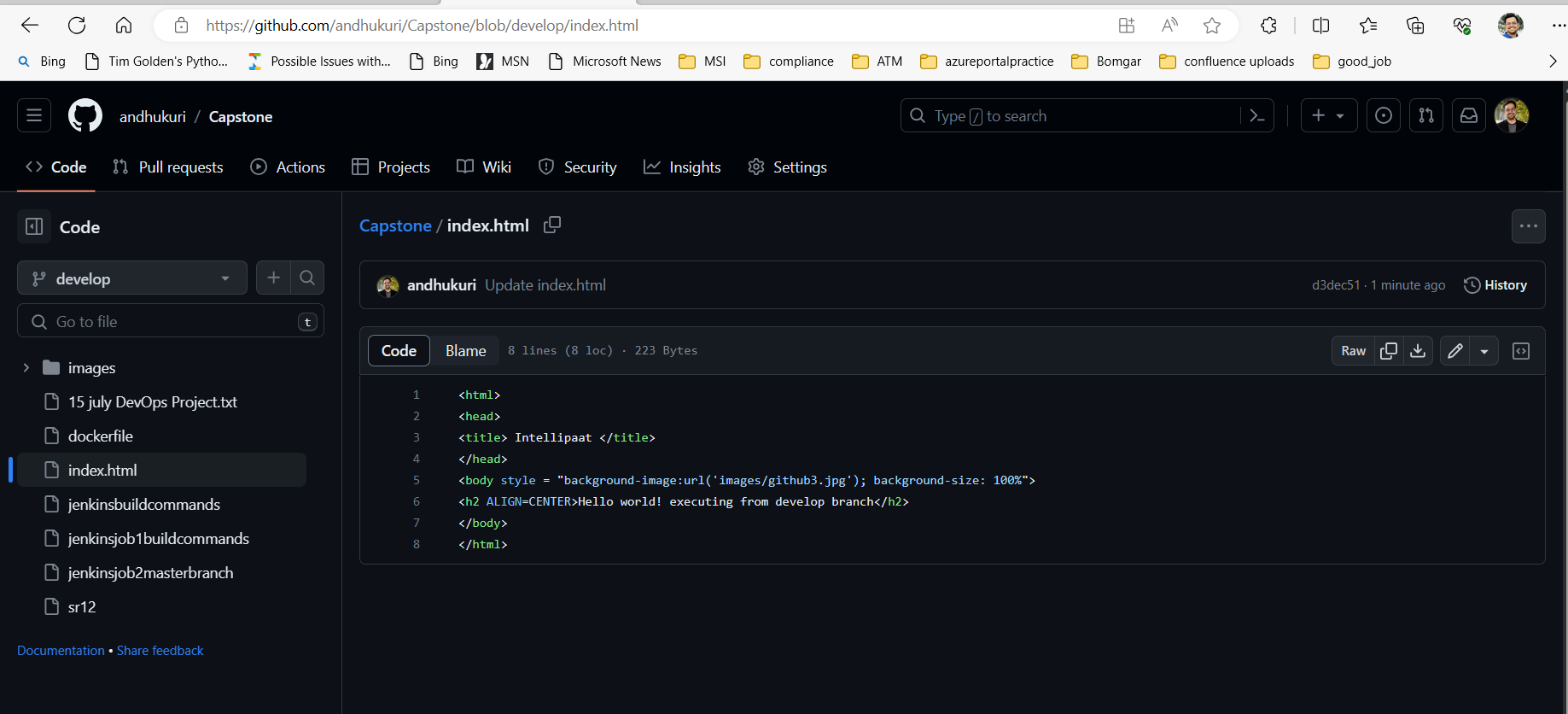


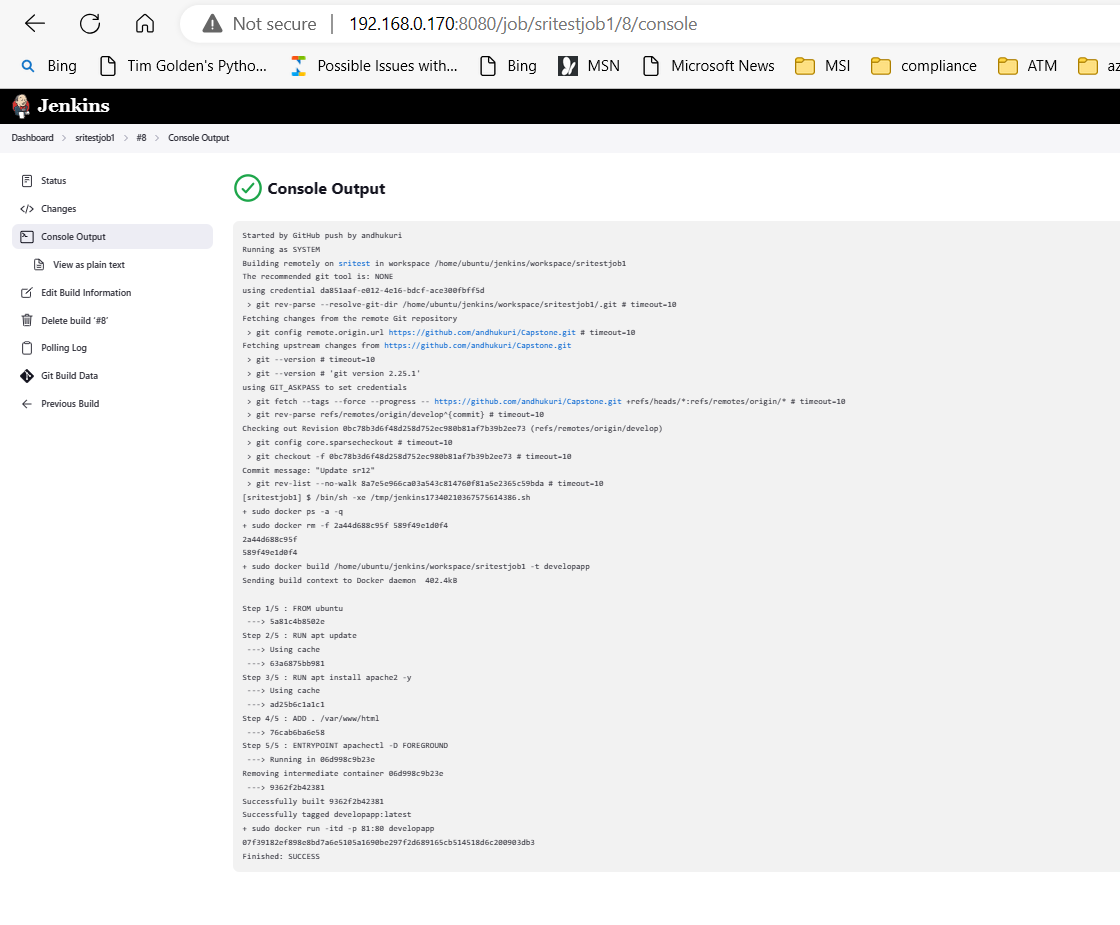


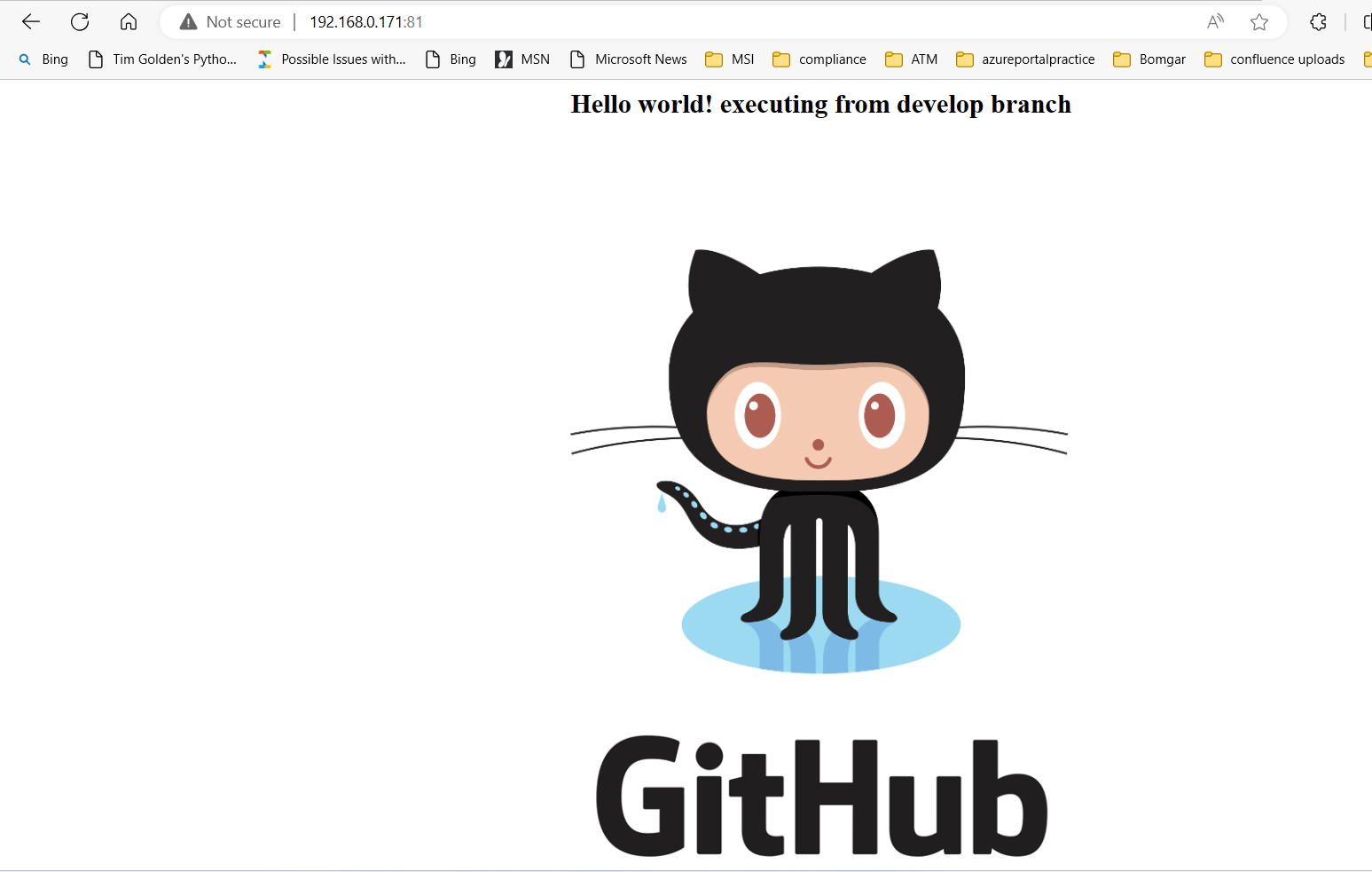
Browser sritest server, since the container has been created in it: 192.168.0.171:81









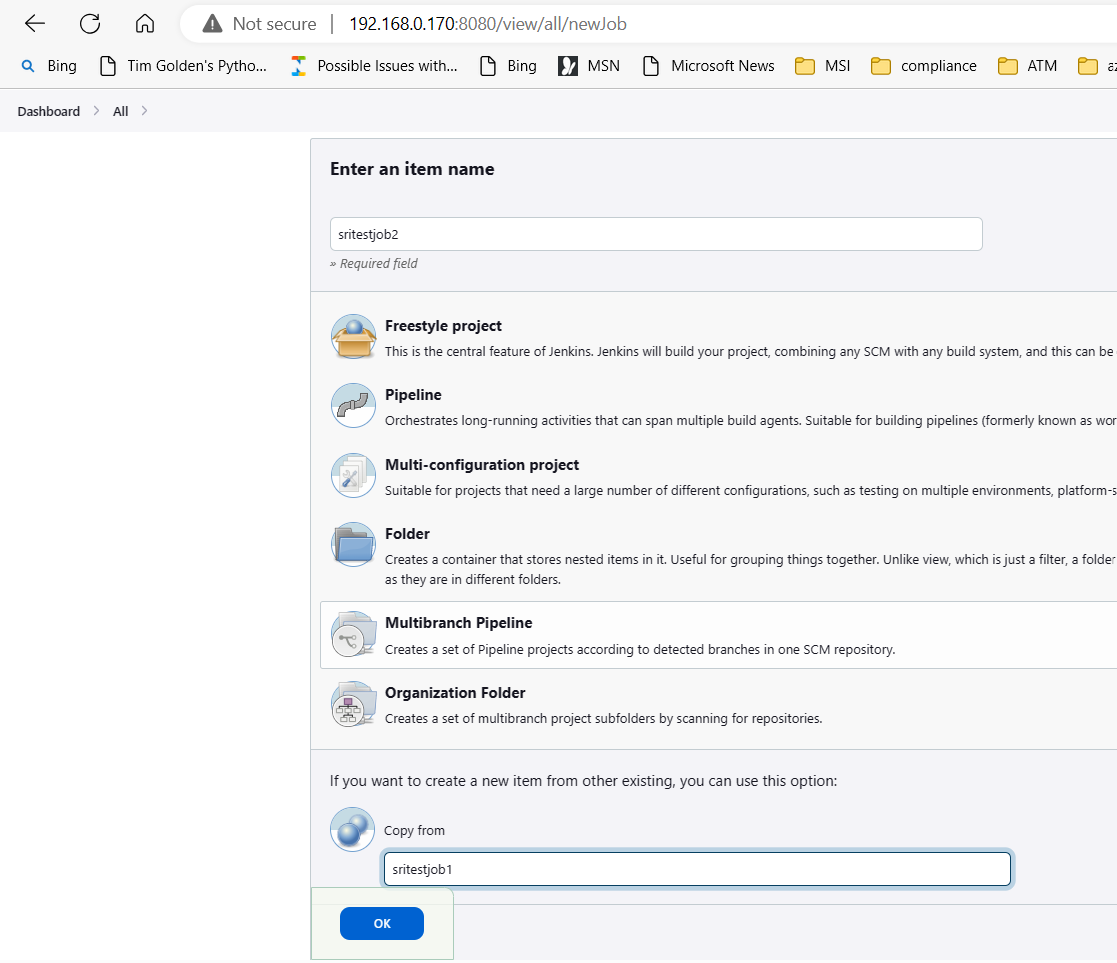


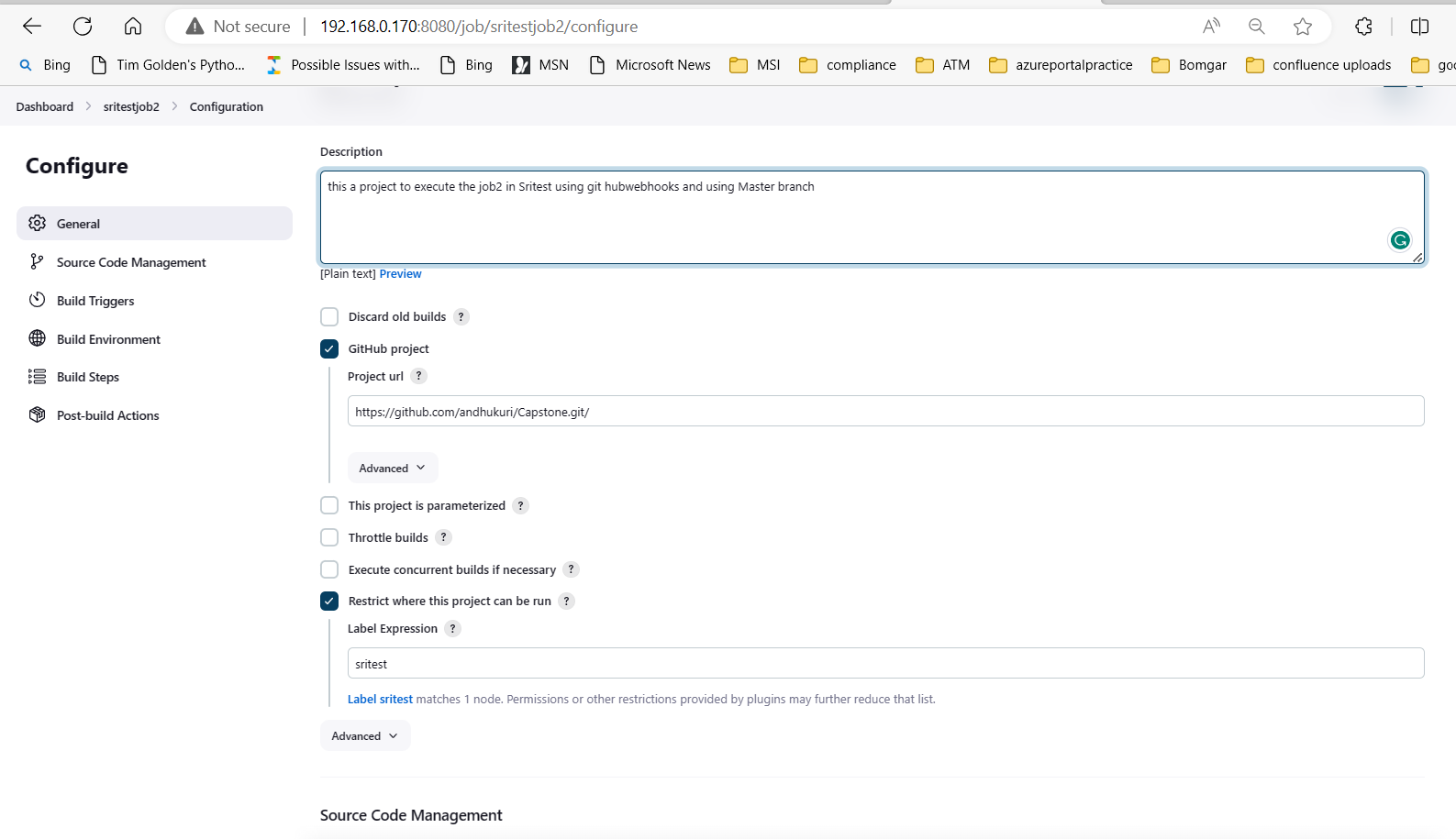
**Job2 or sritestjob2** :

On new job enter an item name sritestjob2 and select copy from sritestjob1 and make the necessary changes in build configuration and job to be executed on sritest at the specific ports.

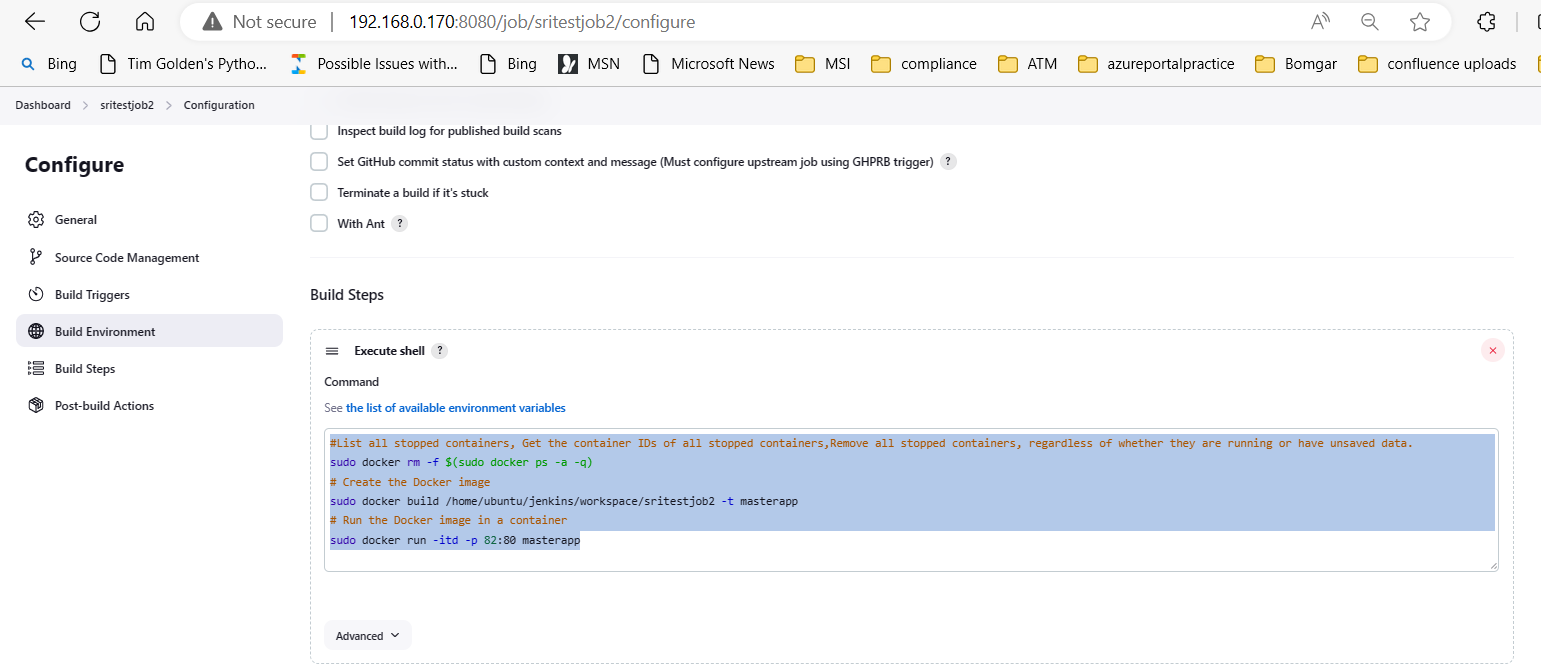
Create new Job to run from git hub project master branch name it as job2 or sritestjob2. Please note that this is still executing in sritest machines

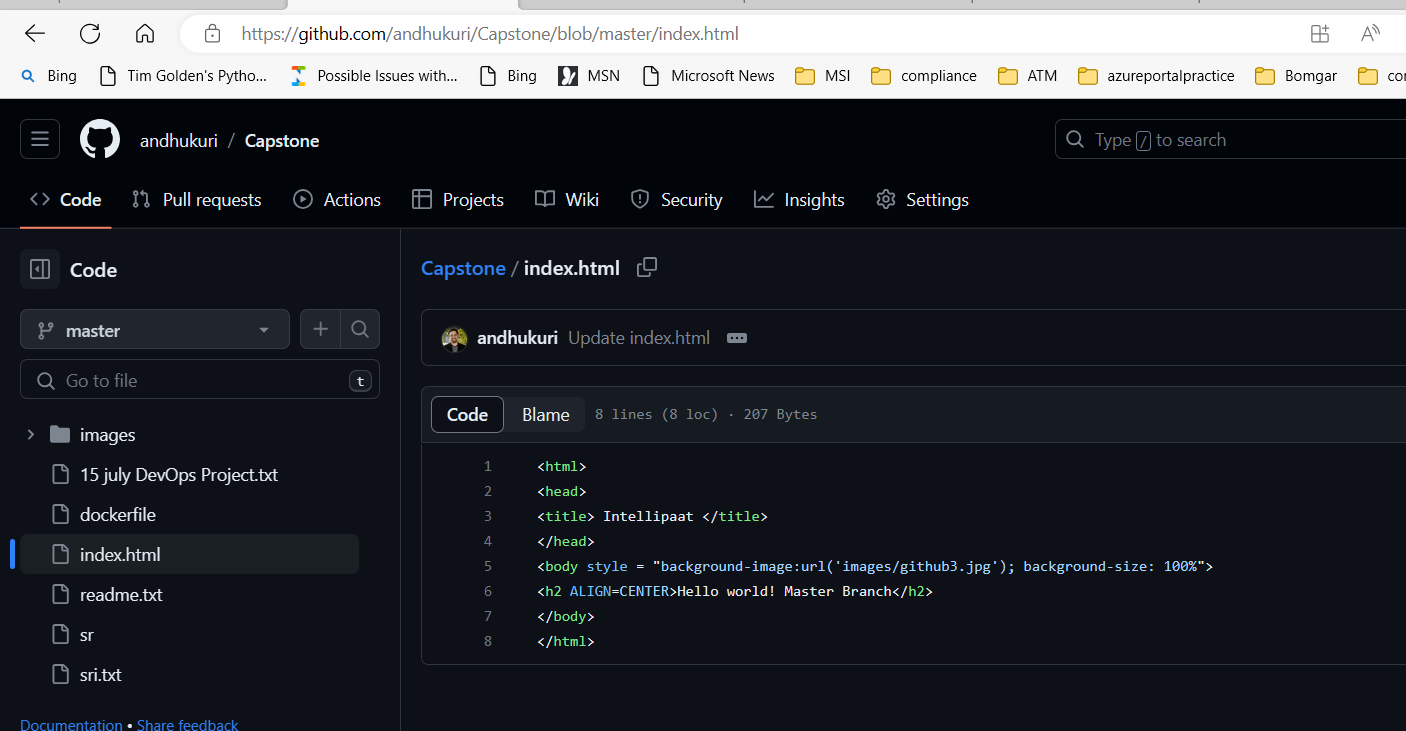
SRITESTJOB2 connecting Master Branch

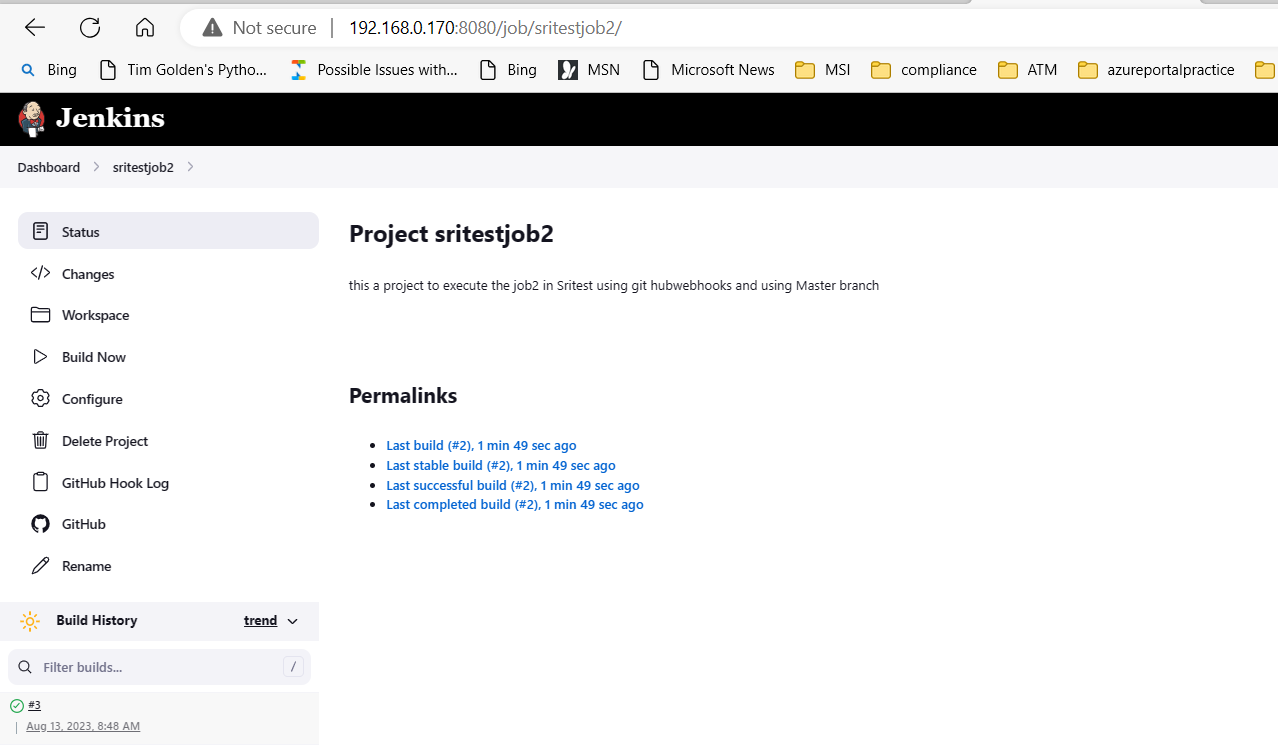


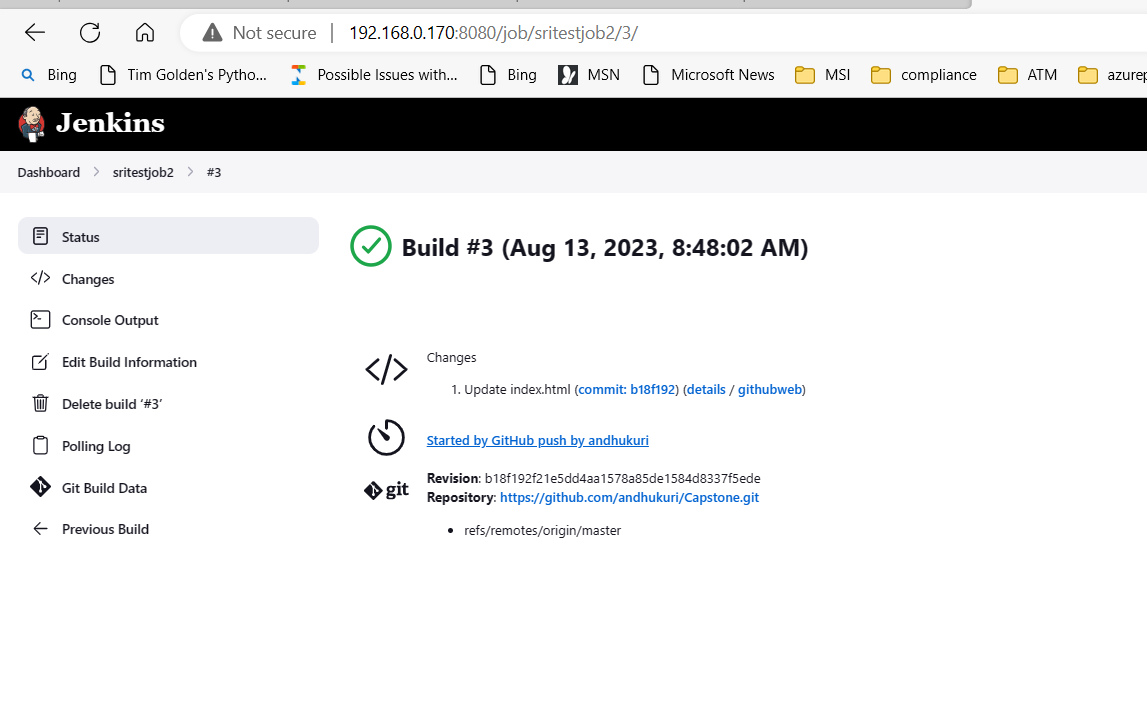


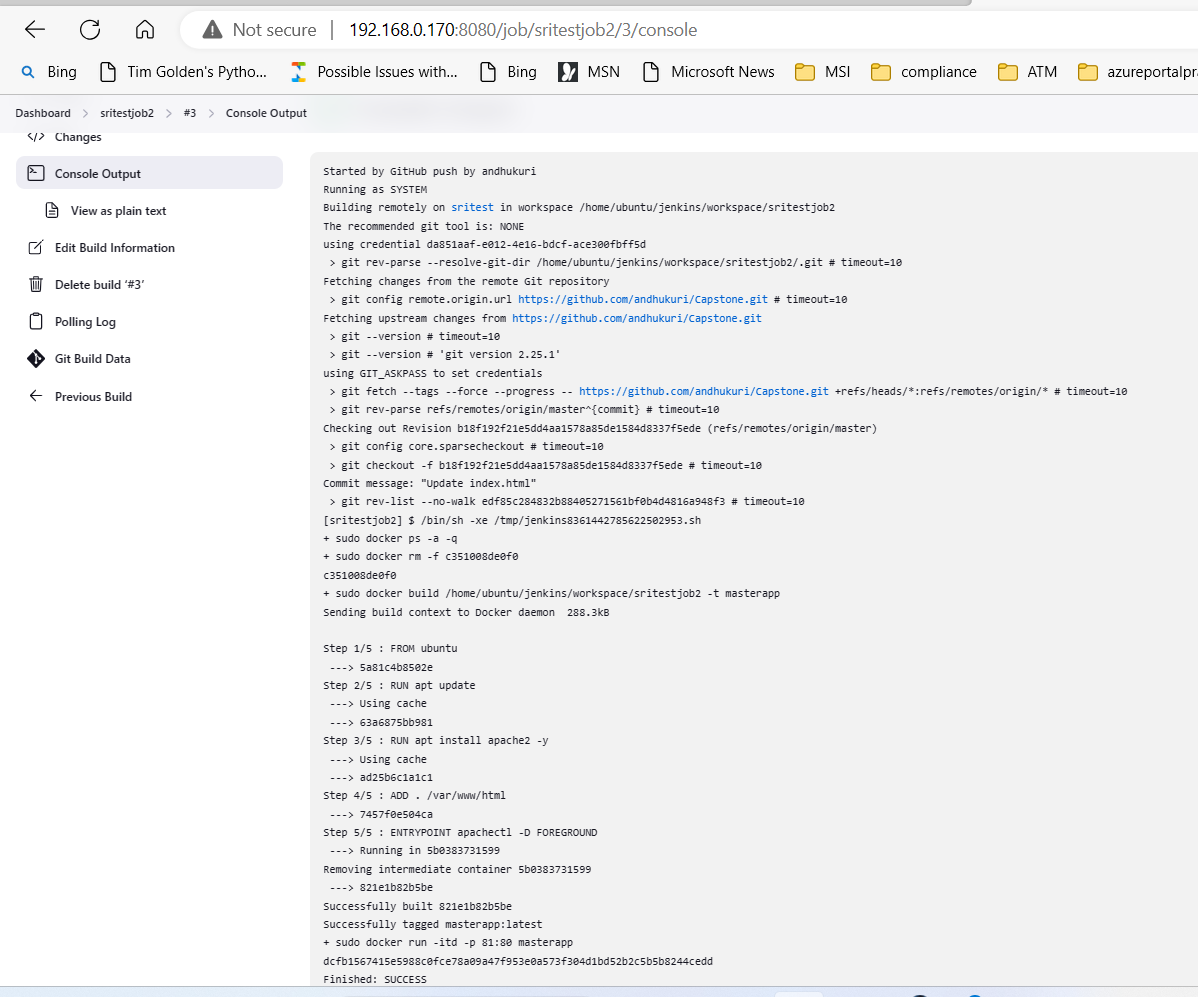


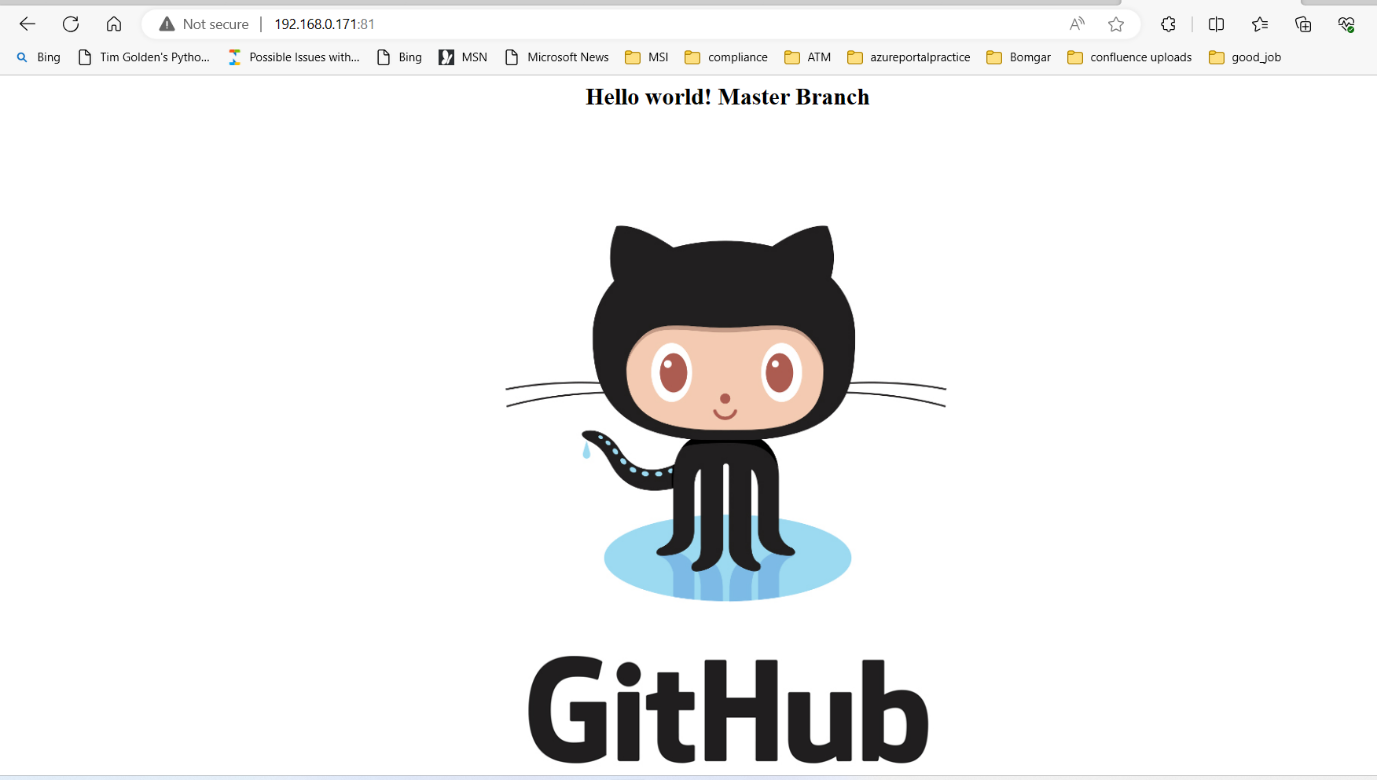


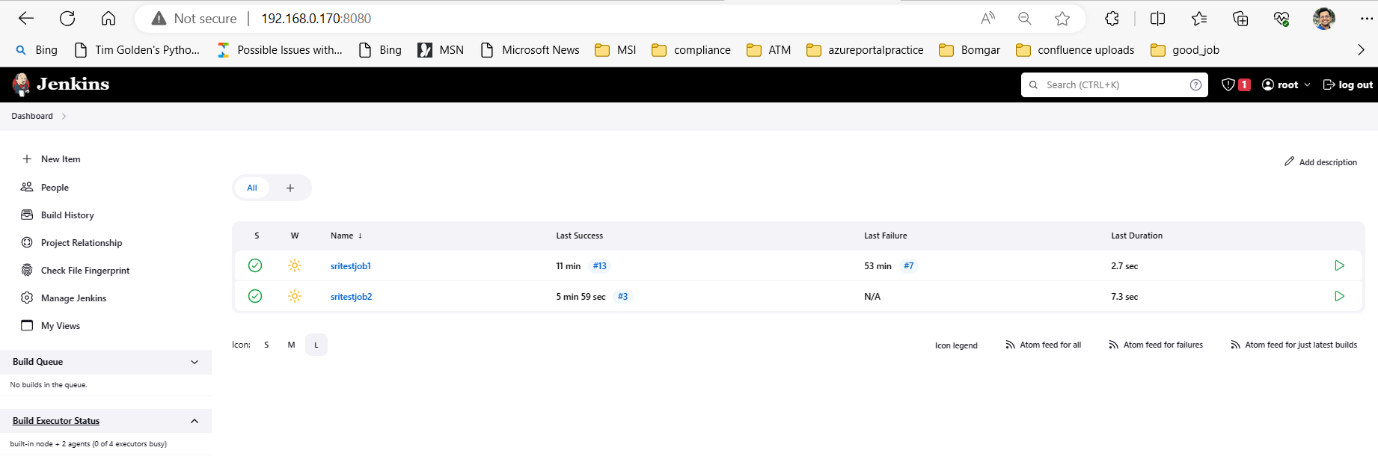








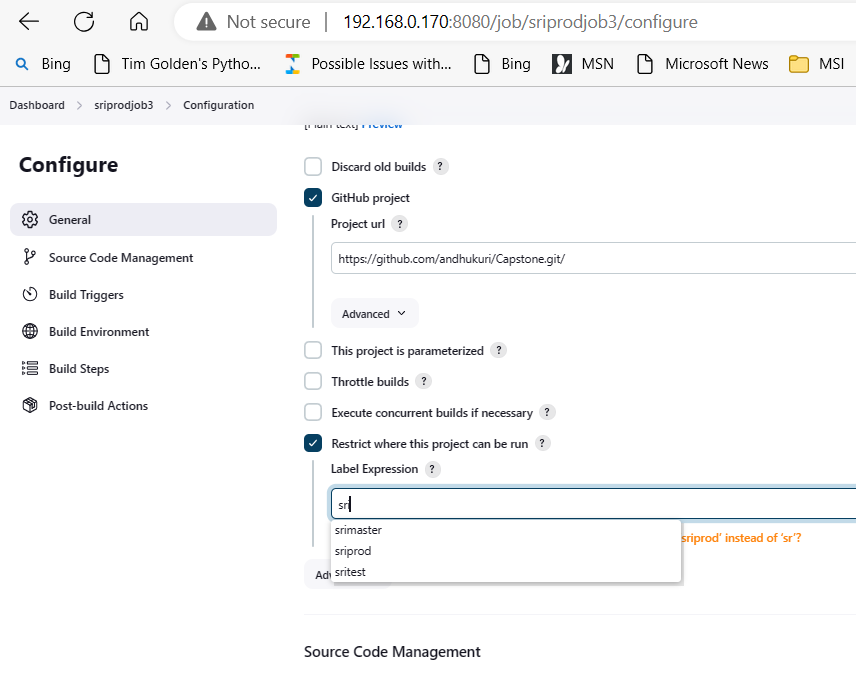


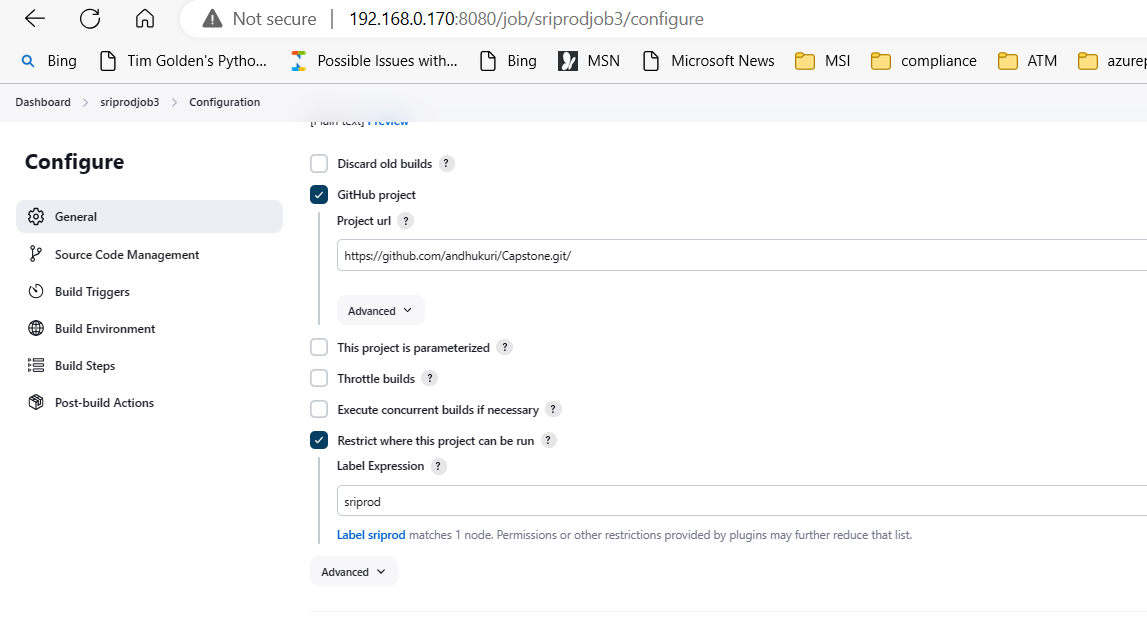


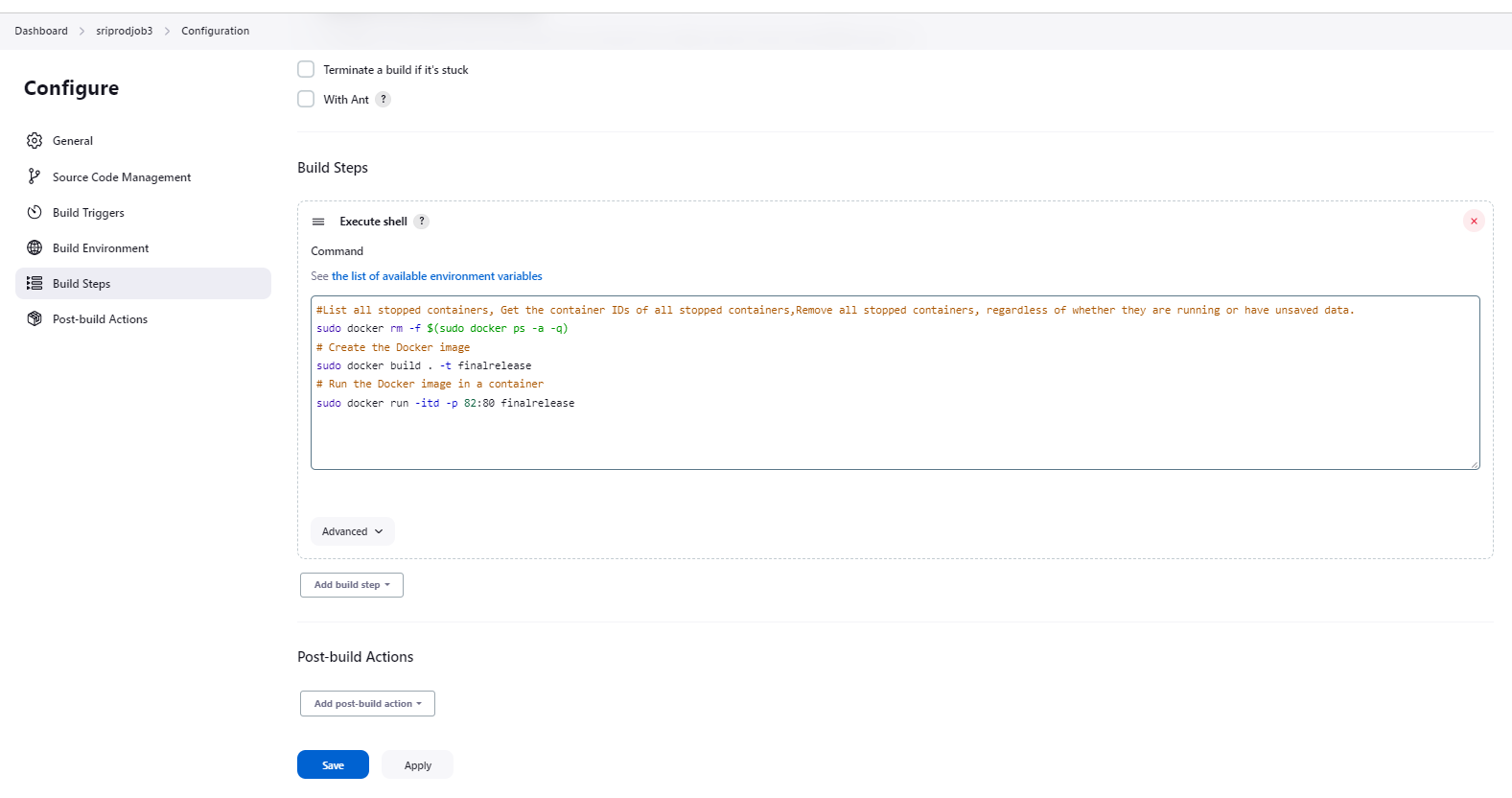
**JOB3 or SRIPRODJOB3:**

On New job enter an item name sriprodjob3 and select copy from sritestjob2 and make the necessary changes in build configuration and job to be executed on sriprod.









#List all stopped containers, Get the container IDs of all stopped containers, Remove all stopped containers, regardless of whether they are running or have unsaved data.

sudo docker rm -f $(sudo docker ps -a -q)

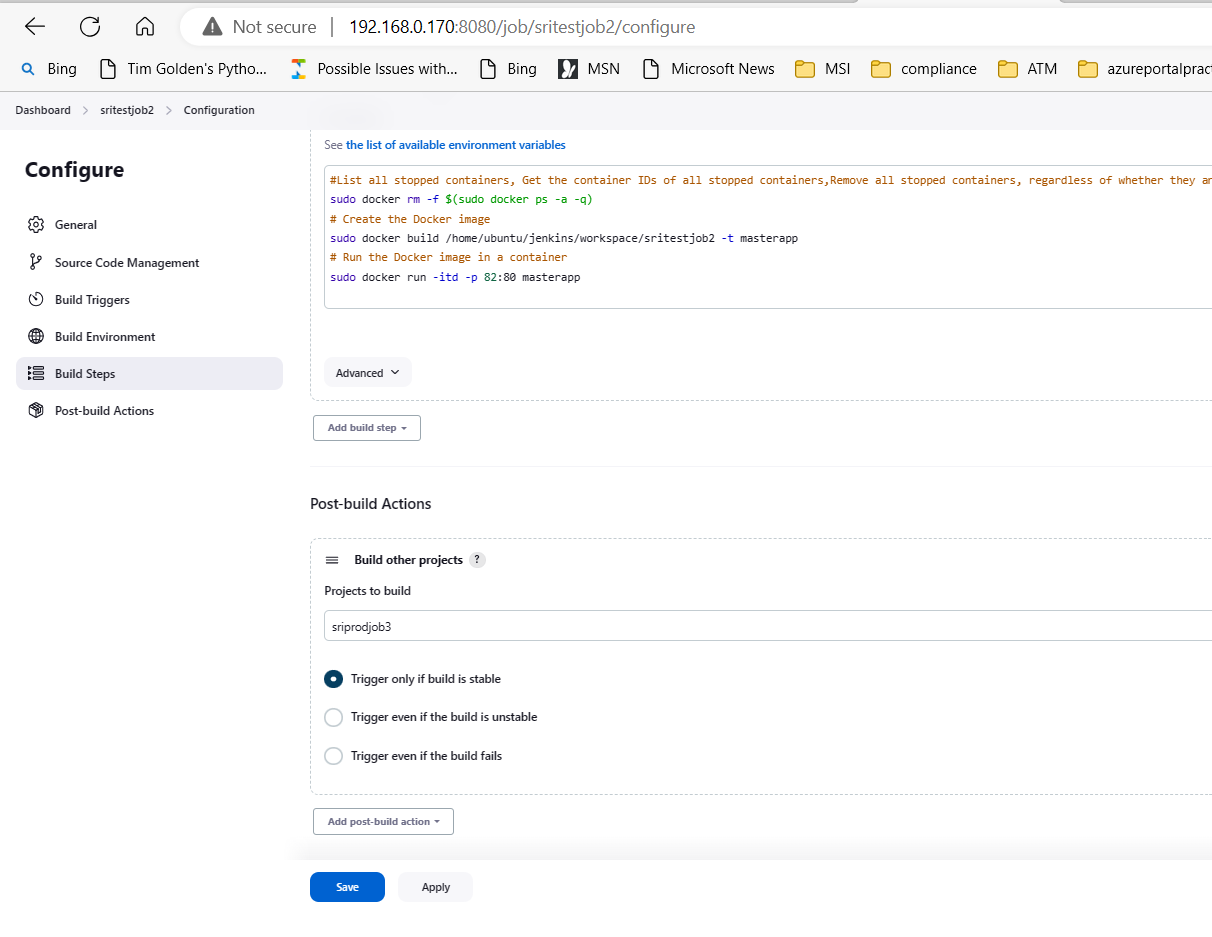
# Create the Docker image . Will auto pull the docker file

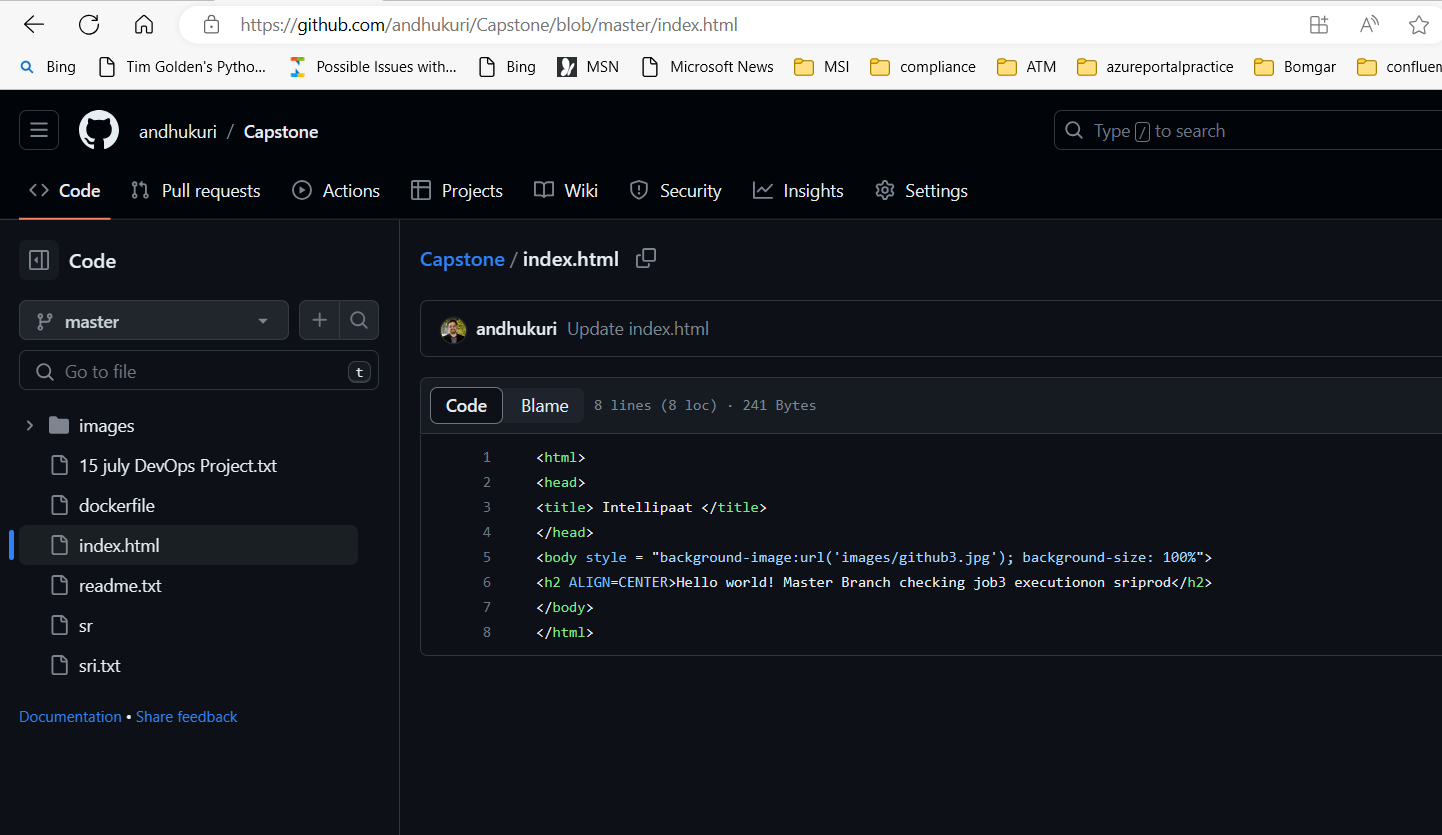
sudo docker build . -t finalrelease

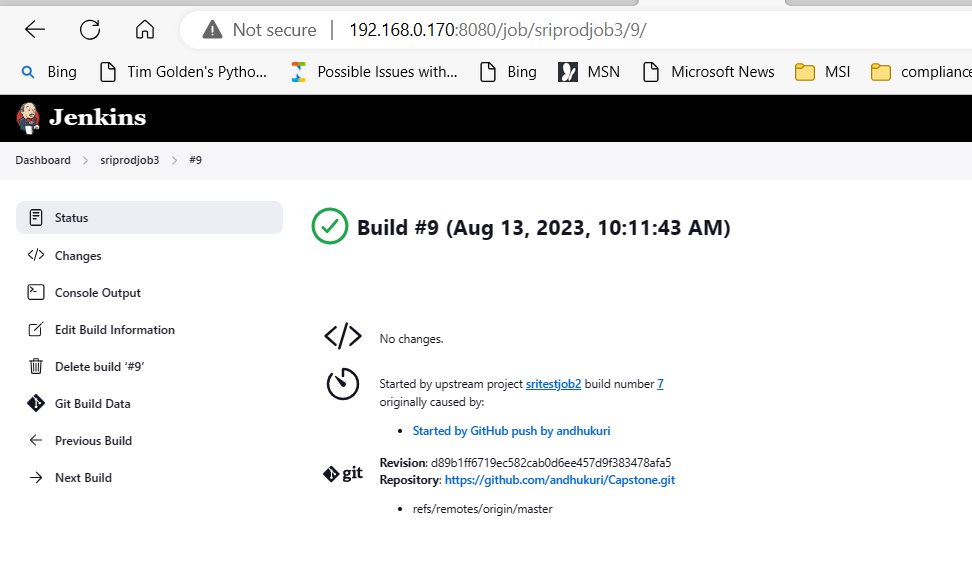
# Run the Docker image in a container

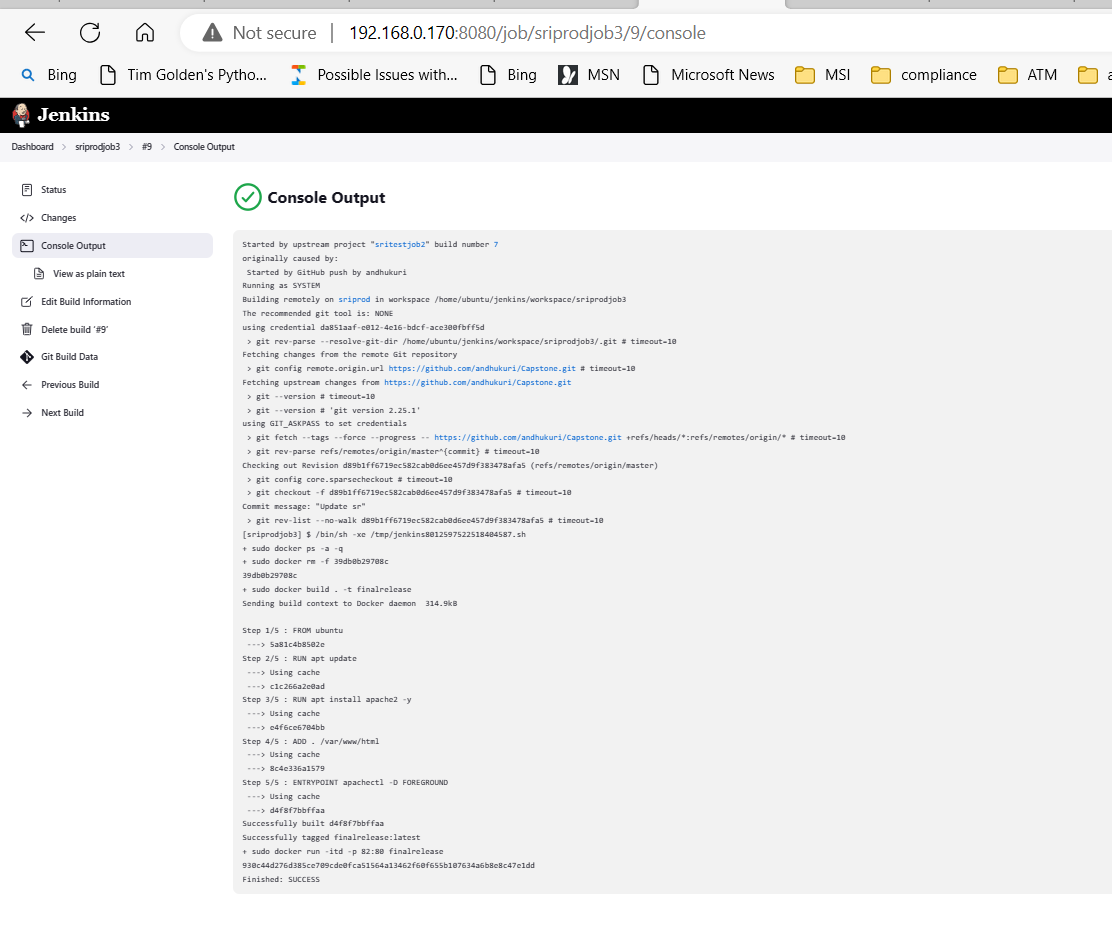
sudo docker run -itd -p 82:80 finalrelease

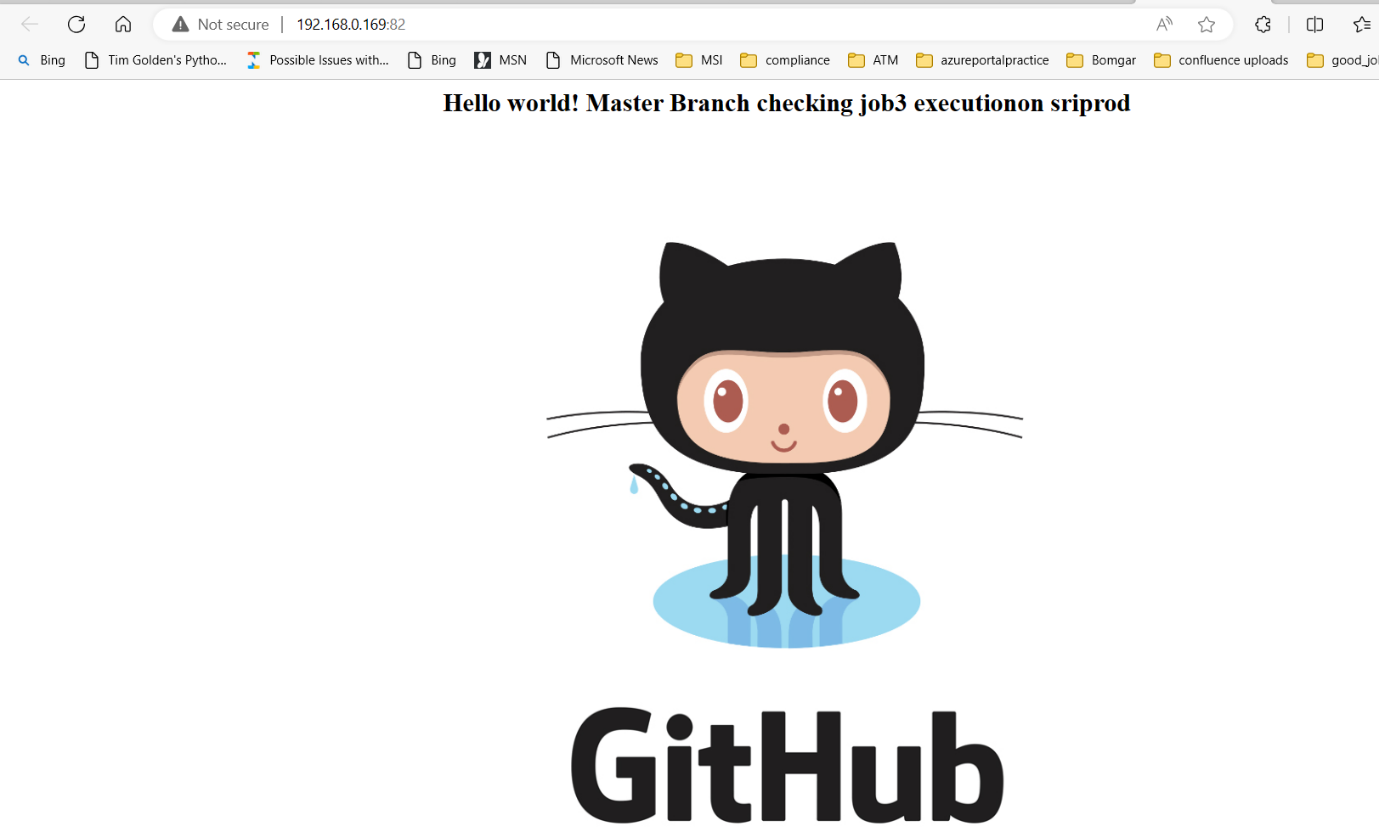
* On successful execution of sritestjob2 or job2. Sriprojjob3 or job3 will be executed configured in job2 post build steps.
* In sudo docker build, will pull the docker file from the current location and “finalrelease” can be any meaningful name.

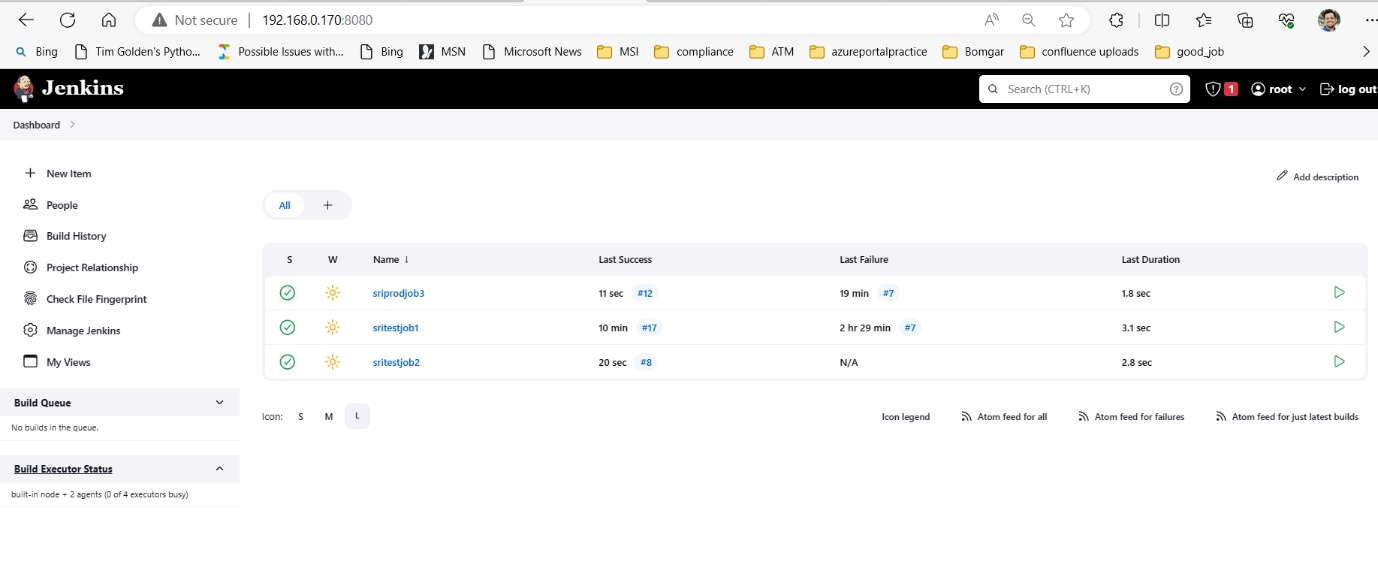












Thank you