

Setting up simple pipeline using Git, Jenkins, and Docker in local mode on an Ubuntu instance

Aim:

To set up a simple pipeline using Git, Jenkins, and Docker in local mode on an Ubuntu instance.

Procedure:

Step 1: Create an EC2 instance with Ubuntu in Amazon Machine Image.

Step 2: Update package information on the Ubuntu EC2 instance.

```
ubuntu@ip-172-31-47-227:~$ sudo apt-get update
```

```
ubuntu@ip-172-31-47-227:~$ sudo apt update
```

Step 3: Install Docker on the Ubuntu instance by downloading the installation script using 'curl' and then executing the script using the 'sh' shell to perform the installation

```
ubuntu@ip-172-31-47-227:~$ curl -fsSL https://get.docker.com -o get-docker.sh
```

```
ubuntu@ip-172-31-47-227:~$ sh get-docker.sh
```

Step 4: Check the version of docker to verify installation.

```
ubuntu@ip-172-31-47-227:~$ docker --version
```

Docker version 24.0.4, build 3713ee1

Step 5: Create a docker hub account in DockerHub website and remember the credentials.

Step 6: Login docker

```
ubuntu@ip-172-31-47-227:~$ sudo docker login -u <username> -p <password>
```

Replace <username> and <password> with DockerHub username and password.

Step 7: Install OpenJDK 17 (Java Runtime Environment).

```
ubuntu@ip-172-31-47-227:~$ sudo apt install openjdk-17-jre
```

Step 8: Check the Java version to ensure it is installed correctly.

```
ubuntu@ip-172-31-47-227:~$ java -version
```

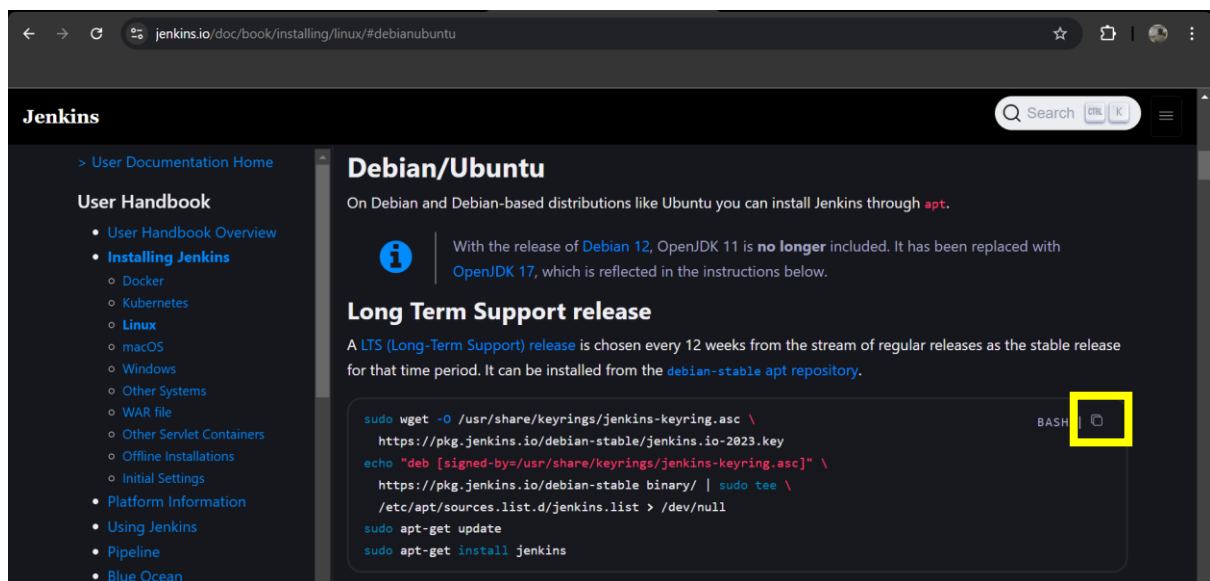
openjdk version "17.0.7" 2023-04-18

OpenJDK Runtime Environment (build 17.0.7+7-Ubuntu-0ubuntu122.04.2)

OpenJDK 64-Bit Server VM (build 17.0.7+7-Ubuntu-0ubuntu122.04.2, mixed mode, sharing)

Step 9: Import Jenkins Repository Key and install jenkins. You can get this from the Jenkins documentation.

Link: <https://www.jenkins.io/doc/book/installing/linux/#debianubuntu>



The screenshot shows the Jenkins documentation page for installing Jenkins on Debian/Ubuntu. The page title is "Debian/Ubuntu". It includes a note about OpenJDK 11 being replaced by OpenJDK 17. The "Long Term Support release" section explains that a LTS release is chosen every 12 weeks. A terminal window shows the commands to install Jenkins: `sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \`, `https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key`, `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 > /dev/null`, `sudo apt-get update`, and `sudo apt-get install jenkins`. A yellow box highlights the terminal icon in the top right corner of the terminal window.

Now to do this copy the whole code and paste in the BASH.

Step 10: Start Jenkins service and check its status.

```
ubuntu@ip-172-31-47-227:~$ sudo systemctl start jenkins.service
```

```
ubuntu@ip-172-31-47-227:~$ sudo systemctl status jenkins
```

- *jenkins.service - Jenkins Continuous Integration Server*

Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)

Active: active (running) since Fri 2023-07-21 12:50:18 UTC; 1min 57s ago

Main PID: 6680 (java)

Tasks: 38 (limit: 1111)

Memory: 292.6M

CPU: 1min 17.623s

CGroup: /system.slice/jenkins.service

*└─6680 /usr/bin/java -Djava.awt.headless=true -jar
/usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080*

Jul 21 12:49:30 ip-172-31-47-227 jenkins[6680]

Step 11: Configure firewall to allow traffic on port 8080 for Jenkins, SSH connections for remote access, port 8000 for container communication.

```
ubuntu@ip-172-31-47-227:~$ sudo ufw allow 8080
```

```
ubuntu@ip-172-31-47-227:~$ sudo ufw allow OpenSSH
```

```
ubuntu@ip-172-31-47-227:~$ sudo ufw allow 8000
```

Step 12: Enable the firewall.

```
ubuntu@ip-172-31-47-227:~$ sudo ufw enable
```

Note: Don't just hit 'Enter' as ufw will not be enabled. Just write 'y' and hit 'Enter' if it asks y/n.

Step 13: Check the firewall status.

```
ubuntu@ip-172-31-47-227:~$ sudo ufw status
```

Status: active

<i>To</i>	<i>Action</i>	<i>From</i>
--	-----	----
8080	ALLOW	Anywhere
OpenSSH	ALLOW	Anywhere
8000	ALLOW	Anywhere
8080 (v6)	ALLOW	Anywhere (v6)
OpenSSH (v6)	ALLOW	Anywhere (v6)
8000 (v6)	ALLOW	Anywhere (v6)

Step 14: Retrieve the initialAdminPassword for Jenkins setup.

```
ubuntu@ip-172-31-47-227:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

c84c43b9adf14d60873f4185427328ba

Step 15: Add the Jenkins user to the docker group for Docker access.

```
ubuntu@ip-172-31-47-227:~$ sudo usermod -aG docker jenkins
```

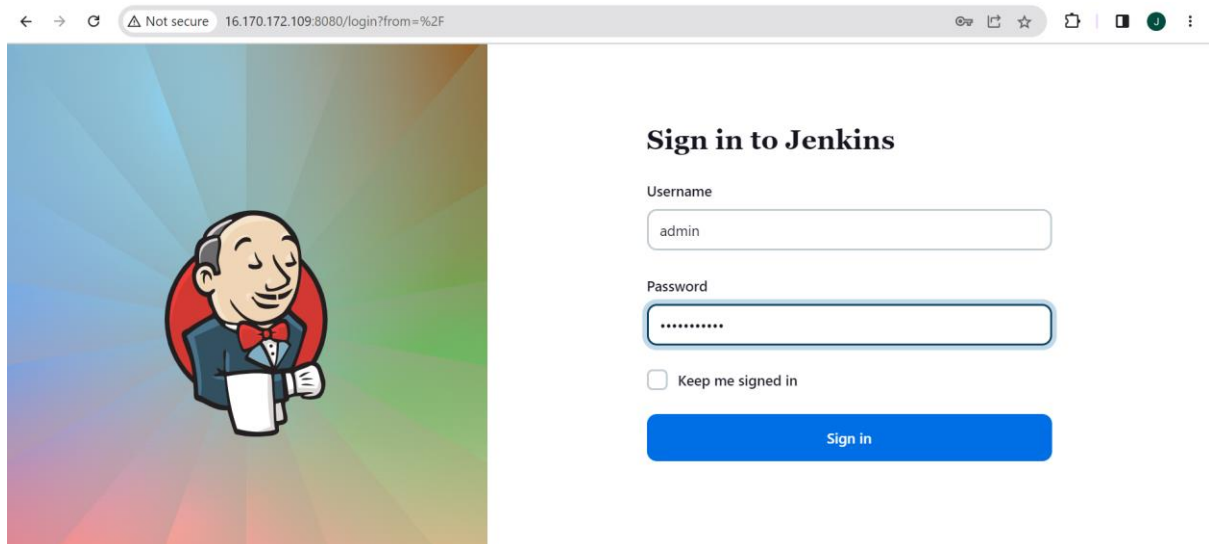
Step 16: Restart Jenkins service to apply the changes.

```
ubuntu@ip-172-31-47-227:~$ sudo systemctl restart jenkins
```

Step 17: Allow Custom TCP port 8080, 8000 in Inbound rules.

Step 18: Now open <Public_IP_Address_of_EC2_Instance>:8080 in Browser to open Jenkins.

Step 22: Login with initialAdminPassword that we retrieved in Step 14 – Install suggested plugins – Setup new username and password. After that you will get this page or you will be logged in



← → ↻ ⚠ Not secure 16.170.172.109:8080/login?from=%2F 🔍 📄 ☆ 🗑 📱 🔒 🔍

Sign in to Jenkins

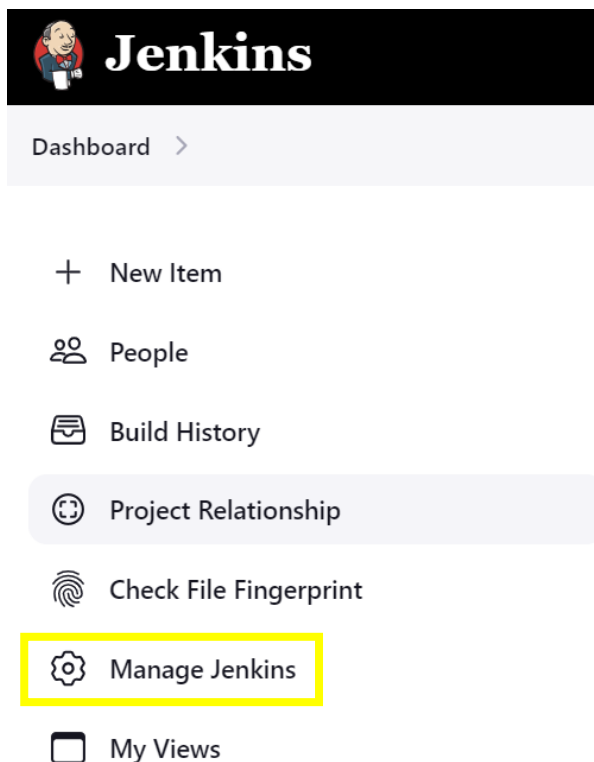
Username
admin

Password
.....

☐ Keep me signed in

Sign in

Step 23: Setup Credentials in Jenkins.





<next>


Manage Jenkins


Search settings


System Configuration

**System**
Configure global settings and paths.


**Tools**
Configure tools, their locations and automatic installers.


**Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.


**Nodes**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

**Clouds**
Add, remove, and configure cloud instances to provision agents on-demand.


Security

**Security**
Secure Jenkins; define who is

**Credentials**
Configure credentials

**Credential Providers**
Configure the credential providers

<next>

**Jenkins**

Search (CTRL+K)


log out

Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store	Domain	ID	Name
---	---	-------	--------	----	------

Stores scoped to Jenkins

P	Store	Domains
	System	(global)

Icon: S M L

<next>

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

<next>

Now add your **Docker hub User name** and **password** and save it with Id (I have used **test1**). Also make necessary changes in the Jenkinsfile like setting **credentials id** and **username of docker**

New credentials

Kind
Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
jenish007

☐ Treat username as secret ?

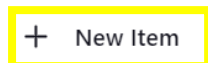
Password ?


ID ?
test1


Description ?
dockerhub credentials




Step 24: Create Pipeline in Jenkins.





 People

 Build History

 Project Relationship

 Check File Fingerprint

 Manage Jenkins

 My Views


<next>


Enter an item name of your wish.


Enter an item name


app

» Required field

**Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.


**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a

<next>

Add some description:

General

Enabled 

Description

automated builds

Plain text [Preview](#)

<next>

Build Triggers with Poll SCM.

This poll SCM with Schedule ‘* * * * *’ will check the git repository every minute and if a commit is encounter it automatically build up...

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☒ Poll SCM ?

Schedule ?

* * * * *

<next>

Display name – The same name given in item name

Advanced Project Options

Advanced ^ Edited

Display Name ?

app

<next>

Setting up pipeline definition from my GitHub Repository.

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/JenishRevaldo/jenkins-demo.git

Credentials ?

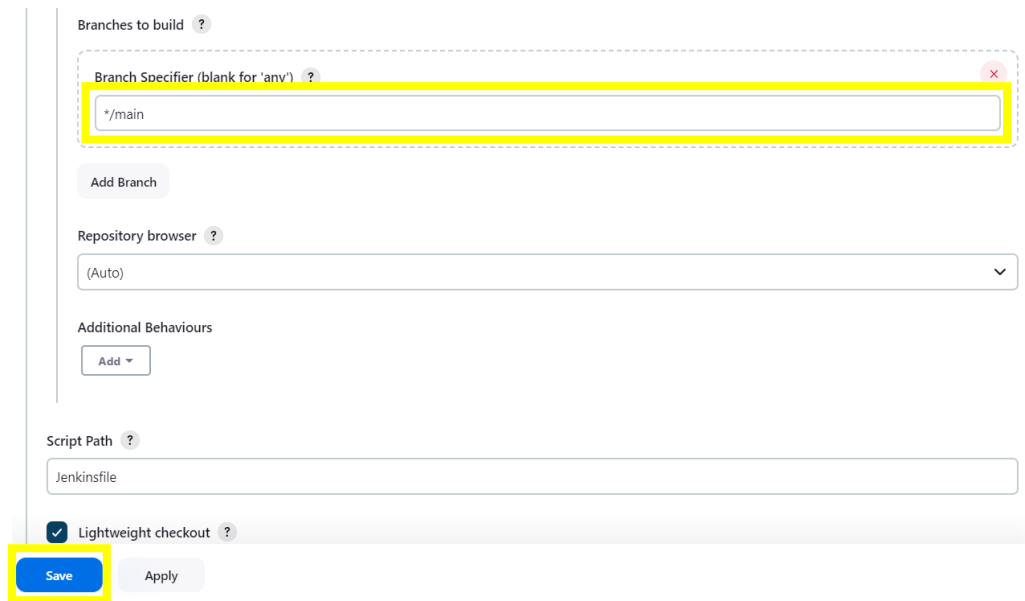
- none -

Add ▾

Advanced ▾

<next>

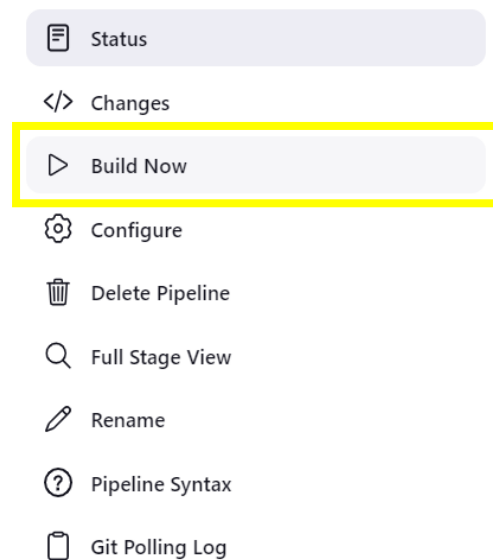
Change the branch specifier according to your repository and save the pipeline.



A screenshot of the Jenkins Pipeline Configuration page. The 'Branches to build' section is highlighted with a yellow dashed border. Inside this section, the 'Branch Specifier (blank for \'any\')' field is highlighted with a solid yellow border and contains the text '*/main'. Below this is an 'Add Branch' button. The 'Repository browser' dropdown menu is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. The 'Script Path' field contains 'Jenkinsfile'. At the bottom, the 'Lightweight checkout' checkbox is checked. The 'Save' button is highlighted with a yellow border.

<next>

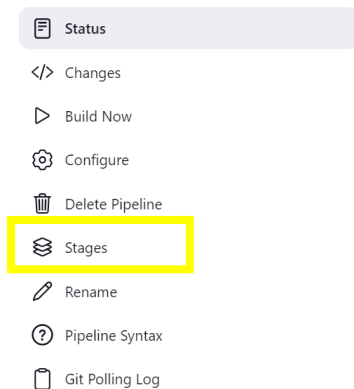
Now build the pipeline.



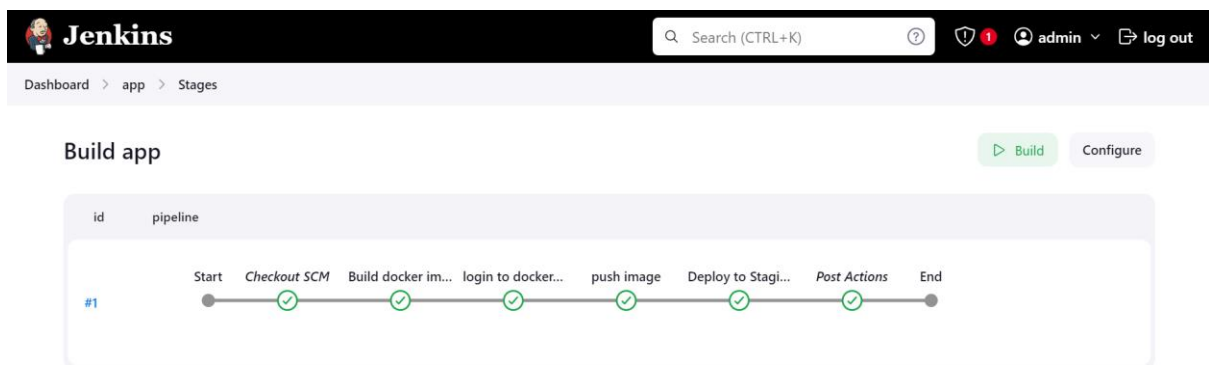
A screenshot of the Jenkins Pipeline Actions menu. The 'Build Now' button is highlighted with a yellow border. The menu includes the following options: 'Status', 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Full Stage View', 'Rename', 'Pipeline Syntax', and 'Git Polling Log'.

<next>

Stages of pipeline will get executed. Click on the ‘Stages’ Menu to see the stages.

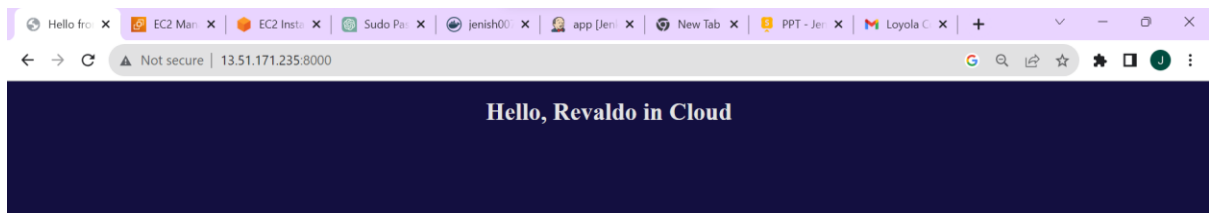


You can see the stages are successful now:



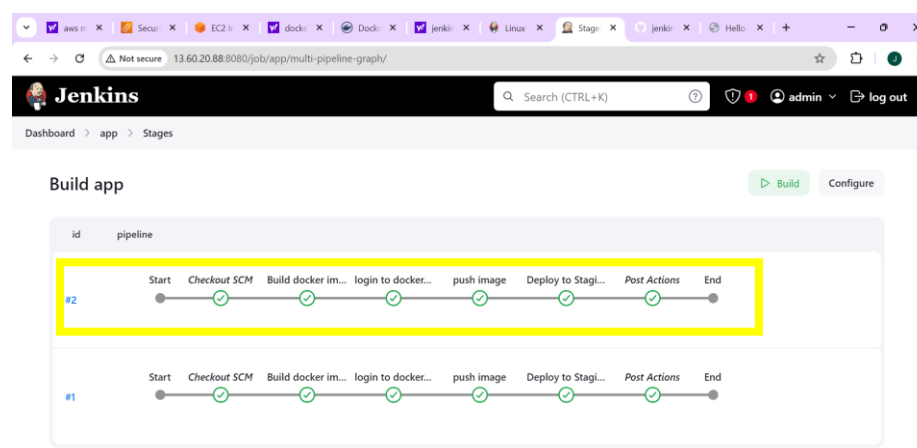
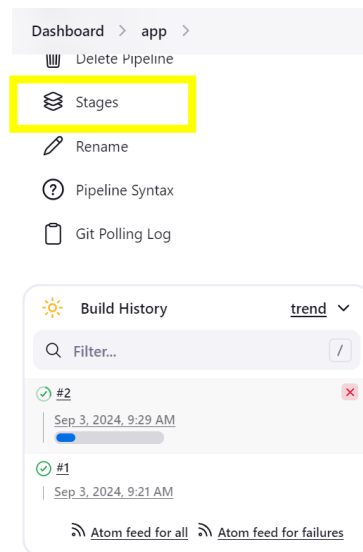
Step 25: Now open <Public_IP_Address_of_EC2_Instance>:8000 in Browser

Output:

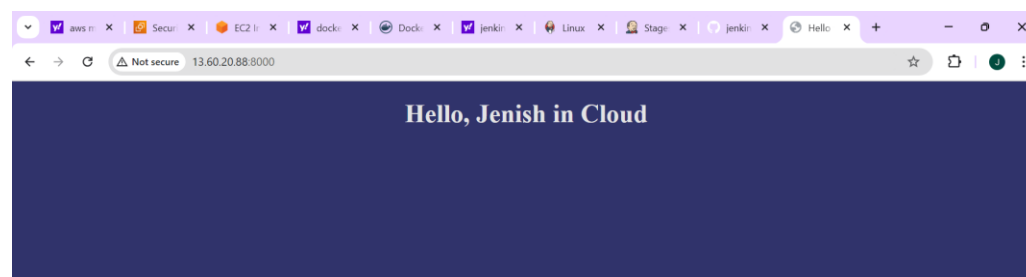


Now you can make changes in the **hello.html** file located in the templates folder in the repository and commit it, within one minute it gets reflect in the end page.

After committing changes in hello.html file you can see the triggers automatic execution in the 'Stages' menu.



Now you can see the changes in the end page:



The local pipeline setup for Git, Jenkins, and Docker has been successfully established on the Ubuntu instance, enabling efficient software development, testing, and deployment processes.