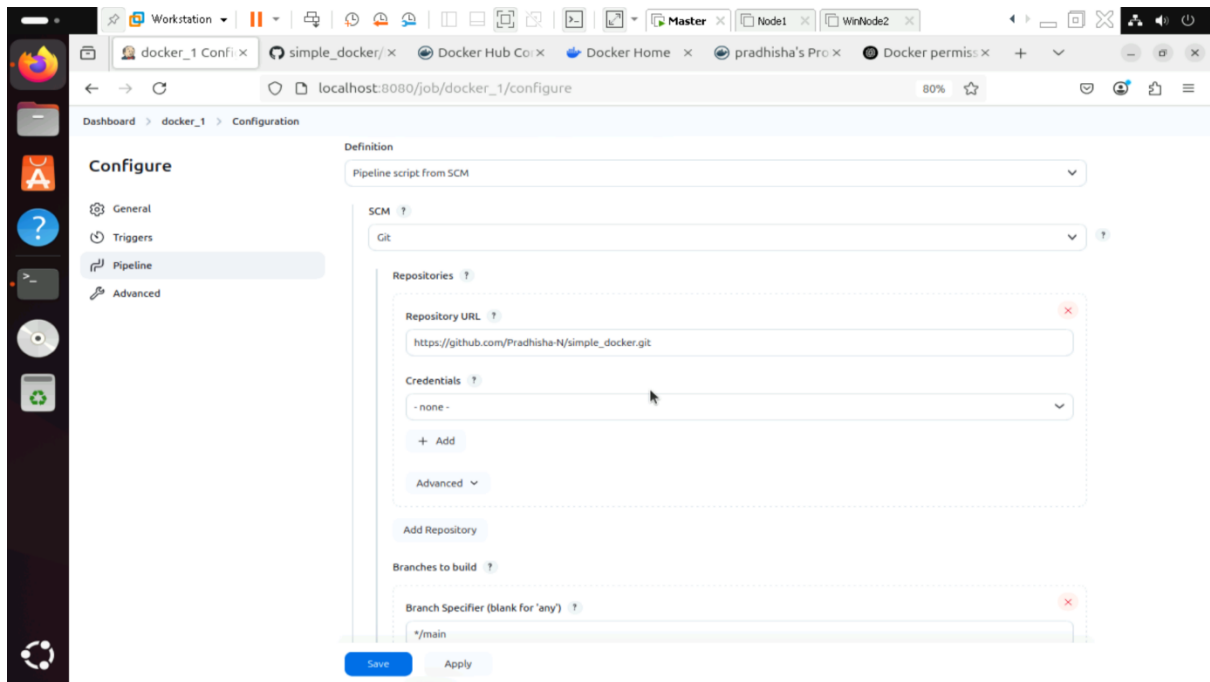
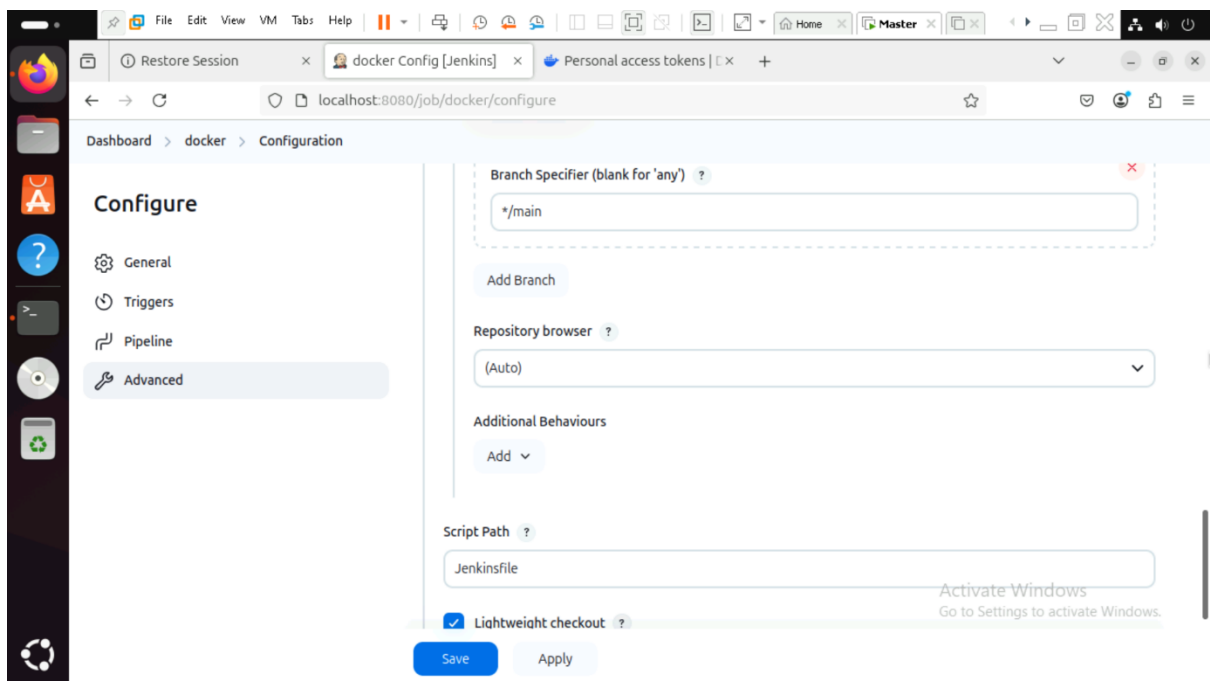


1. Deploy a Python Flask App using Docker

Step 1: Jenkins Pipeline



Step 2 : Save the Pipeline



Step 3: Build and run the docker container

The screenshot shows the Jenkins web interface in a browser window. The URL is `localhost:8080/job/docker/`. The page displays the 'Stage View' of a pipeline named 'docker'. The pipeline consists of four stages: 'Declarative: Checkout SCM', 'Clone Repository', 'Build Docker Image', and 'Push Docker Image'. The 'Push Docker Image' stage is currently failing, indicated by a red box and the text 'failed'. The average stage times are shown as: Declarative: Checkout SCM (1s), Clone Repository (1s), Build Docker Image (12s), and Push Docker Image (8s). The full run time is approximately 39s. The left sidebar shows the Jenkins dashboard with various options like 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Full Stage View', 'Favorite', 'Open Blue Ocean', 'Stages', 'Rename', and 'Pipeline Syntax'.

Step 4: Push the docker files to Github

The screenshot shows the GitHub repository page for 'Pradhisha-N/simple_docker'. The repository is public and contains the following files: 'Dockerfile', 'Jenkinsfile', 'README.md', 'app.py', and 'requirements.txt'. The repository has 5 commits and 0 stars. The 'About' section on the right indicates that there is no description, website, or topics provided. The 'Releases' section shows that no releases have been published. The 'Packages' section shows that no packages have been published.

Step 5: Output

```
Master - VMware Workstation
File Edit View VM Tabs Help
Home Master Node1 WinNode2
Mar 5 12:40
master@mastervm: ~
Main PID: 1913 (dockerd)
Tasks: 9
Memory: 111.0M (peak: 111.3M)
CPU: 4.575s
CGroup: /system.slice/docker.service
└─1913 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont
Mar 05 11:11:24 mastervm dockerd[1913]: time="2025-03-05T11:11:24.734117789+05:
Mar 05 11:11:24 mastervm dockerd[1913]: time="2025-03-05T11:11:24.774226029+05:
Mar 05 11:11:25 mastervm dockerd[1913]: time="2025-03-05T11:11:25.638271939+05:
Mar 05 11:11:25 mastervm dockerd[1913]: time="2025-03-05T11:11:25.840168351+05:
Mar 05 11:11:27 mastervm dockerd[1913]: time="2025-03-05T11:11:27.724311288+05:
Mar 05 11:11:27 mastervm dockerd[1913]: time="2025-03-05T11:11:27.941697029+05:
Mar 05 11:11:28 mastervm dockerd[1913]: time="2025-03-05T11:11:28.197976077+05:
Mar 05 11:11:28 mastervm dockerd[1913]: time="2025-03-05T11:11:28.200891166+05:
Mar 05 11:11:28 mastervm systemd[1]: Started docker.service - Docker Applicatio
Mar 05 11:11:28 mastervm dockerd[1913]: time="2025-03-05T11:11:28.508723895+05:
lines 1-22/22 (END)
master@mastervm:~$ docker pull pradhisha/my-flask-app
Using default tag: latest
latest: Pulling from pradhisha/my-flask-app
Digest: sha256:2b4e71ce8cd79b6c8a1f87f97a8f79997910c0e908271b0f7d767134cba08cda
Status: Image is up to date for pradhisha/my-flask-app:latest
docker.io/pradhisha/my-flask-app:latest
master@mastervm:~$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
pradhisha/my-flask-app latest         114e8fa55c8e    4 minutes ago   136MB
pradhisha/my-flask-app <none>         bfd8add60f4f    18 hours ago    136MB
python              3.9-slim       1a47c1aaa88f    3 months ago    126MB
master@mastervm:~$ docker run pradhisha/my-flask-app
Hello
master@mastervm:~$
```

To return to your computer, move the mouse pointer outside or press Ctrl+Alt.

Type here to search

ENG 12:40
US 05-03-2025

2. Installing and enabling docker inside ubuntu terminal

Step 1: `sudo apt upgrade -y`

```
master@master-vm:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
  libpostproc55 libavcodec58 libavutil56 libswscale5 libswresample3
  libavformat58 libavfilter7
Learn more about Ubuntu Pro at https://ubuntu.com/pro
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Step 2 : `sudo apt install -y ca-certificates curl gnupg lsb-release`

```
master@master-vm:~$ sudo apt install -y ca-certificates curl gnupg lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
ca-certificates is already the newest version (20240203-22.04.1).
ca-certificates set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 194 kB of archives.
After this operation, 455 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.20 [194 kB]
Fetched 194 kB in 1s (225 kB/s)
Selecting previously unselected package curl.
(Reading database ... 205241 files and directories currently installed.)
Preparing to unpack .../curl_7.81.0-1ubuntu1.20_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.20) ...
Setting up curl (7.81.0-1ubuntu1.20) ...
Processing triggers for man-db (2.10.2-1) ...
```

Step 3 : `sudo mkdir -p /etc/apt/keyrings`

```
master@master-vm:~$ sudo mkdir -p /etc/apt/keyrings
```

Step 4 : `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/keyrings/docker.asc > /dev/null`

```
master@master-vm:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/keyrings/docker.asc > /dev/null
[sudo] password for master:
```

Step 5 : `sudo chmod a+r /etc/apt/keyrings/docker.asc`

```
master@master-vm:~$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Step 6 : `echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null`

Step 7 : `sudo apt update`

```

master@master-vm: $ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
master@master-vm: $ sudo apt update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [45.7 kB]
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [762 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,114 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,354 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [594 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [392 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [103 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [3,008 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [330 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43.1 kB]
Get:16 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [528 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2,904 kB]
Get:18 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 DEP-11 Metadata [208 B]
Get:19 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [759 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [510 kB]
Get:21 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 DEP-11 Metadata [208 B]
Get:22 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,192 kB]
Get:23 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [652 kB]
Get:24 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [966 kB]
Get:25 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [293 kB]
Get:26 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [207 kB]
Get:27 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [126 kB]

```

Step 8 : `sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin`

```

master@master-vm: $ sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker' is not installed, so not removed
The following additional packages will be installed:
  docker-ce-rootless-extras
  docker-compose-plugin git git-man
  liberror-perl libslirp0 pigz
  slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite
  git-daemon-run | git-daemon-sysvinit
  git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin
  docker-ce docker-ce-cli
  docker-ce-rootless-extras
  docker-compose-plugin git git-man
  liberror-perl libslirp0 pigz
  slirp4netns
0 upgraded, 12 newly installed, 0 to remove and 19 not upgraded.

```

Step 9 : `sudo systemctl start docker`

Step 10 : `sudo systemctl enable docker`

Step 11 : `sudo docker --version`

Step 12 : `sudo usermod -aG docker $USER`

Step 13 : `newgrp docker`

```

master@master-vm: $ sudo systemctl start docker
master@master-vm: $ sudo systemctl start docker
master@master-vm: $ sudo docker --version
Docker version 28.0.1, build 068a01e
master@master-vm: $ sudo usermod -aG docker $USER
master@master-vm: $ newgrp docker

```

Connecting ubuntu terminal with GUI based portainer

Step 1 : `sudo docker pull portainer/portainer-ce`

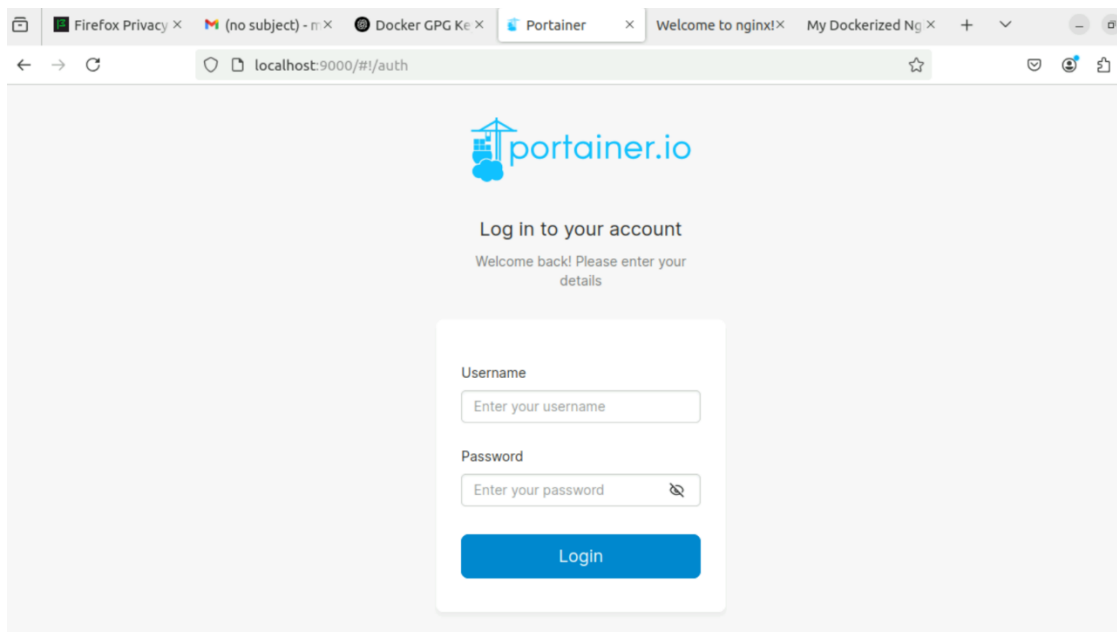
Step 2 : `sudo docker run -d -p 9000:9000 --name=portainer --restart=always \-v /var/run/docker.sock:/var/run/docker.sock \-v portainer_data:/data \portainer/portainer-ce`


```

master@master-vm:~$ sudo docker pull portainer/portainer-ce
Using default tag: latest
latest: Pulling from portainer/portainer-ce
436768c74267: Pull complete
d61825c69234: Pull complete
04de093ad5ed: Pull complete
a528983d077c: Pull complete
26eb502a78ed: Pull complete
b2724536dfda: Pull complete
5b45cfb2ea0c: Pull complete
20b115ea6339: Pull complete
8e73efb50b28: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:99c3047d44991af08f2a34df16e69ae2654bee43444b2e9857aa6b5864c4f602
Status: Downloaded newer image for portainer/portainer-ce:latest
docker.io/portainer/portainer-ce:latest
master@master-vm:~$ sudo docker run -d -p 9000:9000 --name=portainer --restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data \
portainer/portainer-ce
99ca4a984d3119de16ca1ed2a0557a776bd9b103f3f76a1a562033e2391a9fa8

```

Step 3 : go to <http://localhost:9000> and enter password



Deploy an Nginx Web Server with Docker

Goal: Run an Nginx server using Docker.

Prerequisites: Install Docker

Step 1: Pull the Nginx Image

Run the following command to pull the official Nginx image:

docker pull nginx

```

master@master-vm:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
7cf63256a31a: Pull complete
bf9acace214a: Pull complete
513c3649bb14: Pull complete
d014f92d532d: Pull complete
9dd21ad5a4a6: Pull complete
943ea0f0c2e4: Pull complete
103f50cb3e9f: Pull complete
Digest: sha256:9d6b58feebd2dbd3c56ab5853333d627cc6e281011cfd6050fa4bcf2072c9496
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

```

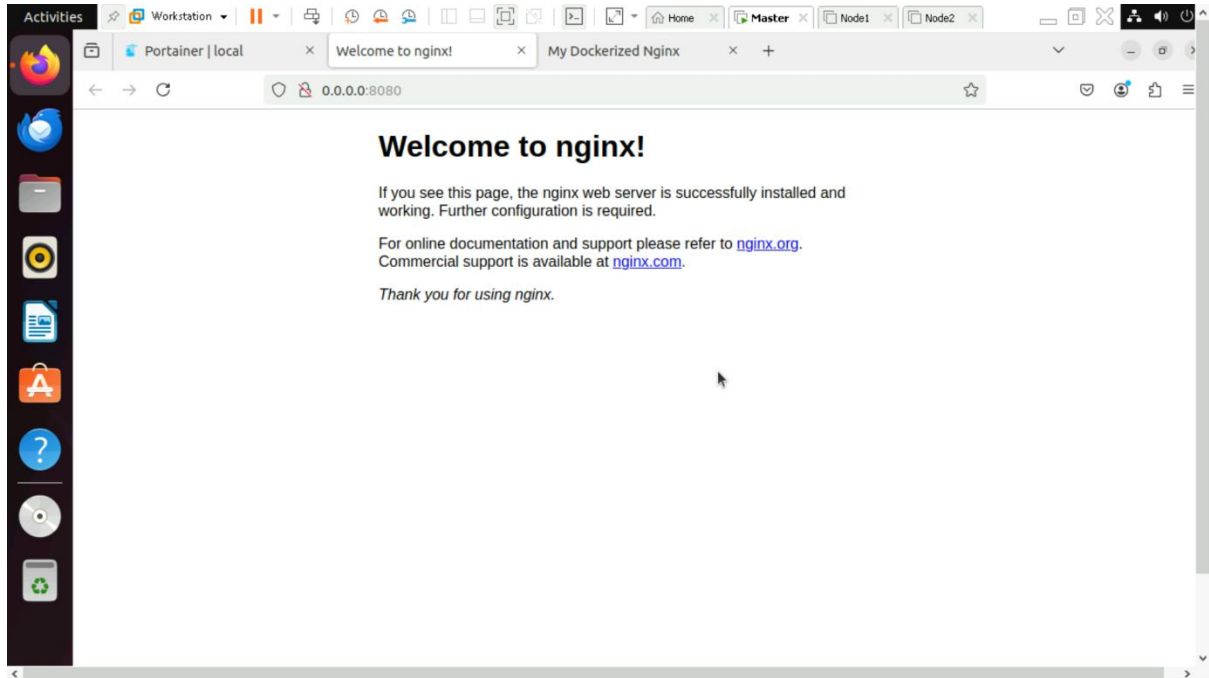
Step 2: Run an Nginx Container

Start a container and map port 80 to access it from your browser:

```
docker run -d -p 8080:80 --name my-nginx nginx
```

```
master@master-vm:~$ docker run -d -p 8080:80 --name my-nginx nginx
75a598e8e8c67a578886d1da9d87af36ac3886784c55b3e4af705e015ed42b90
```

Now, open <http://localhost:8080> in your browser, and see the Nginx welcome page.



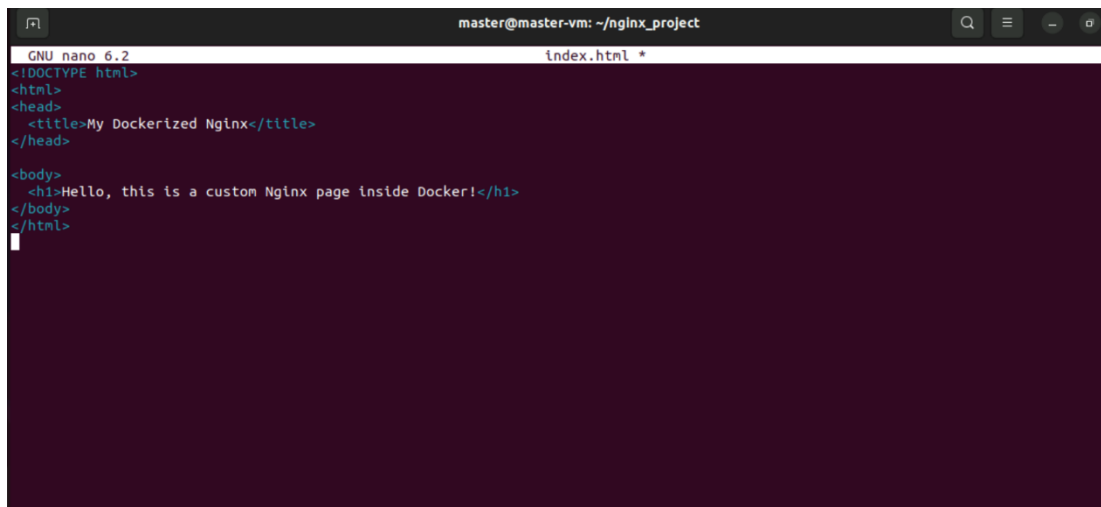
Step 3: Customize Nginx with Your Own HTML Page

Create a directory for your Nginx files:

```
mkdir nginx_project && cd nginx_project
```

```
master@master-vm:~$ mkdir nginx_project
master@master-vm:~$ cd nginx_project
```

Create an index.html file inside this directory:



Run a new Nginx container with your custom HTML page:

```
docker run -d -p 8081:80 --name custom-nginx -v $(pwd):/usr/share/nginx/html nginx
```

```
master@master-vn:~/nginx_project$ docker run -d -p 8081:80 --name custom-nginx -v $(pwd):/usr/share/nginx/html nginx
1298710adfb18b4b19ebd9472e4fd53900a7af119c2cd337c11ecfc626f47d6e
```

Refresh <http://localhost:8081>, and see the custom page



Hello, this is a custom Nginx page inside Docker!

