

Docker Assignment 8 (Task 1)

Step 1: Launched instances for our Jenkins Master and Slave:

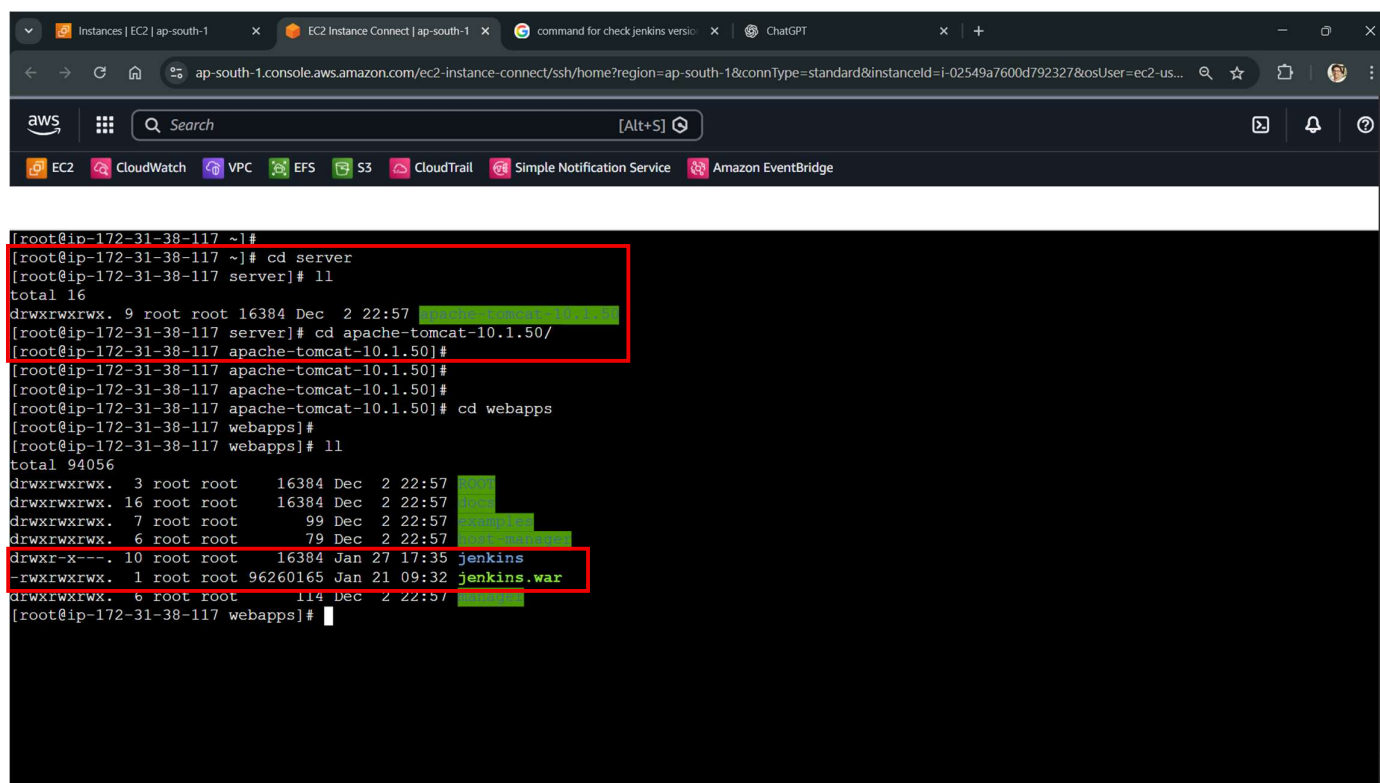
The screenshot shows the AWS Management Console for the 'Instances' page. The left sidebar contains navigation links for EC2, Dashboard, AWS Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, and Elastic Block Store. The main content area shows a list of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Three instances are listed: D7 Linux (Stopped), D8 Jenkins Master (Running), and D8 Slave (Running). The D8 Jenkins Master and D8 Slave instances are highlighted with a red box. Below the list, the 'Monitoring' section shows various metrics like CPU utilization, network in/out, and network packets in.

Step 2: Installed Java-17 and Docker and Docker Compose on the Jenkins Master Instance and Slave also:

The screenshot shows the AWS Management Console for the 'Instances' page, specifically the terminal output of the Jenkins Master instance. The terminal shows the installation of Docker and Docker Compose. The Docker version is 25.0.14, build 0bab007. Docker Compose version 5.0.1 is being installed from the GitHub repository. The installation is successful, and the files are located in /usr/local/bin.

```
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]# docker -v
Docker version 25.0.14, build 0bab007
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]# curl -SL https://github.com/docker/compose/releases/download/v5.0.1/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
100 29.8M 100 29.8M 0 0 259M 0 0 0 0 0 0 0 0 0 363M
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]# chmod +x /usr/local/bin/docker-compose
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]#
[root@ip-172-31-10-152 ~]#
```

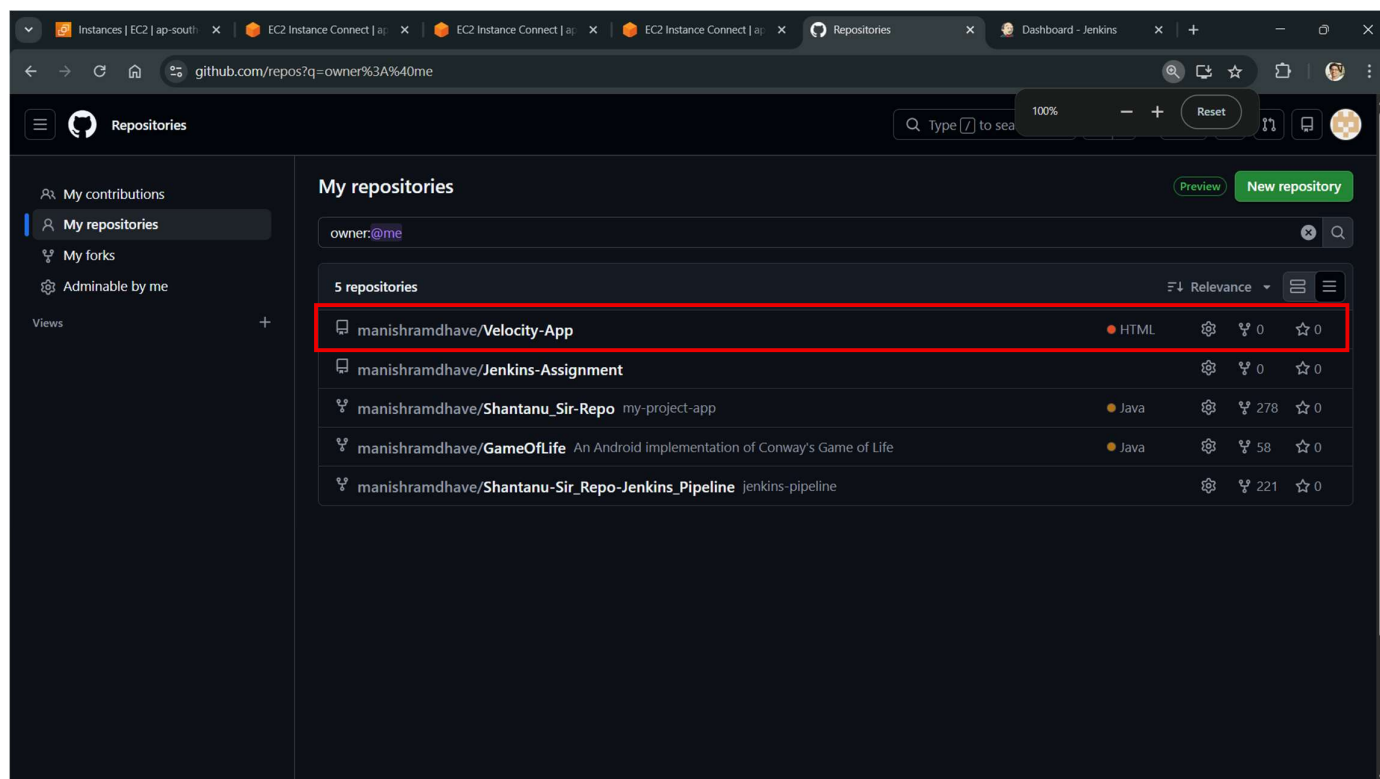
Step 3: Installed Apache-Tomcat-10 and Jenkins on the Jenkins Master Instance:



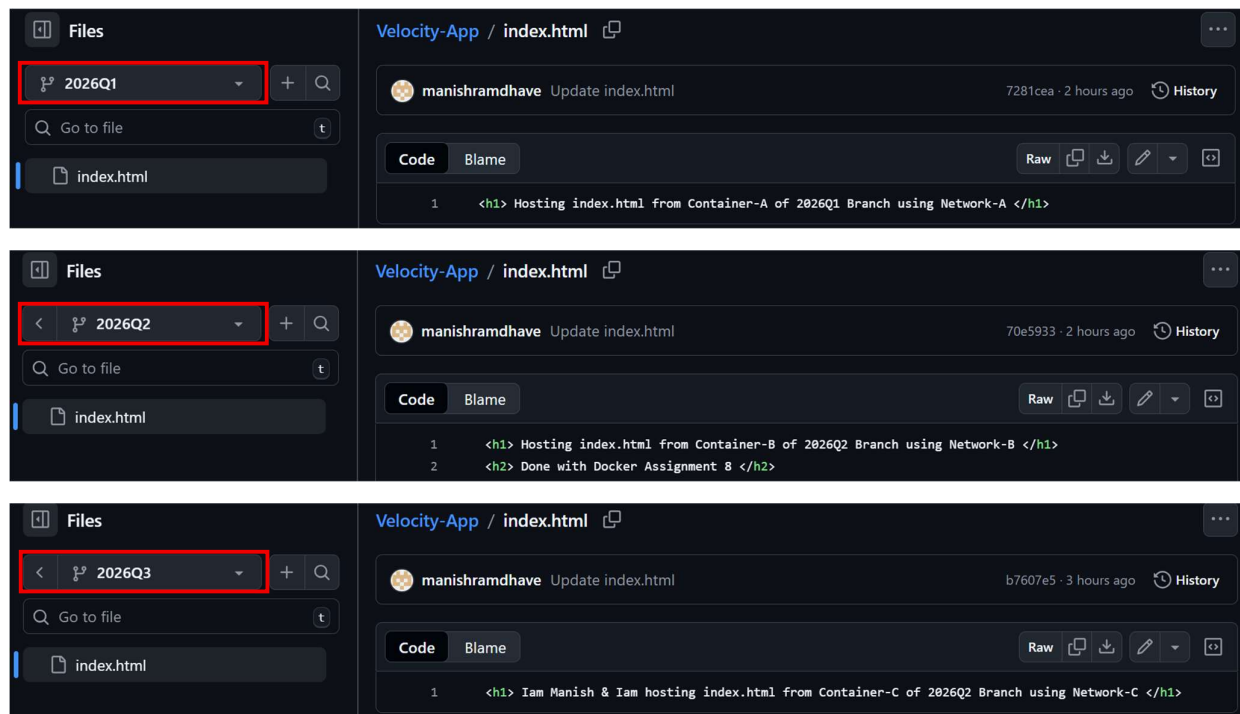
The screenshot shows the AWS Management Console with a terminal window open for an EC2 instance. The terminal output shows the following commands and results:

```
[root@ip-172-31-38-117 ~]#  
[root@ip-172-31-38-117 ~]# cd server  
[root@ip-172-31-38-117 server]# ll  
total 16  
drwxrwxrwx. 9 root root 16384 Dec 2 22:57   
[root@ip-172-31-38-117 server]# cd apache-tomcat-10.1.50/  
[root@ip-172-31-38-117 apache-tomcat-10.1.50]#  
[root@ip-172-31-38-117 apache-tomcat-10.1.50]#  
[root@ip-172-31-38-117 apache-tomcat-10.1.50]# cd webapps  
[root@ip-172-31-38-117 webapps]#  
[root@ip-172-31-38-117 webapps]# ll  
total 94056  
drwxrwxrwx. 3 root root 16384 Dec 2 22:57   
drwxrwxrwx. 16 root root 16384 Dec 2 22:57   
drwxrwxrwx. 7 root root 99 Dec 2 22:57   
drwxrwxrwx. 6 root root 79 Dec 2 22:57   
drwxr-x--. 10 root root 16384 Jan 27 17:35 jenkins  
-rwxrwxrwx. 1 root root 96260165 Jan 21 09:32 jenkins.war  
drwxrwxrwx. 6 root root 114 Dec 2 22:57   
[root@ip-172-31-38-117 webapps]#
```

Step 4: Made a Private Repository named 'Velocity-App' in GitHub account:



Step 5: Created three branches, **2026Q1**, **2026Q2** and **2026Q3** in the ‘Velocity-App’ Repository and pushed **three different ‘index.html’** files in the respective branches:



Step 6: Launched the Jenkins and created three different **Freestyle Jobs** in it:

The image shows the Jenkins dashboard with three Freestyle Jobs listed. The jobs are highlighted with a red box. The dashboard includes a sidebar with navigation options like 'New Item', 'Build History', and 'Project Relationship'. The main area shows a table of jobs with their status, name, last success, last failure, and last duration.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☁	Doc-Assign8-JobA	13 min #18	14 min #17	9.5 sec
✓	☀	Doc-Assign8-JobB	1 min 48 sec #3	N/A	9.4 sec
✓	☀	Doc-Assign8-JobC	14 min #2	N/A	5.4 sec

Icon: S M L

REST API Jenkins 2.541.1

Step 7: Created an **API Connection between Jenkins to GitHub** Repositories in ‘Manage Jenkins’ by creating a **Secret Text (Credential)** using a GitHub Token in Jenkins:

GitHub

GitHub Servers ?

GitHub Server ?

Name ?

GitHub-Server

API URL ?

https://api.github.com

Credentials ?

Git

+ Add

Test connection

☒ Manage hooks

Advanced ▾

Save Apply

Step 8: Created an ‘**Node**’ Connection between Jenkins Master and Slave instances ‘Manage Jenkins’ by creating a **SSH Username and Key (Credential)** using a Key-Pair and **Manually trusted key-verification Strategy**:

Nodes

+ New Node Configure Monitors ↻

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

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.70 GiB	0 B	4.70 GiB	0ms
	Slave	Linux (amd64)	In sync	5.09 GiB	0 B	455.69 MiB	33ms
last checked		42 min	42 min	42 min	42 min	42 min	42 min

Icon: S M L Legend

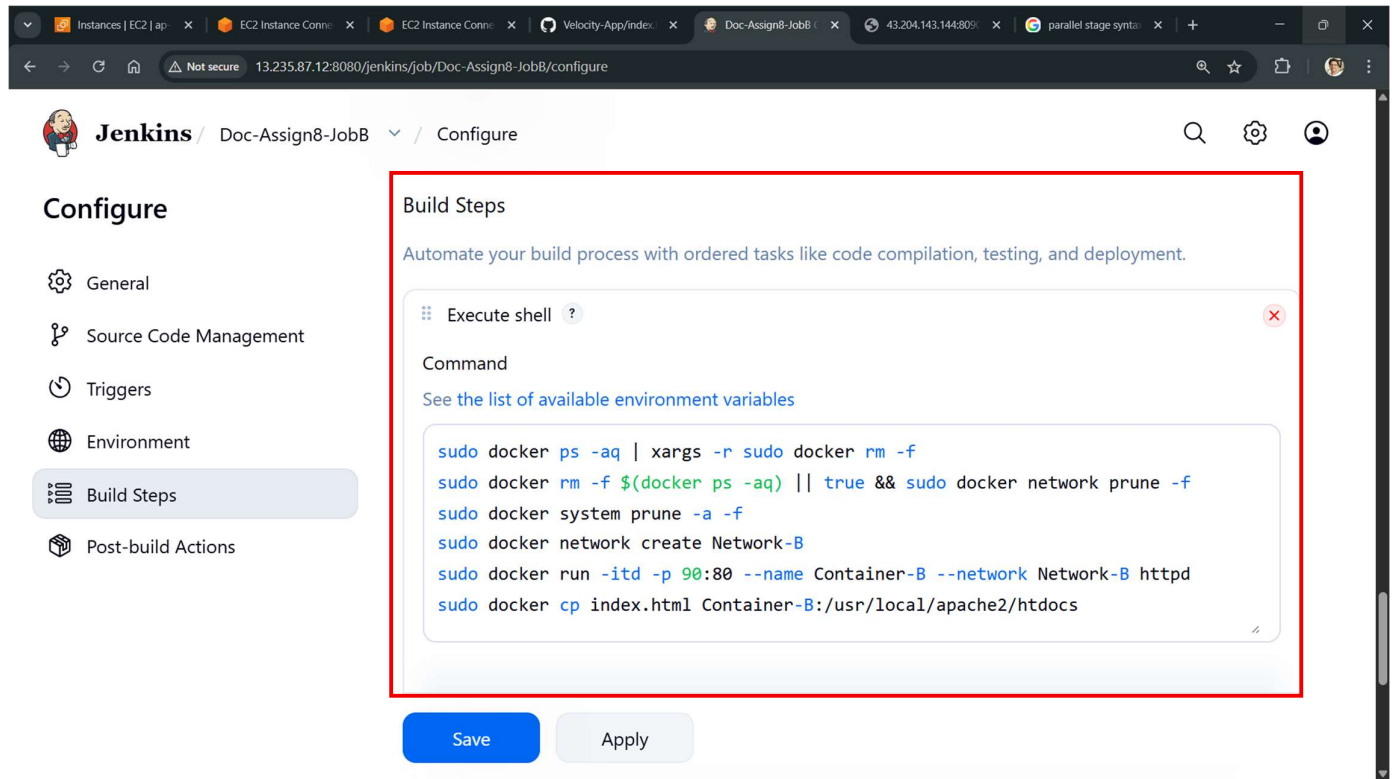
Step 9: Integrated all Git branches, **2026Q1**, **2026Q2** and **2026Q3** with Jenkins by creating ‘Credentials’ by using the Git Token **on all the Jobs** respectively:

The screenshot shows the Jenkins configuration interface for a job named 'Doc-Assign3-Pipeline-A'. The left sidebar lists configuration options: General, Triggers, Pipeline (selected), and Advanced. The main configuration area is titled 'Configure'. It contains several fields: 'Repository URL' with the value 'https://github.com/manishramdhav/Velosity-App.git', 'Credentials' with a dropdown showing 'ec2-user/***** (Git_Token)' and an '+ Add' button, an 'Advanced' dropdown, and a '+ Add Repository' button. Below these is the 'Branches to build' section, which includes a 'Branch Specifier (blank for 'any')' field containing '*/2026Q1'. At the bottom, there are 'Save' and 'Apply' buttons. A red box highlights the 'Repository URL', 'Credentials', and 'Branches to build' sections.

Step 10: In **Build Steps** of **Doc-Assign8-JobA**, we have executed the following shell commands for **Network-A**:

The screenshot shows the Jenkins configuration interface for a job named 'Doc-Assign8-JobA'. The left sidebar lists configuration options: General, Source Code Management, Triggers, Environment, Build Steps (selected), and Post-build Actions. The main configuration area is titled 'Configure'. The 'Build Steps' section is expanded, showing a list of shell commands for setting up a Docker network and running a container. The commands are: `sudo docker ps -aq | xargs -r sudo docker rm -f`, `sudo docker rm -f $(docker ps -aq) || true && sudo docker network prune -f`, `sudo docker system prune -a -f`, `sudo docker network create Network-A`, `sudo docker run -itd -p 80:80 --name Container-A --network Network-A httpd`, and `sudo docker cp index.html Container-A:/usr/local/apache2/htdocs`. At the bottom, there are 'Save' and 'Apply' buttons. A red box highlights the 'Build Steps' section.

Step 11: In **Build Steps of Doc-Assign8-JobB**, we have executed the following shell commands for **Network-B**:

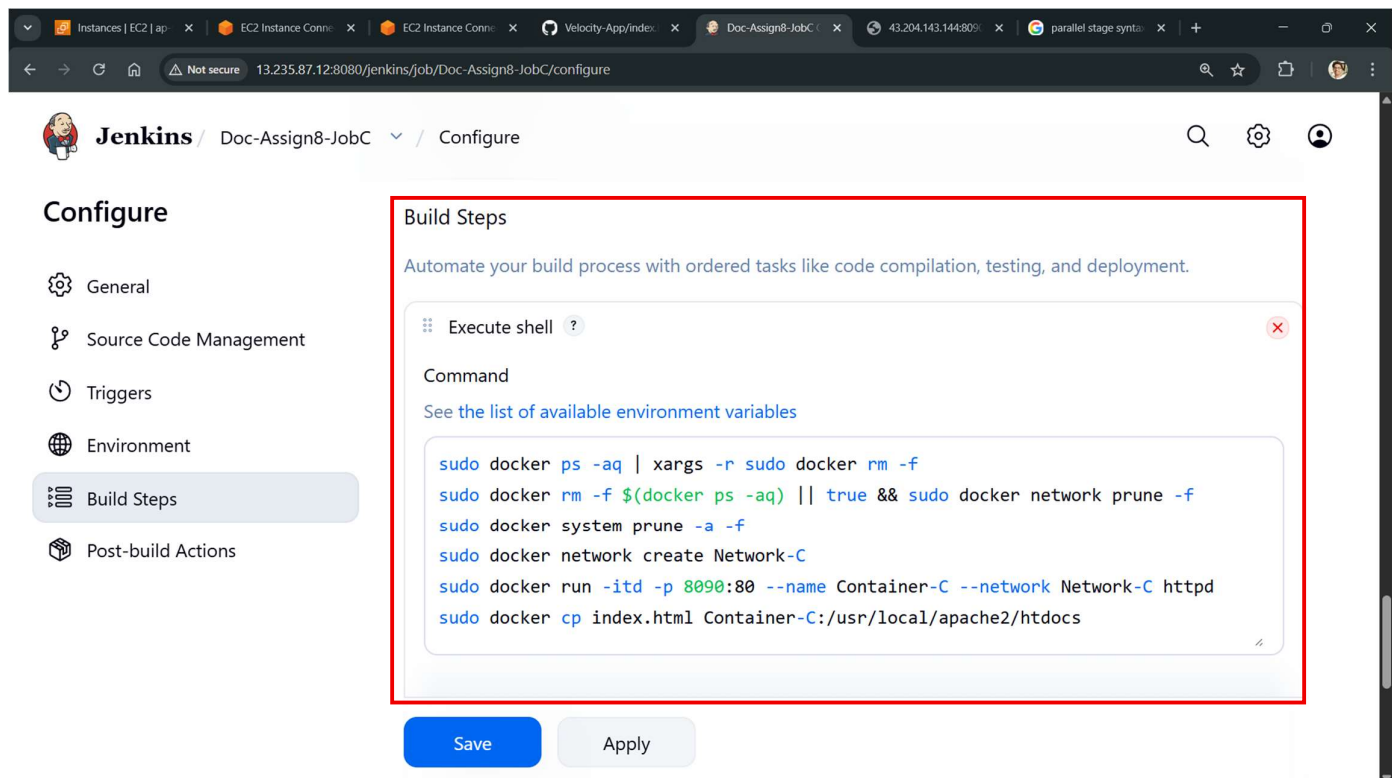


The screenshot shows the Jenkins configuration page for 'Doc-Assign8-JobB' in the 'Build Steps' section. The 'Execute shell' step is configured with the following commands:

```
sudo docker ps -aq | xargs -r sudo docker rm -f
sudo docker rm -f $(docker ps -aq) || true && sudo docker network prune -f
sudo docker system prune -a -f
sudo docker network create Network-B
sudo docker run -itd -p 90:80 --name Container-B --network Network-B httpd
sudo docker cp index.html Container-B:/usr/local/apache2/htdocs
```

The 'Save' button is highlighted in blue.

Step 12: In **Build Steps of Doc-Assign8-JobC**, we have executed the following shell commands for **Network-C**:



The screenshot shows the Jenkins configuration page for 'Doc-Assign8-JobC' in the 'Build Steps' section. The 'Execute shell' step is configured with the following commands:

```
sudo docker ps -aq | xargs -r sudo docker rm -f
sudo docker rm -f $(docker ps -aq) || true && sudo docker network prune -f
sudo docker system prune -a -f
sudo docker network create Network-C
sudo docker run -itd -p 8090:80 --name Container-C --network Network-C httpd
sudo docker cp index.html Container-C:/usr/local/apache2/htdocs
```

The 'Save' button is highlighted in blue.

Results:

1. When changes are done in **2026Q1 branch**, Build is triggered by '**Doc-Assign8-JobA**' and the updated index.html file is hosted from the '**Container-A**' of '**Network-A**' by following the shell script in the '**Build Steps**' file of the same branch and hosted the using **Port No.80**:

The screenshot displays the GitHub repository for 'Velocity-App' at the '2026Q1' branch. The 'index.html' file is highlighted, showing two lines of code: `<h1> Hosting index.html from Container-A of 2026Q1 Branch using Network-A </h1>` and `<h2> Change in A </h2>`. To the right, the Jenkins configuration for 'Doc-Assign8-JobA' is shown, featuring a shell script that sets up Docker containers and networks. Below this, the Jenkins build #24 is displayed, indicating it was successful and started by user 'mmm'.

```
sudo docker ps -aq | xargs -r sudo docker rm -f
sudo docker rm -f $(docker ps -aq) || true && sudo docker network prune -f
sudo docker system prune -a -f
sudo docker network create Network-A
sudo docker run -itd -p 80:80 --name Container-A --network Network-A httpd
sudo docker cp index.html Container-A:/usr/local/apache2/htdocs
```

Hosting index.html from Container-A of 2026Q1 Branch using Network-A

Change in A

2. When changes are done in **2026Q2 branch**, Build is triggered by '**Doc-Assign8-JobB**' and the updated index.html file is hosted from the '**Container-B**' of '**Network-B**' by following the shell script in the '**Build Steps**' file of the same branch and hosted the using **Port No.90**:

The screenshot displays the GitHub repository for 'Velocity-App' at the '2026Q2' branch. The 'index.html' file is highlighted, showing two lines of code: `<h1> Hosting index.html from Container-B of 2026Q2 Branch using Network-B </h1>` and `<h2> Done with Docker Assignment 8 </h2>`. To the right, the Jenkins configuration for 'Doc-Assign8-JobB' is shown, featuring a shell script that sets up Docker containers and networks. Below this, the Jenkins build #7 is displayed, indicating it was successful and started by user 'mmm'.

```
sudo docker ps -aq | xargs -r sudo docker rm -f
sudo docker rm -f $(docker ps -aq) || true && sudo docker network prune -f
sudo docker system prune -a -f
sudo docker network create Network-B
sudo docker run -itd -p 90:80 --name Container-B --network Network-B httpd
sudo docker cp index.html Container-B:/usr/local/apache2/htdocs
```

Hosting index.html from Container-B of 2026Q2 Branch using Network-B

Done with Docker Assignment 8

3. When changes are done in **2026Q3** branch, Build is triggered by '**Doc-Assign8-JobC**' and the updated index.html file is hosted from the '**Container-C**' of '**Network-C**' by following the shell script in the '**Build Steps**' file of the same branch and hosted the using **Port No.8090**:

The workflow is demonstrated through four screenshots:

- Top Left:** GitHub repository view for `Velocity-App/index.html` in the `2026Q3` branch. A commit by `manishramdhav` is shown with the message "Update index.html". The code content is:

```
<h1> Iam Manish & Iam hosting index.html from Container-C of 2026Q2 Branch using Network-C </h1>
<h2> Change 1 </h2>
```
- Top Right:** Jenkins configuration page for `Doc-Assign8-JobC`. The 'Configure' tab shows a shell script in the 'Build Steps' section:

```
sudo docker ps -aq | xargs -r sudo docker rm -f
sudo docker rm -f $(docker ps -aq) || true && sudo docker network prune -f
sudo docker system prune -a -f
sudo docker network create Network-C
sudo docker run -itd -p 8090:80 --name Container-C --network Network-C httpd
sudo docker cp index.html Container-C:/usr/local/apache2/htdocs
```
- Bottom Left:** Jenkins build log for `#7 (16 Feb 2026, 14:04:53)`. The log indicates the build started by user `mmm` and completed successfully. The run spent time is detailed as:
 - 3 ms waiting;
 - 9.9 sec build duration;
 - 9.9 sec total from scheduled to completion.
- Bottom Right:** Browser view of the hosted file at `43.204.143.144:8090/index.html`. The page displays the content from the index.html file:

Iam Manish & Iam hosting index.html from Container-C of 2026Q2 Branch using Network-C

Change 1