

Docker Assignment 8 (Task 1)

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

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like Dashboard, AWS Global View, Events, Instances (selected), 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 area displays 'Instances (3/13) info' with a table. The table has columns: Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. The rows are: D7 Linux (Stopped, t3.micro, -), D8 Jenkins Master (Running, t3.micro, 3/3 checks passed, ap-south-1b), and D8 Slave (Running, t3.micro, 3/3 checks passed, ap-south-1b). A red box highlights the D8 Jenkins Master and D8 Slave rows. Below the table, it says '3 instances selected'. There's a 'Monitoring' section with four charts: CPU utilization (%), Network in (bytes), Network out (bytes), and Network packets in (count). The bottom right corner shows copyright information: © 2026, Amazon Web Services, Inc. or its affiliates.

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

The screenshot shows a CloudShell session for the instance i-019dd75cde023be5d (D8 Jenkins Master). The terminal output is as follows:

```
[root@ip-172-31-10-152 ~]# [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 [root@ip-172-31-10-152 ~]# % Total % Received % Xferd Average Speed Time Time Current Dload Upload Total Spent Left Speed 0 0 0 0 0 0 0 --:--:-- --:--:-- 0 100 29.8M 100 29.8M 0 0 258M 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 ~]#
```

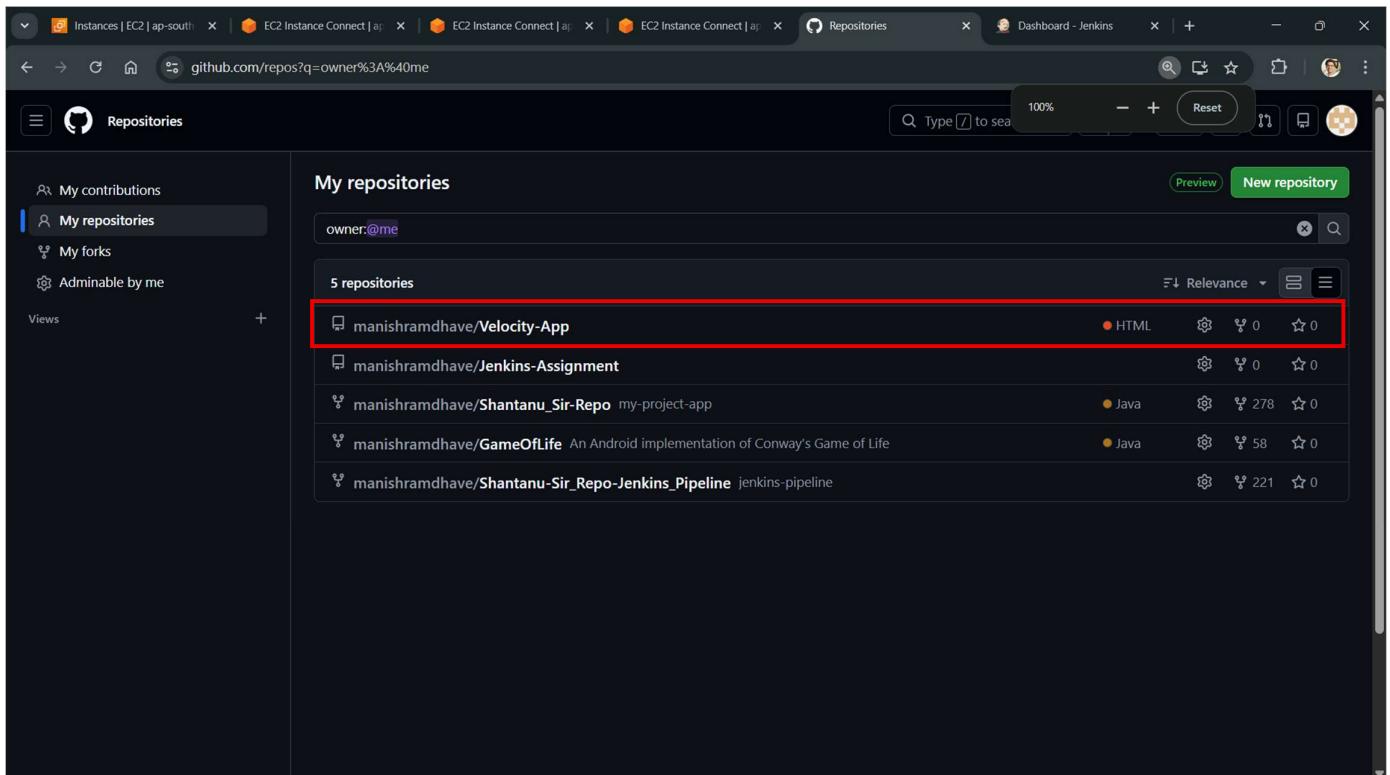
At the bottom, it shows the instance details: i-019dd75cde023be5d (D8 Jenkins Master), Public IPs: 3.110.54.15, Private IP: 172.31.10.152. The bottom right corner shows copyright information: © 2026, Amazon Web Services, Inc. or its affiliates.

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

The screenshot shows a terminal session on an EC2 instance. The user is navigating through directory structures and listing files to verify the installations. A red box highlights the command `cd apache-tomcat-10.1.50/` and the resulting directory listing, which includes the `jenkins.war` file.

```
[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 apache-tomcat-10.1.50/
[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 ROOT
drwxrwxrwx. 16 root root 16384 Dec  2 22:57 docs
drwxrwxrwx. 7 root root 99 Dec  2 22:57 examples
drwxrwxrwx. 6 root root 79 Dec  2 22:57 jenkins
drwxr-x--. 10 root root 16384 Jan 27 17:35 jenkins
-rw-rw-rwx. 1 root root 96260165 Jan 21 09:32 jenkins.war
drwxrwxrwx. 6 root root 114 Dec  2 22:57 war
[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:

The image contains three vertically stacked screenshots of a Git interface, likely GitHub, showing the contents of the 'Velocity-App' repository.

- Screenshot 1 (Top):** Shows the '2026Q1' branch selected in the dropdown menu. The 'index.html' file is displayed with the following content:

```
<h1> Hosting index.html from Container-A of 2026Q1 Branch using Network-A </h1>
```
- Screenshot 2 (Middle):** Shows the '2026Q2' branch selected in the dropdown menu. The 'index.html' file is displayed with the following content:

```
<h1> Hosting index.html from Container-B of 2026Q2 Branch using Network-B </h1>
<h2> Done with Docker Assignment 8 </h2>
```
- Screenshot 3 (Bottom):** Shows the '2026Q3' branch selected in the dropdown menu. The 'index.html' file is displayed with the following content:

```
<h1> Iam Manish & Iam hosting index.html from Container-C of 2026Q3 Branch using Network-C </h1>
```

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

The image shows the Jenkins dashboard at the URL <http://3.110.54.15:8080/jenkins/>. The dashboard displays the following information:

- Left Sidebar:** Includes links for 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Build Queue' (No builds in the queue), 'Build Executor Status' (Built-In Node: 0/2, Slave: 0/10), and status icons for 'Icon: S M L'.
- Central View:** A table listing three Freestyle jobs:

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
- Bottom Right:** Links for 'REST API' and '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:

The screenshot shows the Jenkins 'GitHub' configuration page. A red box highlights the 'Credentials' section, which contains a single item named 'Git'. Below this, there are fields for 'Name' (set to 'GitHub-Server') and 'API URL' (set to 'https://api.github.com'). There is also a checked checkbox for 'Manage hooks' and an 'Advanced' dropdown. At the bottom are 'Save' and 'Apply' buttons.

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:

The screenshot shows the Jenkins 'Nodes' management page. A red box highlights the 'Slave' node entry in the list. The table columns include S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The 'Slave' node has a status of 'In sync' and 5.09 GiB free disk space. At the bottom, there are icons for 'Icon: S M L' and a 'Legend' section.

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

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 Pipeline configuration page for 'Doc-Assign3-Pipeline-A'. The 'Pipeline' tab is selected on the left. A red box highlights the main configuration area. Inside, the 'Repository URL' field contains 'https://github.com/manishramdhav/Velocity-App.git'. The 'Credentials' dropdown shows 'ec2-user/******** (Git_Token)'. Below these, there's an 'Advanced' dropdown and a '+ Add Repository' button. Under 'Branches to build', a red box highlights the 'Branch Specifier' field containing '/2026Q1'. At the bottom are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins Job configuration page for 'Doc-Assign8-JobA'. The 'Build Steps' tab is selected on the left. A red box highlights the 'Execute shell' section. Inside, the 'Command' field contains the following shell script:

```
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
```

At the bottom are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins job configuration page for 'Doc-Assign8-JobB'. The left sidebar has links for General, Source Code Management, Triggers, Environment, Build Steps (which is selected and highlighted in grey), and Post-build Actions. The main content area is titled 'Build Steps' with the sub-instruction 'Automate your build process with ordered tasks like code compilation, testing, and deployment.' Below this is an 'Execute shell' step. The 'Command' field contains the following shell script:

```
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
```

At the bottom are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins job configuration page for 'Doc-Assign8-JobC'. The left sidebar has links for General, Source Code Management, Triggers, Environment, Build Steps (selected and highlighted in grey), and Post-build Actions. The main content area is titled 'Build Steps' with the sub-instruction 'Automate your build process with ordered tasks like code compilation, testing, and deployment.' Below this is an 'Execute shell' step. The 'Command' field contains the following shell script:

```
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
```

At the bottom are 'Save' and 'Apply' buttons.

Results:

- 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 four windows illustrating the process for Network-A:

- GitHub Commit History:** Shows a commit from "manishramdhave" updating index.html in the "Velocity-App" repository's "2026Q1" branch. The commit message includes the Docker command used.
- Jenkins Job Configuration:** Shows the Jenkins configuration for "Doc-Assign8-JobA" with the shell script for Container-A.
- Jenkins Build Log:** Shows the Jenkins build log for "Doc-Assign8-JobA #24" (16 Feb 2026, 14:00:51), indicating the build was started by "mmm".
- Browser Output:** Shows the browser output at port 80 (43.204.143.144/index.html) displaying the updated index.html content.

- 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 four windows illustrating the process for Network-B:

- GitHub Commit History:** Shows a commit from "manishramdhave" updating index.html in the "Velocity-App" repository's "2026Q2" branch. The commit message includes the Docker command used.
- Jenkins Job Configuration:** Shows the Jenkins configuration for "Doc-Assign8-JobB" with the shell script for Container-B.
- Jenkins Build Log:** Shows the Jenkins build log for "Doc-Assign8-JobB #7" (16 Feb 2026, 14:03:25), indicating the build was started by "mmm".
- Browser Output:** Shows the browser output at port 90 (43.204.143.144:90/index.html) displaying the updated index.html content.

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 image displays four browser tabs illustrating a Continuous Integration/Continuous Deployment (CI/CD) pipeline:

- Top Left Tab:** GitHub pull request for Velocity-App/index.html in the 2026Q3 branch. It shows a commit by manishramdhave updating index.html, which contains the following code:

```
<h1> Iam Manish & Iam hosting index.html from Container-C of 2026Q2 Branch using Network-C </h1>
<h2> Change 1 </h2>
```
- Top Right Tab:** Jenkins job configuration for Doc-Assign8-JobC. The "Build Steps" section contains the following shell script:

```
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 Tab:** Jenkins build log for #7 (16 Feb 2026, 14:04:53). It shows the build was started by user mmm and took 9.9 seconds on a Slave. The log also lists the time spent:
 - 3 ms waiting;
 - 9.9 sec build duration;
 - 9.9 sec total from scheduled to completion.
- Bottom Right Tab:** A browser window showing the deployed index.html content at 43.204.143.144:8090/index.html. The page displays the message: "Iam Manish & Iam hosting index.html from Container-C of 2026Q2 Branch using Network-C" and "Change 1".