

Docker Assignment 8 (Task 2)

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

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation 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 search bar and filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Three instances are listed: D7 Linux (Stopped, t3.micro, ap-south-1a), D8 Jenkins Master (Running, t3.micro, ap-south-1b), and D8 Slave (Running, t3.micro, ap-south-1b). A red box highlights the D8 Jenkins Master and D8 Slave rows. Below the instances, a section titled '3 instances selected' shows monitoring metrics for CPU utilization, Network in, Network out, and Network packets in. At the bottom, there are links for CloudShell, Feedback, and Console Mobile App, along with copyright information for 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 terminal window. The user is root on an EC2 instance with IP 172.31.10.152. The terminal output shows the following commands and their results:

```
[root@ip-172-31-10-152 ~]# docker -v
Docker version 25.0.14, build 0bab007
[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  Current
          Dload  Upload   Total Spent  Left  Speed
0  29.8M  100 29.8M  0      0  258M  0  --:--:--  --:--:-- 363M
[root@ip-172-31-10-152 ~]# chmod +x /usr/local/bin/docker-compose
```

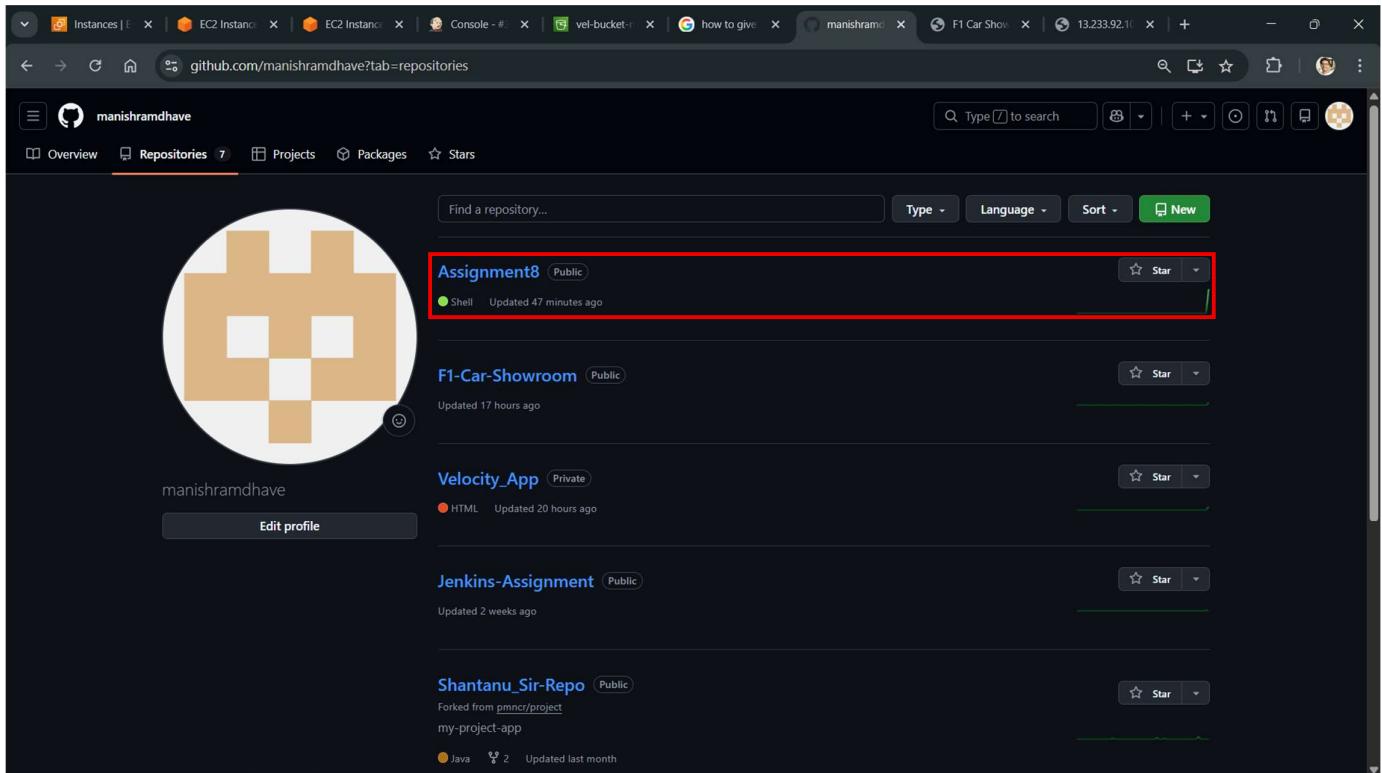
Below the terminal, a modal window titled 'i-019dd75cde023be5d (D8 Jenkins Master)' shows the public and private IP addresses: Public IPs: 3.110.54.15 Private IPs: 172.31.10.152.

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

The screenshot shows a terminal session on an AWS EC2 instance. The user is navigating through the file system to verify the installation of Apache-Tomcat-10 and Jenkins. A red box highlights the directory structure under /server/apache-tomcat-10.1.50/. Another red box highlights the Jenkins WAR file, jenkins.war, located in the webapps directory.

```
[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]# [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 doc [root@ip-172-31-38-117 webapps]# drwxrwxrwx. 16 root root 16384 Dec 2 22:57 docs [root@ip-172-31-38-117 webapps]# drwxrwxrwx. 7 root root 99 Dec 2 22:57 examples [root@ip-172-31-38-117 webapps]# drwxrwxrwx. 6 root root 79 Dec 2 22:57 host-manager [root@ip-172-31-38-117 webapps]# drwxr-x---. 10 root root 16384 Jan 27 17:35 jenkins [root@ip-172-31-38-117 webapps]# -rwxrwxrwx. 1 root root 96260165 Jan 21 09:32 jenkins.war [root@ip-172-31-38-117 webapps]# [root@ip-172-31-38-117 webapps]#
```

Step 4: Made a Private Repository named ‘Assignment8’ in GitHub account:



Step 5: Created three branches, **2026Q1**, **2026Q2** and **2026Q3** in the ‘Velocity-App’ Repository and pushed **three different ‘Access.sh’, ‘Jenkinsfile’and ‘docker-compose.yaml’** files in all the respective branches:

The screenshot shows three GitHub repository branches: 2026Q1, 2026Q2, and 2026Q3. Each branch has 3 branches and 0 tags. The 2026Q1 branch has 72 commits, 2026Q2 has 10 commits, and 2026Q3 has 9 commits. All branches have 'Access.sh', 'Jenkinsfile', and 'docker-compose.yaml' files.

- 2026Q1**: 72 commits, 3 branches, 0 tags. Commits: Create Access.sh (19 hours ago), Update Jenkinsfile (1 hour ago), Update docker-compose.yaml (1 hour ago).
- 2026Q2**: 10 commits, 3 branches, 0 tags. Commits: Create Access.sh (19 hours ago), Update Jenkinsfile (50 minutes ago), Update docker-compose.yaml (52 minutes ago).
- 2026Q3**: 9 commits, 3 branches, 0 tags. Commits: Create Access.sh (19 hours ago), Update Jenkinsfile (50 minutes ago), Update docker-compose.yaml (53 minutes ago).

Step 6: 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 Manage Jenkins > System > GitHub configuration page. It displays a GitHub server configuration with the following details:

- Name**: GitHub-Server
- API URL**: https://api.github.com
- Credentials**: Git (selected)

The 'Git' credential is highlighted with a red box. There is also a checkbox for 'Manage hooks' and a 'Test connection' button.

Step 7: Launched the Jenkins and created three different Pipeline Jobs in it:

The screenshot shows the Jenkins dashboard with three Pipeline jobs listed in a table:

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	Cloud with rain icon	Doc-Assign8-Pipeline-A	1 hr 55 min #48	1 hr 55 min #47	22 sec
✓	Sun icon	Doc-Assign8-Pipeline-B	1 hr 46 min #3	N/A	24 sec
✓	Cloud icon	Doc-Assign8-Pipeline-C	41 sec #2	16 hr #1	33 sec

Build Queue: No builds in the queue.

Build Executor Status: Built-In Node 0/2, Slave 0/10.

Icon: S M L.

REST API Jenkins 2.541.1

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 Manage Jenkins / Nodes page with the Slave node highlighted in a red box:

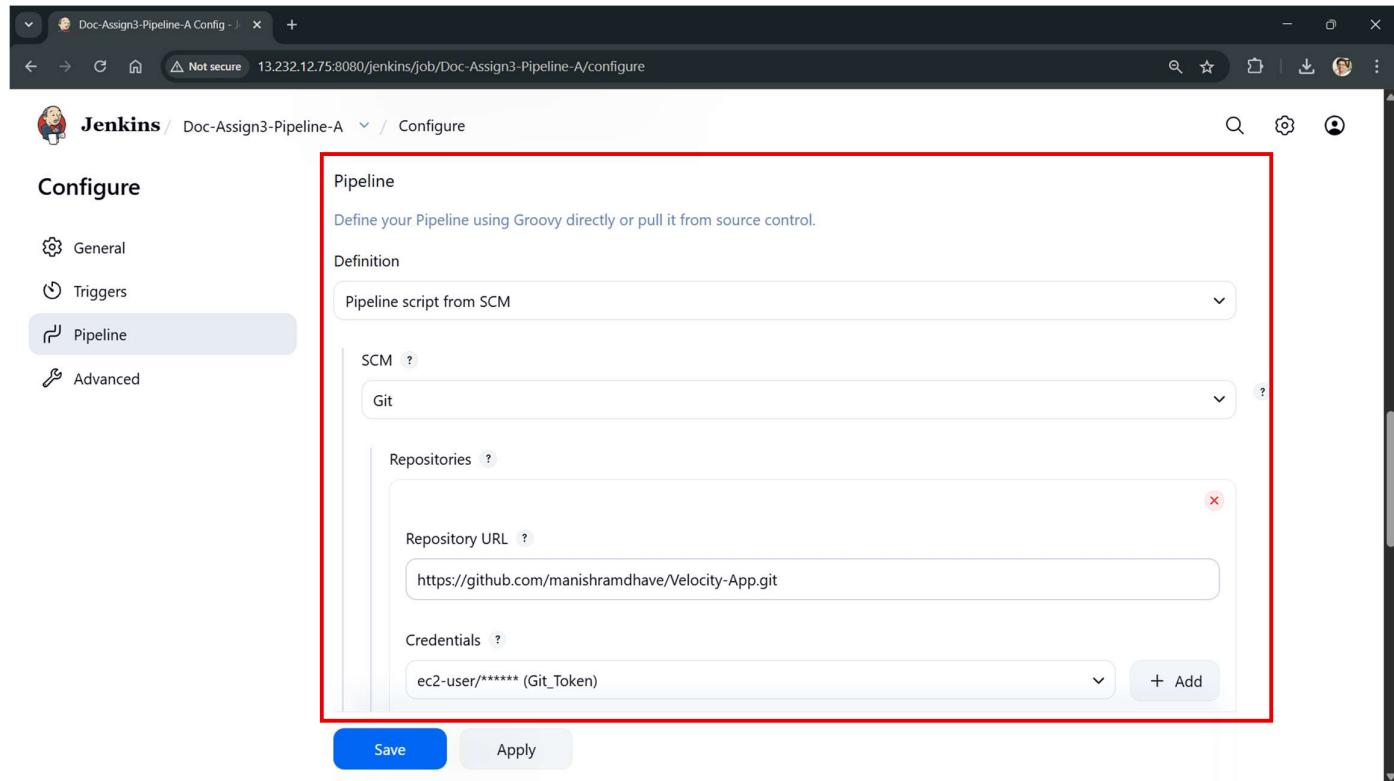
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
Monitor icon	Built-In Node	Linux (amd64)	In sync	4.70 GiB	! 0 B	4.70 GiB	0ms
Monitor icon	Slave	Linux (amd64)	In sync	5.09 GiB	! 0 B	455.69 MiB	33ms

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

Icon: S M L.

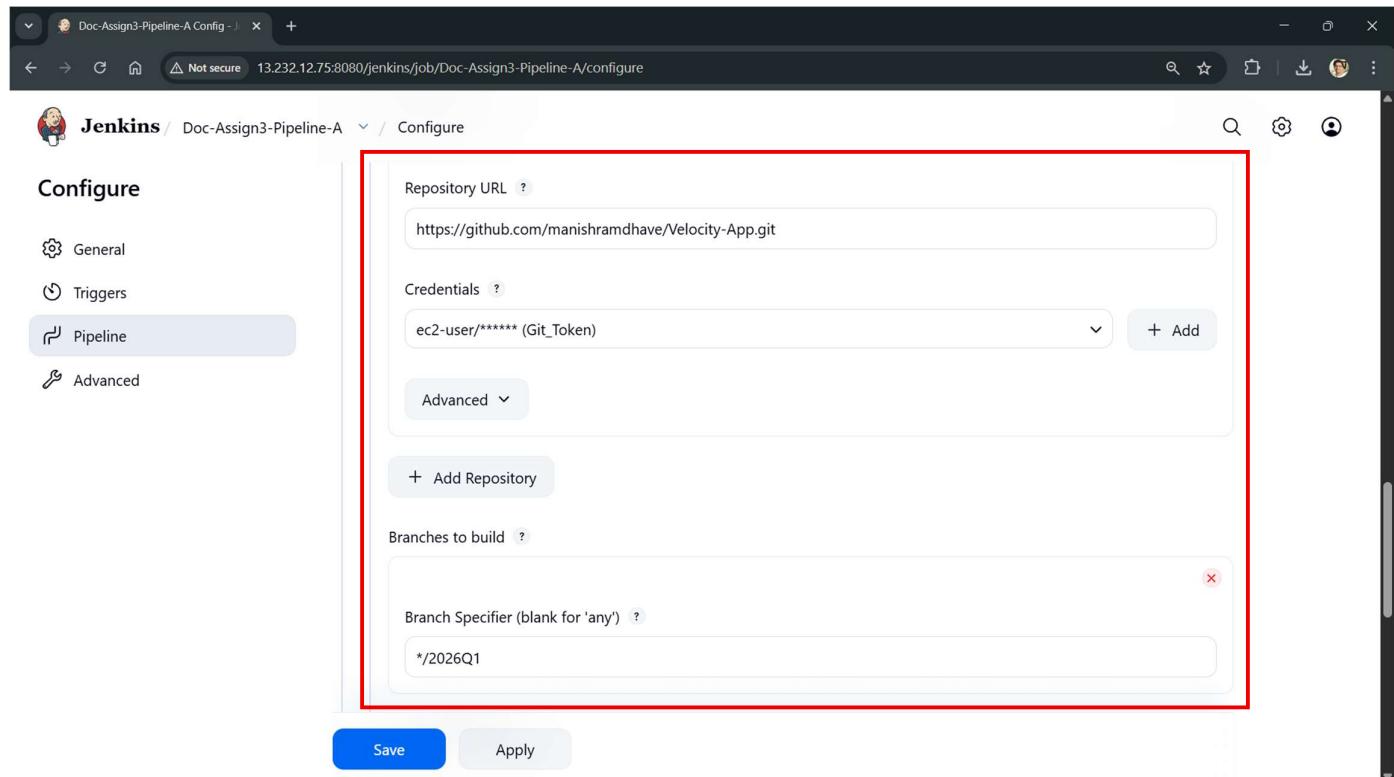
Legend REST API Jenkins 2.541.1

Step 9: Used ‘Pipeline script from SCM’ and selected the SCM as a ‘Git’ which will follow the Jenkins pipeline script by using the ‘Jenkinsfile’ in the repository **in each pipeline job:**



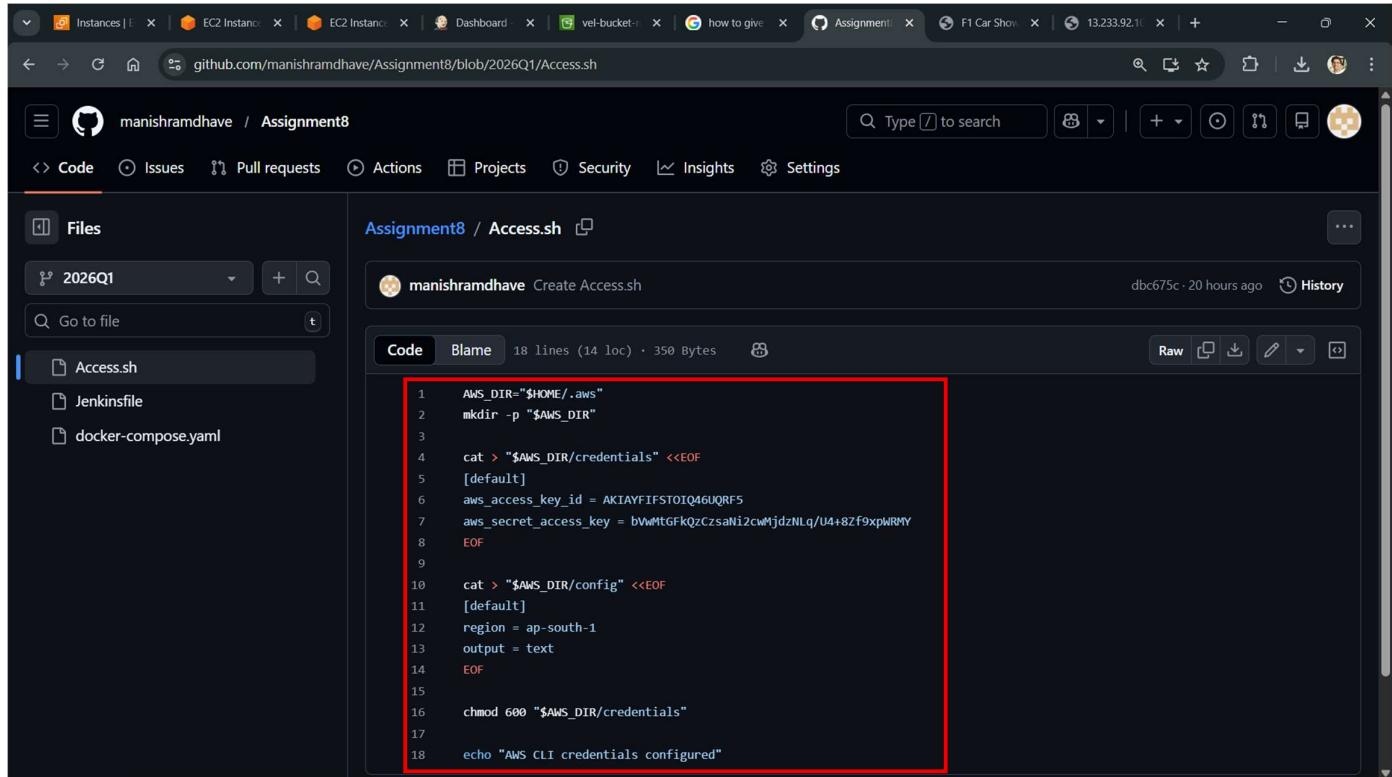
The screenshot shows the Jenkins Pipeline configuration page for a job named "Doc-Assign3-Pipeline-A". The left sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled "Pipeline" and contains a "Definition" section set to "Pipeline script from SCM". Below this is an "SCM" section set to "Git". Under "Repositories", there is one entry with a red "x" icon. The "Repository URL" is set to "https://github.com/manishramdhav/Velocity-App.git" and the "Credentials" are set to "ec2-user/******** (Git_Token)". There are "Save" and "Apply" buttons at the bottom.

Step 10: Integrated all Git branches, **2026Q1, 2026Q2 and 2026Q3 with Jenkins by creating ‘Credentials’ by using the Git Token **on all the pipeline** respectively:**



The screenshot shows the Jenkins Pipeline configuration page for the same job. The left sidebar has tabs for General, Triggers, Pipeline (selected), and Advanced. The main area shows the "Repository URL" as "https://github.com/manishramdhav/Velocity-App.git" and the "Credentials" as "ec2-user/******** (Git_Token)". A red box highlights the "Advanced" dropdown and the "Branches to build" section. In "Branches to build", the "Branch Specifier" is set to "*2026Q1". There are "Save" and "Apply" buttons at the bottom.

Step 11: Added AWS Configurations (i.e. Access Key, Secret Key etc.) in the ‘Access.sh’ file which run by Jenkinsfile for AWS CLI command processes in each Branch:



The screenshot shows a GitHub repository page for 'Assignment8'. The 'Code' tab is selected, displaying the 'Access.sh' file. The file contains AWS configuration code, specifically for setting up credentials and region. A red box highlights the sensitive AWS key and secret key values.

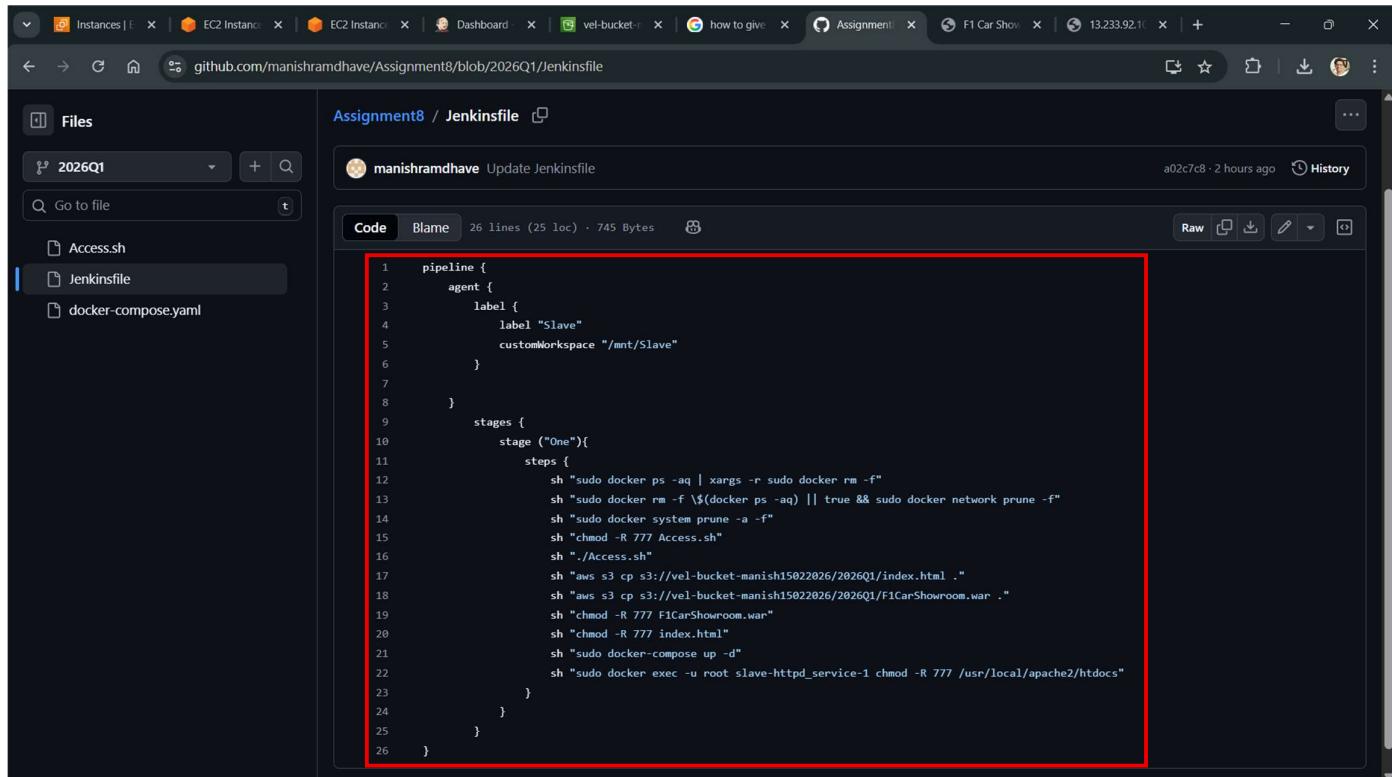
```
AWS_DIR="$HOME/.aws"
mkdir -p "$AWS_DIR"

cat > "$AWS_DIR/credentials" <<EOF
[default]
aws_access_key_id = AKIAVFIESTOIQ46UQRFS
aws_secret_access_key = bWMMtgFkQzCzaNi2cwMjdzNLq/U4+8Zf9xpwRMY
EOF

cat > "$AWS_DIR/config" <<EOF
[default]
region = ap-south-1
output = text
EOF

chmod 600 "$AWS_DIR/credentials"
echo "AWS CLI credentials configured"
```

Step 12: Added AWS Configurations (i.e. Access Key, Secret Key etc.) in the ‘Access.sh’ file which run by Jenkinsfile for AWS CLI command processes:



The screenshot shows a GitHub repository page for 'Assignment8'. The 'Code' tab is selected, displaying the 'Jenkinsfile'. The file defines a Jenkins pipeline with stages and steps, including AWS CLI commands for Docker and S3 operations. A red box highlights the AWS access key and secret key values used in the pipeline steps.

```
pipeline {
    agent {
        label {
            label "Slave"
            customWorkspace "/mnt/Slave"
        }
    }
    stages {
        stage ("One"){
            steps {
                sh "sudo docker ps -aq | xargs -r sudo docker rm -f"
                sh "sudo docker rm -f $(docker ps -aq) || true && sudo docker network prune -f"
                sh "sudo docker system prune -a -f"
                sh "chmod -R 777 Access.sh"
                sh "./Access.sh"
                sh "aws s3 cp s3://vel-bucket-manish15822026/2026Q1/index.html ."
                sh "aws s3 cp s3://vel-bucket-manish15822026/2026Q1/F1CarShowroom.war ."
                sh "chmod -R 777 F1CarShowroom.war"
                sh "chmod -R 777 index.html"
                sh "sudo docker-compose up -d"
                sh "sudo docker exec -u root slave-httpd_service-1 chmod -R 777 /usr/local/apache2/htdocs"
            }
        }
    }
}
```

Step 13: Added ‘docker-compose.yaml’ file which will run by Jenkinsfile for Container creation in each branch:

The screenshot shows a GitHub repository page for 'Assignment8'. In the left sidebar, under 'Files', the 'docker-compose.yaml' file is selected. The main area displays the YAML code for the docker-compose file, which defines two services: 'htpd_service' and 'tomcat_service', each with specific configurations like image, ports, volumes, and networks. A red box highlights the entire code block.

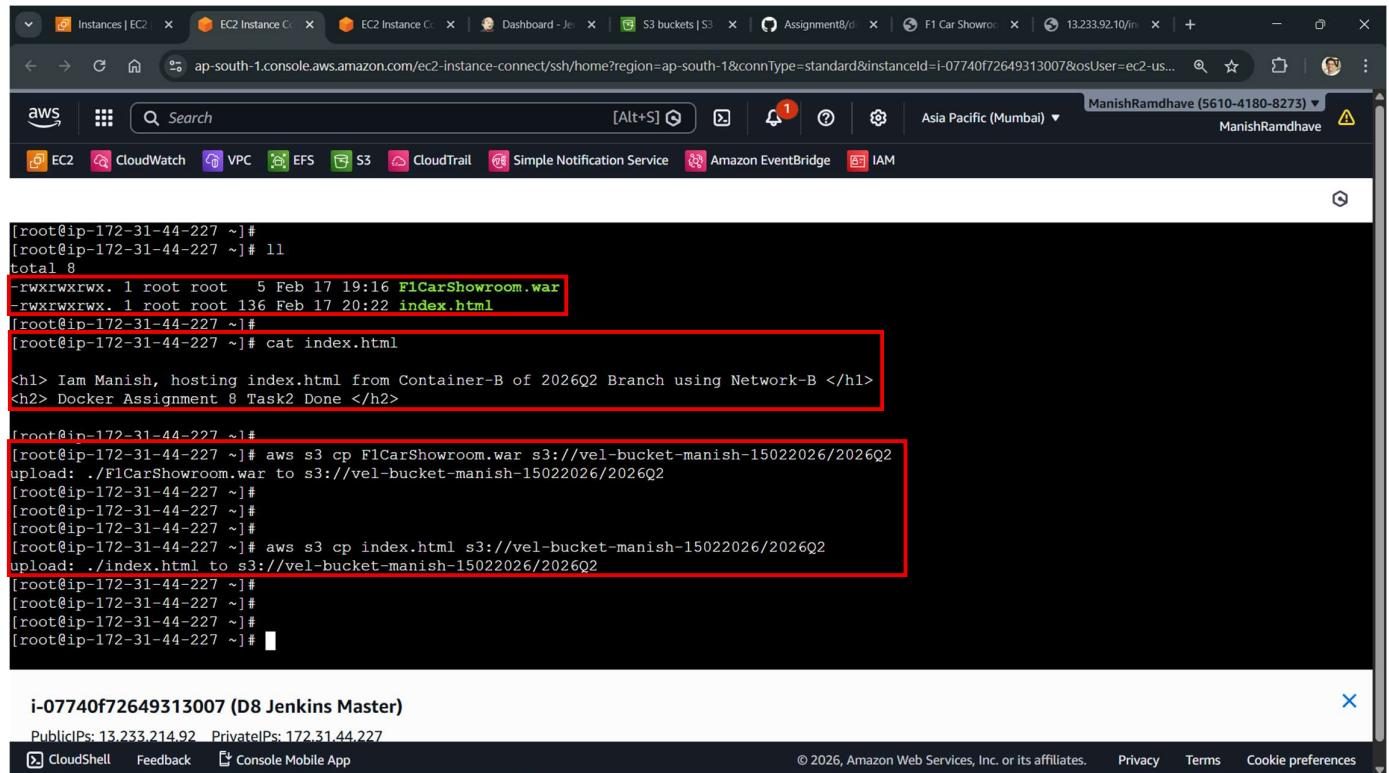
```
1 services:
2   htpd_service:
3     image: 'httpd'
4     networks:
5       - 'Network-A'
6     ports:
7       - '80:80'
8     volumes:
9       - '/mnt/Slave:/usr/local/apache2/htdocs'
10
11   tomcat_service:
12     image: 'tomcat:9'
13     networks:
14       - 'Network-B'
15     ports:
16       - '8080:8080'
17     volumes:
18       - '/mnt/Slave:/usr/local/tomcat/webapps/'
19
20   networks:
21     Network-A:
22     Network-B:
```

Step 14: By using the AWS CLI commands, we have uploaded the ‘F1CarShowroom.war’ and ‘index.html’ files in the ‘vel-bucket-manish15022026/2026Q1’ S3 Bucket folder:

The screenshot shows an AWS CloudShell terminal window. The user has uploaded a WAR file and an HTML file to an S3 bucket. The terminal output shows the upload command and the resulting object details in the S3 bucket. Two specific lines of the terminal output are highlighted with red boxes: the AWS configuration command and the AWS s3 cp command used to upload the files.

```
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]# ll
total 8
-rwxrwxrwx. 1 root root 5 Feb 17 19:16 F1CarShowroom.war
-rwxrwxrwx. 1 root root 117 Feb 17 20:18 index.html
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]# cat index.html
<h1> Iam Manish, hosting index.html from Container-A of 2026Q1 Branch using Network-A </h1>
<h2> Change in A </h2>
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]# aws configure
AWS Access Key ID [*****QRFP]: AWS Secret Access Key [*****WRMY]: Default region name [ap-south-1]: Default output format [text]: [root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]# aws s3 mb s3://vel-bucket-manish-15022026
make_bucket: vel-bucket-manish-15022026
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]# aws s3 cp F1CarShowroom.war s3://vel-bucket-manish-15022026/2026Q1
upload: ./F1CarShowroom.war to s3://vel-bucket-manish-15022026/2026Q1
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]# aws s3 cp index.html s3://vel-bucket-manish-15022026/2026Q1
upload: ./index.html to s3://vel-bucket-manish-15022026/2026Q1
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]#
```

Step 15: By using the AWS CLI commands, we have uploaded the ‘F1CarShowroom.war’ and ‘index.html’ files in the ‘vel-bucket-manish15022026/2026Q1’ S3 Bucket folder:



The screenshot shows a terminal session in AWS CloudShell. The user has uploaded the 'F1CarShowroom.war' and 'index.html' files to the 'vel-bucket-manish15022026/2026Q1' S3 bucket. The terminal output is as follows:

```
[root@ip-172-31-44-227 ~]# ls
[root@ip-172-31-44-227 ~]# ll
total 8
-rwxrwxrwx. 1 root root 5 Feb 17 19:16 F1CarShowroom.war
-rwxrwxrwx. 1 root root 136 Feb 17 20:22 index.html
[root@ip-172-31-44-227 ~]# cat index.html
<h1> Iam Manish, hosting index.html from Container-B of 2026Q2 Branch using Network-B </h1>
<h2> Docker Assignment 8 Task2 Done </h2>

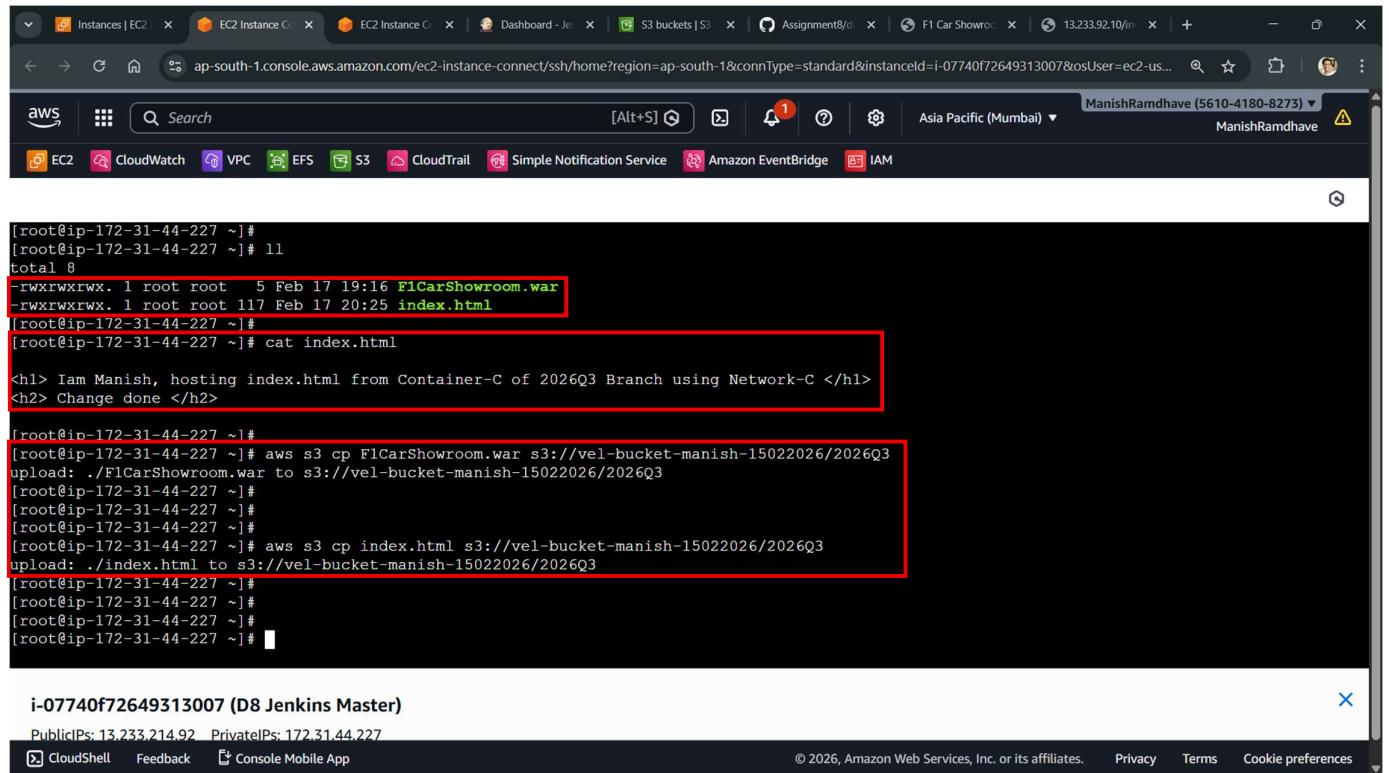
[root@ip-172-31-44-227 ~]# aws s3 cp F1CarShowroom.war s3://vel-bucket-manish-15022026/2026Q2
upload: ./F1CarShowroom.war to s3://vel-bucket-manish-15022026/2026Q2
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]# aws s3 cp index.html s3://vel-bucket-manish-15022026/2026Q2
upload: ./index.html to s3://vel-bucket-manish-15022026/2026Q2
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]#
```

The 'index.html' content is highlighted in red:

```
<h1> Iam Manish, hosting index.html from Container-B of 2026Q2 Branch using Network-B </h1>
<h2> Docker Assignment 8 Task2 Done </h2>
```

The AWS CloudShell interface is visible at the bottom, showing the instance ID (i-07740f72649313007), public and private IPs, and various navigation links.

Step 16: By using the AWS CLI commands, we have uploaded the ‘F1CarShowroom.war’ and ‘index.html’ files in the ‘vel-bucket-manish15022026/2026Q1’ S3 Bucket folder:



The screenshot shows a terminal session in AWS CloudShell. The user has uploaded the 'F1CarShowroom.war' and 'index.html' files to the 'vel-bucket-manish15022026/2026Q3' S3 bucket. The terminal output is as follows:

```
[root@ip-172-31-44-227 ~]# ls
[root@ip-172-31-44-227 ~]# ll
total 8
-rwxrwxrwx. 1 root root 5 Feb 17 19:16 F1CarShowroom.war
-rwxrwxrwx. 1 root root 117 Feb 17 20:25 index.html
[root@ip-172-31-44-227 ~]# cat index.html
<h1> Iam Manish, hosting index.html from Container-C of 2026Q3 Branch using Network-C </h1>
<h2> Change done </h2>

[root@ip-172-31-44-227 ~]# aws s3 cp F1CarShowroom.war s3://vel-bucket-manish-15022026/2026Q3
upload: ./F1CarShowroom.war to s3://vel-bucket-manish-15022026/2026Q3
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]# aws s3 cp index.html s3://vel-bucket-manish-15022026/2026Q3
upload: ./index.html to s3://vel-bucket-manish-15022026/2026Q3
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]#
[root@ip-172-31-44-227 ~]#
```

The 'index.html' content is highlighted in red:

```
<h1> Iam Manish, hosting index.html from Container-C of 2026Q3 Branch using Network-C </h1>
<h2> Change done </h2>
```

The AWS CloudShell interface is visible at the bottom, showing the instance ID (i-07740f72649313007), public and private IPs, and various navigation links.

Results:

- When Build is done in by ‘Doc-Assign8-Pipeline-A’, the ‘F1CarShowroom’ application and ‘index.html’ file is hosted from the container of ‘Network-A’ by following the ‘docker-compose.yaml’ script in the GitHub Repository of the 2026Q1 Branch by using Port No.80:

The screenshot displays two browser windows. The top window is a Jenkins job titled 'Doc-Assign8-Pipeline-A #57'. It shows a green checkmark icon, the build number '#57 (17 Feb 2026, 20:43:30)', and a status message 'Started by user mmm'. The bottom window shows the 'F1 Car Showroom' application at '13.233.92.10:8080/F1CarShowroom/'. The page features a red header with 'F1 Car Showroom' and 'Ultimate Formula 1 Experience', followed by a grid of Formula 1 cars.

- When Build is done in by ‘Doc-Assign8-Pipeline-B’, the ‘F1CarShowroom’ application and ‘index.html’ file is hosted from the container of ‘Network-B’ by following the ‘docker-compose.yaml’ script in the GitHub Repository of the 2026Q2 Branch by using Port No.80:

The screenshot displays two browser windows. The top window is a Jenkins job titled 'Doc-Assign8-Pipeline-B #5'. It shows a green checkmark icon, the build number '#5 (17 Feb 2026, 20:47:16)', and a status message 'Started by user mmm'. The bottom window shows the 'F1 Car Showroom' application at '13.233.92.10:8080/F1CarShowroom/'. The page features a red header with 'F1 Car Showroom' and 'Ultimate Formula 1 Experience', followed by a grid of Formula 1 cars.

3. When Build is done in by ‘Doc-Assign8-Pipeline-C’, the ‘F1CarShowroom’ application and ‘index.html’ file is hosted from the container of ‘Network-A’ by following the ‘docker-compose.yaml’ script in the GitHub Repository of the 2026Q3 Branch by using Port No.80:

