# A.M. Patel Institute of Computer Studies
# (Ganpat University)

## CERTIFICATE

**Enrolment No:**

**Exam seat No:**

This is to certify that the practical work satisfactorily carried and hence recorded in this journal, is the bonafide work of

**Mr./Mrs**……………………………………………………………………

of the……………………course in the sem…… of this department

during academic year……………. ..Subject code…………………..

Subject Name……………………………………………………………

No. of practical certified ………. out of ………….

**FACULTY SIGNATURE**

**HEAD OF THE DEPARTMENT**

Ganpat University | A.M. Patel Institute of Computer Studies

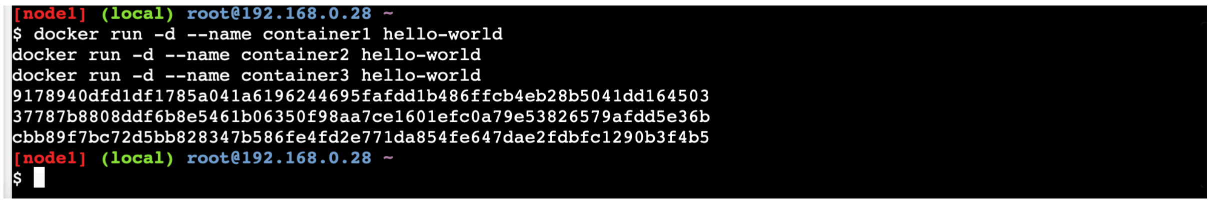
**DATE**

**REMARK OF EXAMINER**

**SEAL OF THE INSTITUTE**

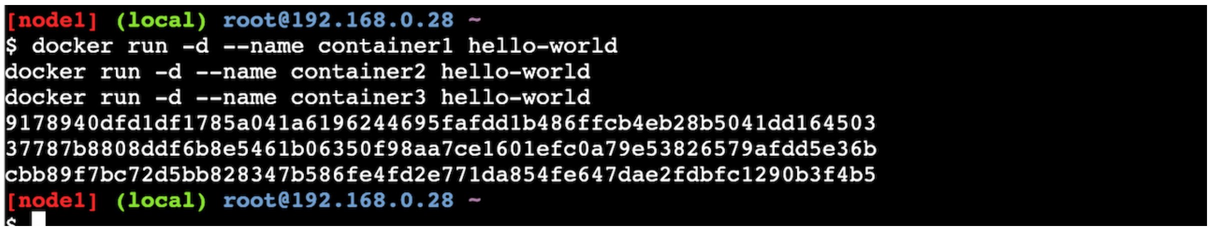
| No | Practical | date | sign |
|---|---|---|---|
| 1 | Pull images of hello-world-python,Create containers with them and check for  response using run docker command. | | |
| 2 | Update above program to create three containers in detached mode. | | |
| 3 | Demonstrate the docker inspect image_id command with example. | | |
| 4 | Demonstrate the docker stop and docker kill commands with example. | | |
| 5 | Demonstrate the docker container prune command | | |
| 6 | Demonstrate difference between docker container ls and docker container ls –a command. | | |
| 7 | Demonstrate the process of launching Jenkins as Docker Container with practical  steps. | | |
| 8 | Demonstrate how to build jobs in jenkins | | |
| 9 | Write a Jenkins program to demonstrate scripted pipeline in Jenkins. | | |

## 1. Pull images of hello-world-python,Create containers with them and check for response using run docker command.



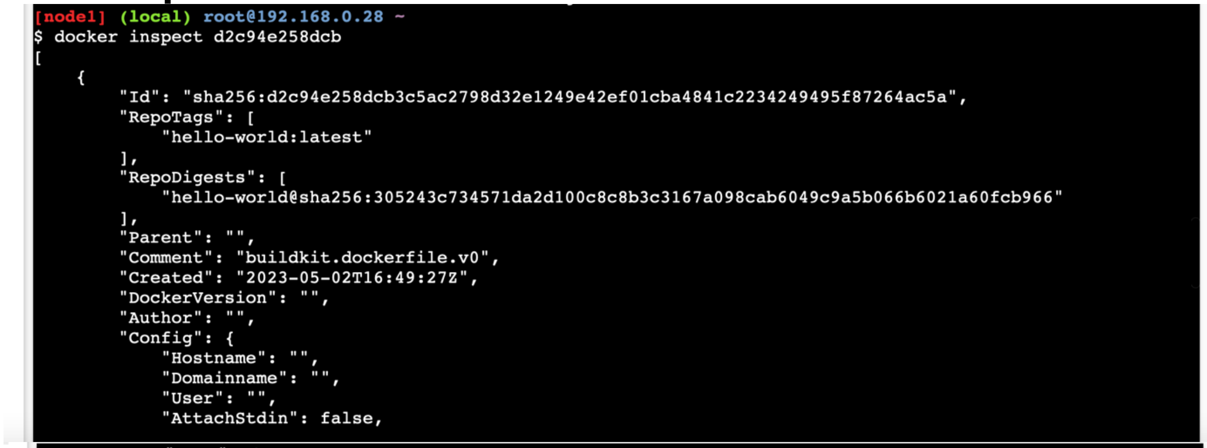
```
[node1] (local) root@192.168.0.28 ~
$ docker run -d --name container1 hello-world
docker run -d --name container2 hello-world
docker run -d --name container3 hello-world
9178940dfd1df1785a041a6196244695fafdd1b486ffcb4eb28b5041dd164503
37787b8808ddf6b8e5461b06350f98aa7ce1601efc0a79e53826579afdd5e36b
cbb89f7bc72d5bb828347b586fe4fd2e771da854fe647dae2fdbfc1290b3f4b5
[node1] (local) root@192.168.0.28 ~
$ 
```

## 2. Update above program to create three containers in detached mode.

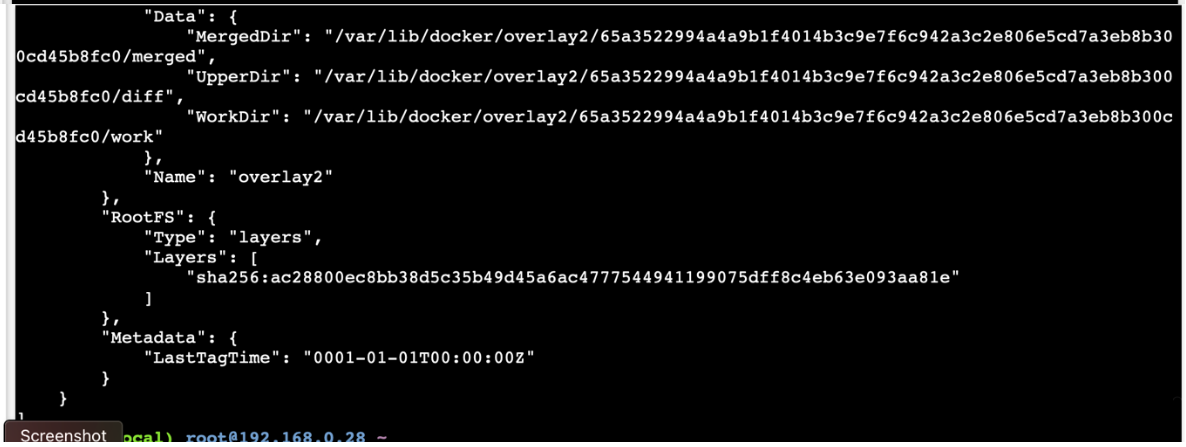


```
[node1] (local) root@192.168.0.28 ~
$ docker run -d --name container1 hello-world
docker run -d --name container2 hello-world
docker run -d --name container3 hello-world
9178940dfd1df1785a041a6196244695fafdd1b486ffcb4eb28b5041dd164503
37787b8808ddf6b8e5461b06350f98aa7ce1601efc0a79e53826579afdd5e36b
cbb89f7bc72d5bb828347b586fe4fd2e771da854fe647dae2fdbfc1290b3f4b5
[node1] (local) root@192.168.0.28 ~
$ 
```

## 3. Demonstrate the docker inspect image_id command with example.



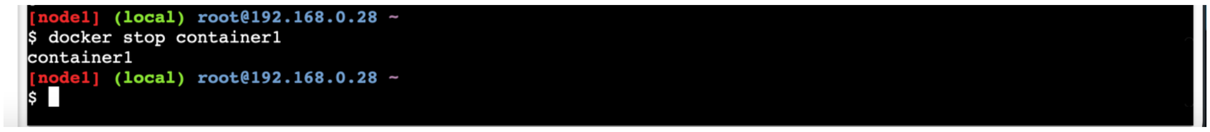
```
[node1] (local) root@192.168.0.28 ~
$ docker inspect d2c94e258dcb
[
    {
        "Id": "sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a",
        "RepoTags": [
            "hello-world:latest"
        ],
        "RepoDigests": [
            "hello-world@sha256:305243c734571da2d100c8c8b3c3167a098cab6049c9a5b066b6021a60fcb966"
        ],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2023-05-02T16:49:27Z",
        "DockerVersion": "",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
```



```
                "Data": {
                    "MergedDir": "/var/lib/docker/overlay2/65a3522994a4a9b1f4014b3c9e7f6c942a3c2e806e5cd7a3eb8b30
0cd45b8fc0/merged",
                    "UpperDir": "/var/lib/docker/overlay2/65a3522994a4a9b1f4014b3c9e7f6c942a3c2e806e5cd7a3eb8b300
cd45b8fc0/diff",
                    "WorkDir": "/var/lib/docker/overlay2/65a3522994a4a9b1f4014b3c9e7f6c942a3c2e806e5cd7a3eb8b300c
d45b8fc0/work"
                },
                "Name": "overlay2"
            },
            "RootFS": {
                "Type": "layers",
                "Layers": [
                    "sha256:ac28800ec8bb38d5c35b49d45a6ac4777544941199075dff8c4eb63e093aa81e"
                ]
            },
            "Metadata": {
                "LastTagTime": "0001-01-01T00:00:00Z"
            }
    }
]
```

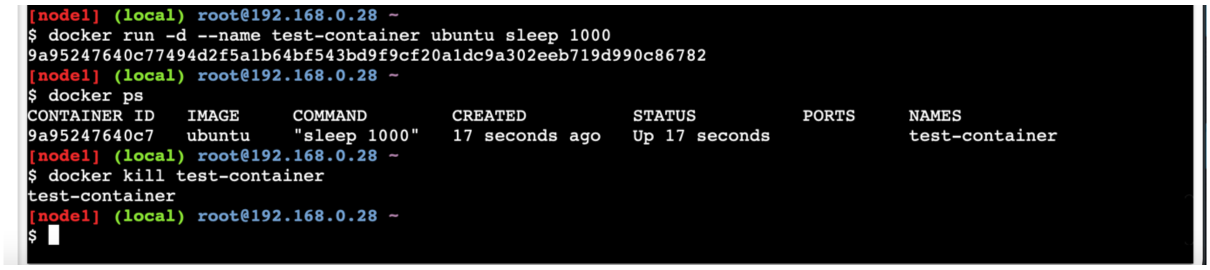
## 4. Demonstrate the docker stop and docker kill commands with example.

### Stop command-



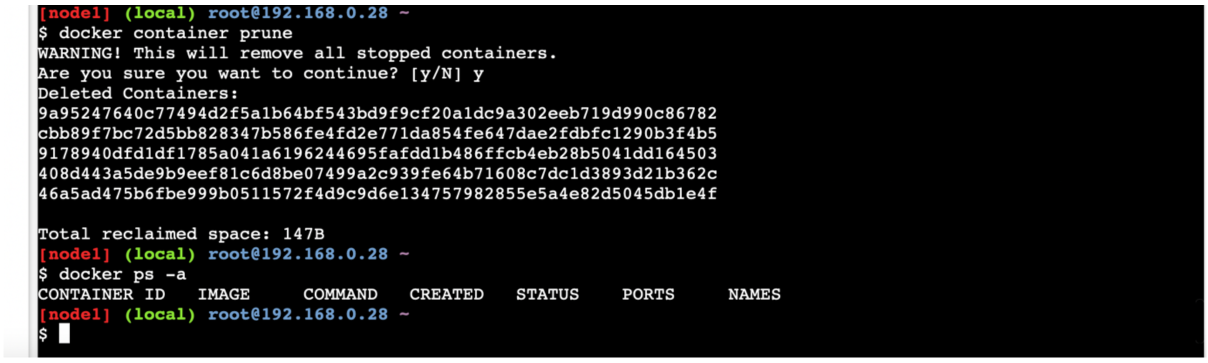
```
[node1] (local) root@192.168.0.28 ~
$ docker stop container1
container1
[node1] (local) root@192.168.0.28 ~
$ 
```

### Kill command



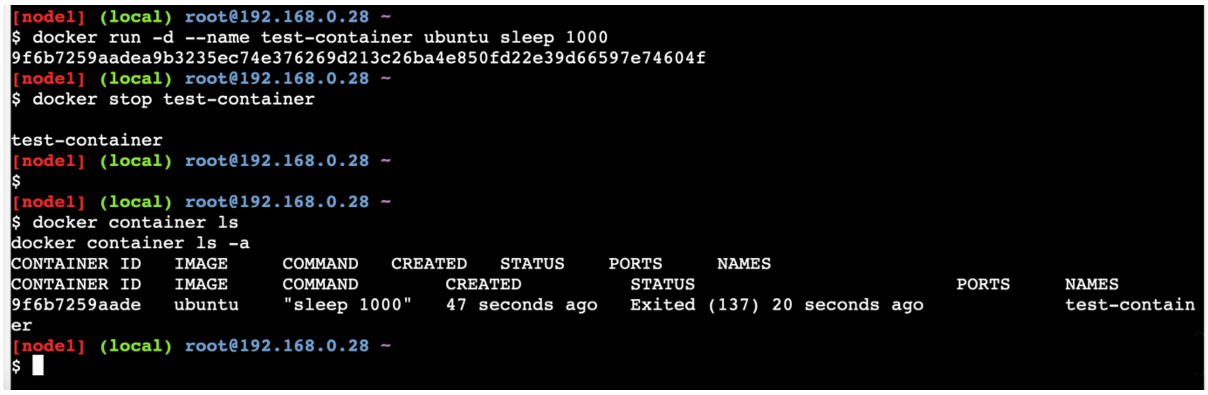
```
[node1] (local) root@192.168.0.28 ~
$ docker run -d --name test-container ubuntu sleep 1000
9a95247640c77494d2f5a1b64bf543bd9f9cf20a1dc9a302eeb719d990c86782
[node1] (local) root@192.168.0.28 ~
$ docker ps
CONTAINER ID   IMAGE     COMMAND         CREATED          STATUS          PORTS     NAMES
9a95247640c7   ubuntu    "sleep 1000"   17 seconds ago   Up 17 seconds             test-container
[node1] (local) root@192.168.0.28 ~
$ docker kill test-container
test-container
[node1] (local) root@192.168.0.28 ~
$ 
```

## 5. Demonstrate the docker container prune command.



```
[node1] (local) root@192.168.0.28 ~
$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
9a95247640c77494d2f5a1b64bf543bd9f9cf20a1dc9a302eeb719d990c86782
cbb89f7bc72d5bb828347b586fe4fd2e771da854fe647dae2fdbfc1290b3f4b5
9178940dfd1df1785a041a6196244695fafdd1b486ffcb4eb28b5041dd164503
408d443a5de9b9eef81c6d8be07499a2c939fe64b71608c7dc1d3893d21b362c
46a5ad475b6fbe999b0511572f4d9c9d6e134757982855e5a4e82d5045db1e4f

Total reclaimed space: 147B
[node1] (local) root@192.168.0.28 ~
$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[node1] (local) root@192.168.0.28 ~
$ 
```

## 6. Demonstrate difference between docker container ls and docker container ls –a command.



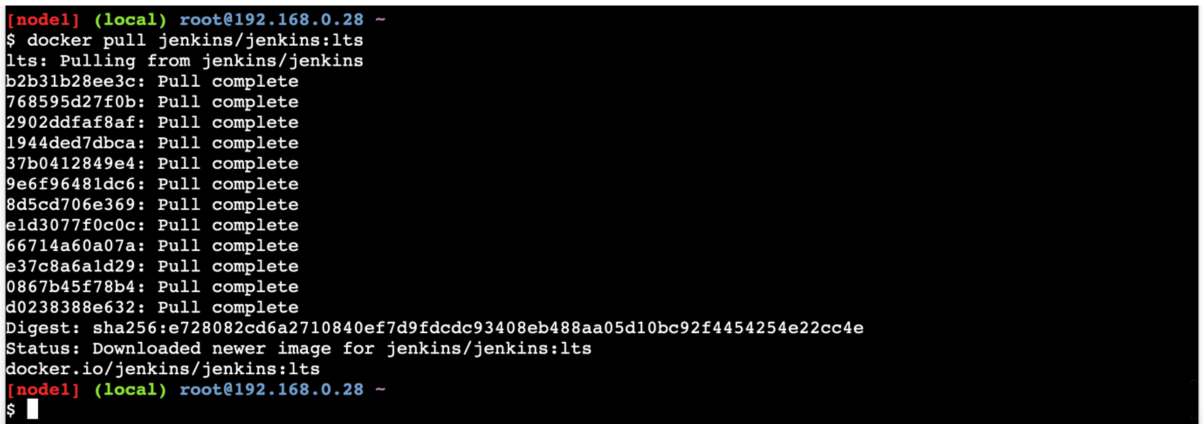
```
[node1] (local) root@192.168.0.28 ~
$ docker run -d --name test-container ubuntu sleep 1000
9f6b7259aadea9b3235ec74e376269d213c26ba4e850fd22e39d66597e74604f
[node1] (local) root@192.168.0.28 ~
$ docker stop test-container

test-container
[node1] (local) root@192.168.0.28 ~
$
[node1] (local) root@192.168.0.28 ~
$ docker container ls
docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
CONTAINER ID   IMAGE     COMMAND         CREATED          STATUS                       PORTS     NAMES
9f6b7259aade   ubuntu    "sleep 1000"    47 seconds ago   Exited (137) 20 seconds ago            test-contain
er
[node1] (local) root@192.168.0.28 ~
$ 
```

## 7. Demonstrate the process of launching Jenkins as Docker Container with practical steps.

**Step 1: Pull the Jenkins Image**

**Open Terminal and run the following command to pull the official Jenkins LTS image:**

**docker pull jenkins/jenkins:lts**



```
[node1] (local) root@192.168.0.28 ~
$ docker pull jenkins/jenkins:lts
lts: Pulling from jenkins/jenkins
b2b31b28ee3c: Pull complete
768595d27f0b: Pull complete
2902ddfaf8af: Pull complete
1944ded7dbca: Pull complete
37b0412849e4: Pull complete
9e6f96481dc6: Pull complete
8d5cd706e369: Pull complete
e1d3077f0c0c: Pull complete
66714a60a07a: Pull complete
e37c8a6a1d29: Pull complete
0867b45f78b4: Pull complete
d0238388e632: Pull complete
Digest: sha256:e728082cd6a2710840ef7d9fdcdc93408eb488aa05d10bc92f4454254e22cc4e
Status: Downloaded newer image for jenkins/jenkins:lts
docker.io/jenkins/jenkins:lts
[node1] (local) root@192.168.0.28 ~
$ 
```

**Step 2: Run Jenkins in Detached Mode**

**Run Jenkins as a container with the following command:**

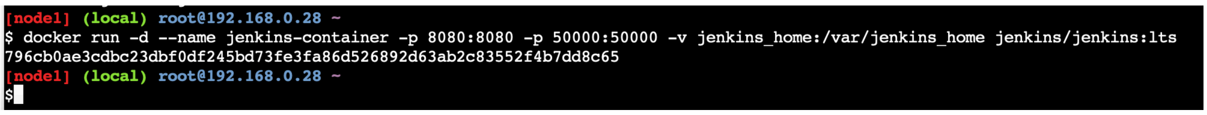
**docker run -d -p 8080:8080 -p 50000:50000 --name jenkins jenkins/jenkins:lts**

**-d: Run the container in detached mode.**

**-p 8080:8080: Expose port 8080 for Jenkins web interface.**

**-p 50000:50000: Expose port 50000 for Jenkins agents.**

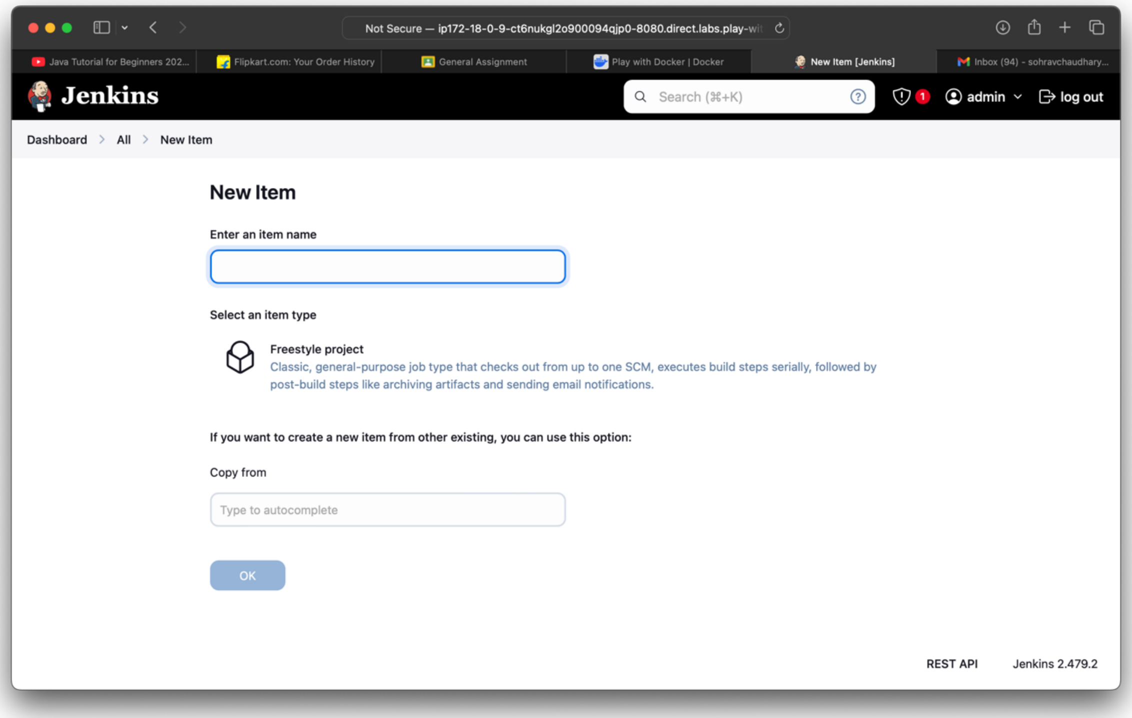
**--name jenkins: Assign the name "jenkins" to the container.**



```
[node1] (local) root@192.168.0.28 ~
$ docker run -d --name jenkins-container -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts
796cb0ae3cdbc23dbf0df245bd73fe3fa86d526892d63ab2c83552f4b7dd8c65
[node1] (local) root@192.168.0.28 ~
$
```

## 8. Demonstrate how to build jobs in Jenkins
## How to Build Jobs in Jenkins

### 1. Create a New Job:
- Open Jenkins and click on "New Item".
- Enter a name for the job (e.g., "MyFirstJob"), select "Freestyle project", and click "OK".


Not Secure — ip172-18-0-9-ct6nukgl2o900094qjp0-8080.direct.labs.play-wi

Jenkins

Dashboard > All > New Item

New Item

Enter an item name

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

If you want to create a new item from other existing, you can use this option:

Copy from

Type to autocomplete

OK

REST API Jenkins 2.479.2

### 2. Configure Source Code Management:
- In the "Source Code Management" section, select "Git".
- Enter your repository URL (e.g., "https://github.com/username/repo.git").
- If needed, add credentials for authentication.

### 3. Set Build Triggers:
- In the "Build Triggers" section, select a trigger like "Poll SCM" or "GitHub hook trigger".
- Set a schedule (e.g., "H/5 * * * *" for every 5 minutes) or configure a webhook for automatic builds.

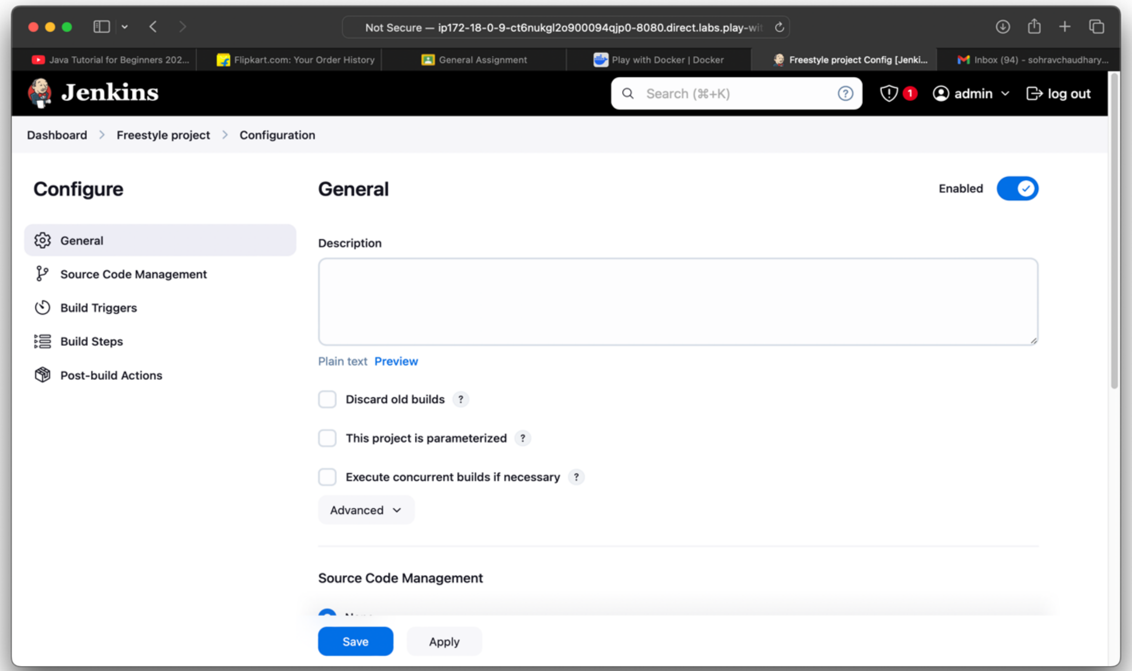
### 4. Define Build Steps:
- In the "Build" section, click "Add build step".

- Select "Execute Shell" (or the appropriate option for your environment).
- Add commands to build the project, such as:

```
git pull origin main
npm install
npm run build
```

5. Add Post-Build Actions:
   - In the "Post-build Actions" section, click "Add post-build action".
   - Select actions like "Archive the artifacts" (to save build files) or "Publish JUnit test results".
   - Configure the post-build action as needed (e.g., path to archived files).

6. Trigger the Build:
   - After configuring, click "Save".
   - Trigger the build manually by clicking "Build Now" on the job page.


Not Secure — ip172-18-0-9-ct6nukgl2o900094qjp0-8080.direct.labs.play-wi
Java Tutorial for Beginners 202...
Flipkart.com: Your Order History
General Assignment
Play with Docker | Docker
Freestyle project Config [Jenki...
Inbox (94) - sohravchaudhary...
Jenkins
Search (⌘+K)
admin
log out
Dashboard > Freestyle project > Configuration
Configure
General
Source Code Management
Build Triggers
Build Steps
Post-build Actions
Save
Apply

General

Enabled ✓

Description

Plain text Preview

☐ Discard old builds ?

☐ This project is parameterized ?

☐ Execute concurrent builds if necessary ?

Advanced

Source Code Management

**9. Write a Jenkins program to demonstrate scripted pipeline in Jenkins.**

**Jenkins Scripted Pipeline Example**

**A scripted pipeline in Jenkins is written using Groovy and allows flexibility in defining build steps.**

**Example Pipeline Code:**

```
node {
  stage('Checkout') {
    checkout scm  // Checkout code from the repository
  }

  stage('Build') {
    sh 'npm install'  // Install dependencies
    sh 'npm run build'  // Build the application
  }

  stage('Test') {
    sh 'npm test'  // Run tests
  }

  stage('Deploy') {
    sh 'npm run deploy'  // Deploy the application
  }
}
```

**Explanation:**

**1. node: Specifies that the pipeline will run on a Jenkins agent.**

**2. Checkout: Pulls the code from the source repository.**

**3. Build: Installs dependencies and builds the application.**
**4. Test: Runs tests for the application.**
**5. Deploy: Deploys the built application.**

**This is a basic structure for a Jenkins scripted pipeline.**