1 Lab Department of Information Technology oratory Manual for

**I.T./ T.E./ Sem V/**

**ITL503: DevOps Lab Journal**

**By**

**Ms. Sunantha Krishnan**

Don Bosco Institute of Technology, Mumbai 400070.

(Affiliated to the University of Mumbai)

**Name: Hemesh chopade**

**Class: TE IT**

**Roll No. 11**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.n** | **Name of the experiment** | **Date** | **Page No.** |
| A | Self study | 12/07/2024 | 2-5 |
| 1. | Git and Github Versioning control | 24/07/2024 | 6-11 |
| 2. | Jenkins Installation | 07/08/2024 | 12-18 |
| 3. | Jenkins Pipeline | 28/08/2024 | 19-32 |
| 4. | Docker | 04/09/2024 | 33-41 |
| **5.** | Mini Project- chef | 27/10/2024 | 42-51 |

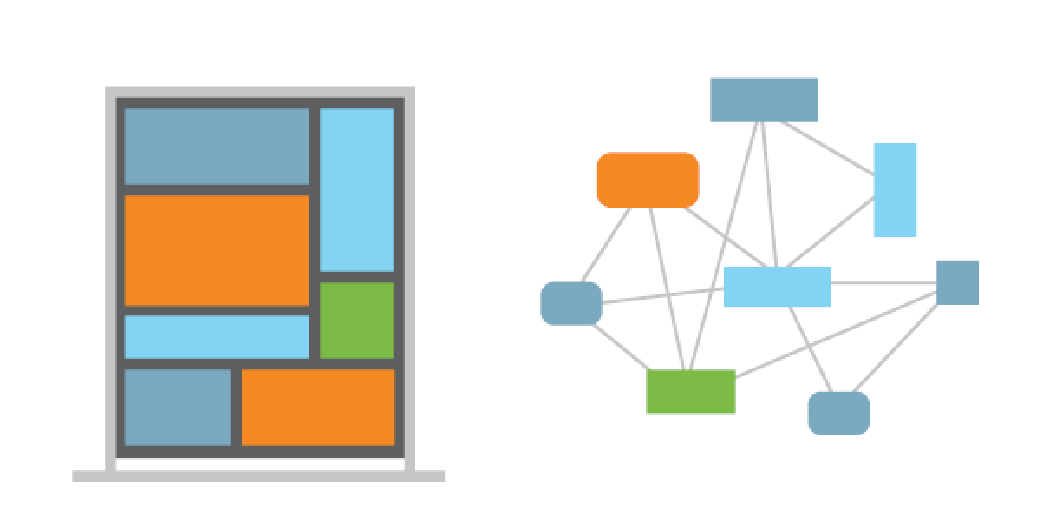
**Experiment A: Self Study**

**Prerequisite :** Microservice , Web Applications, App. Development **Aim : i**s to understand the need and requirement of DevOps in the Industries.

**Question :**

* **What do you mean by Microservice?**

Microservices describes the architectural process of building a distributed application from separately deployable services that perform specific business functions and communicate over web interfaces.



* **What's meant by a lightweight server?**

Lightweight servers are designed to be lean and efficient, consuming fewer resources than their traditional counterparts. The philosophy behind these servers is to provide only the essential functionality necessary for specific tasks, eliminating any superfluous elements that may consume unnecessary resources.

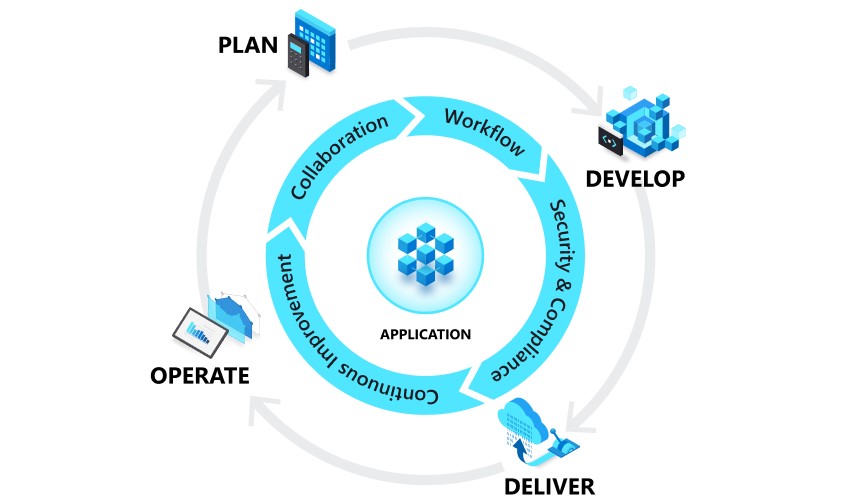
* **What is meant by CI/CD?**

CI And CD is the practice of automating the integration of code changes from multiple developers into a single codebase. It is a software development practice where the developers commit their work frequently to the central code repository (Github or Stash).

* **What is DevOps?**

DevOps is a software development process that combines development (Dev) and operations (Ops) to unite people, process, and technology in application planning, development, delivery, and operations.

DevOps enables coordination and collaboration.



* **What are the stages of Devops ?** the key phases of the [DevOps lifecycle:](https://www.atlassian.com/devops)

○ Discover

○ Plan

○ Build

○ Test

○ Monitor

○ Operate

○ Continuous feedback

○

* **What are the various tools that are used at every stage ?**

○ Discover



○ Plan



○ Build

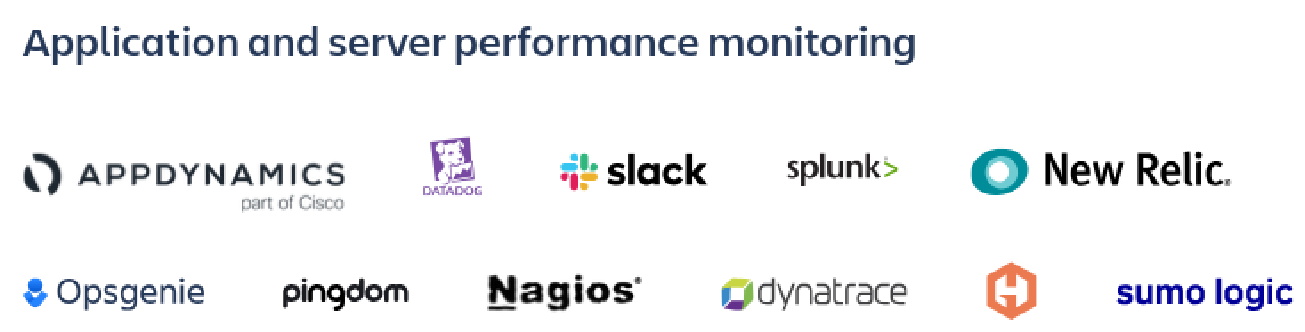




○ Test



○ Monitor



○ Operate



○ Continuous feedback



* **What do you mean by versioning?**

The sole purpose of version control in DevOps is **to keep a tab on the history of all the code changes made over the course of the project life cycle**. This helps developers get a clear idea of who made what changes and when as well as restore any previous version of a codebase if needed.

**Rubrics**: For Evaluation

|  |  |  |
| --- | --- | --- |
| With the hour & excellent | Within the next week | Within the semester |
| A++ to A | A maximum | B+ |
| A++ = 10  A+=9  A=8 | A=8  B++ = 7  B+= 6 | B= 5 |

**Conclusion :** All the given questions were learned and successfully documented .

**References:**

* 1. [**https://learn.microsoft.com/en-us/devops/develop/git/what-is-version-control**](https://learn.microsoft.com/en-us/devops/develop/git/what-is-version-control)
  2. [**https://www.geeksforgeeks.org/what-is-ci-cd/**](https://www.geeksforgeeks.org/what-is-ci-cd/)
  3. [**https://www.google.com/search?client=ubuntu&channel=fs&q=various+tools+that+ are+used+at+every+stage+od+devops**](https://www.google.com/search?client=ubuntu&channel=fs&q=various+tools+that+are+used+at+every+stage+od+devops)

EXPERIMENT 1 & 2

Git installation and versioning

Aim: To install git (local repository) and synchronize with github (remote repository) and perform version controlling.

Steps for installation and version control: git config

Usage: git config –global user.name “[name]”

Usage: git config –global user.email “[email address]”

This command sets the author name and email address respectively to be used with your commits.

git init

Usage: git init [repository name]

This command is used to start a new repository.

git clone

Usage: git clone [url]

This command is used to obtain a repository from an existing URL.

git add

Usage: git add [file]

This command adds a file to the staging area.

Usage: git add \*

This command adds one or more to the staging area.

git commit

Usage: git commit -m “[ Type in the commit message]”

This command records or snapshots the file permanently in the version history.

Usage: git commit -a

This command commits any files you’ve added with the git add command and also commits any files you’ve changed since then.

git diff

Usage: git diff

This command shows the file differences which are not yet staged.

Usage: git diff –staged

This command shows the differences between the files in the staging area and the latest version is present.

Usage: git diff [first branch] [second branch]

This command shows the differences between the two branches mentioned.

git reset

Usage: git reset [file]

This command unstages the file, but it preserves the file contents.

Usage: git reset [commit]

This command undoes all the commits after the specified commit and preserves the changes locally.

Usage: git reset –hard [commit] This command discards all history and goes back to the specified commit.

git status

Usage: git status

This command lists all the files that have to be committed.

git rm

Usage: git rm [file]

This command deletes the file from your working directory and stages the deletion.

git log

Usage: git log

This command is used to list the version history for the current branch.

Usage: git log –follow[file]

This command lists version history for a file, including the renaming of files also.

git show

Usage: git show [commit]

This command shows the metadata and content changes of the specified commit.

git tag

Usage: git tag [commitID]

This command is used to give tags to the specific commit.

git branch

Usage: git branch

This command lists all the local branches in the current repository.

Usage: git branch [branch name]

This command creates a new branch.

Usage: git branch -d [branch name] This command deletes the feature branch.

git checkout

Usage: git checkout [branch name]

This command is used to switch from one branch to another.

Usage: git checkout -b [branch name]

This command creates a new branch and also switches to it.

git merge

Usage: git merge [branch name]

This command merges the specified branch’s history into the current branch.

git remote

Usage: git remote add [variable name] [Remote Server Link] This command is used to connect your local repository to the remote server.

git push

Usage: git push [variable name] master

This command sends the committed changes of master branch to your remote repository.

Usage: git push [variable name] [branch]

This command sends the branch commits to your remote repository.

Usage: git push –all [variable name]

This command pushes all branches to your remote repository.

Usage: git push [variable name] :[branch name]

This command deletes a branch on your remote repository.

git pull

Usage: git pull [Repository Link]

This command fetches and merges changes on the remote server to your working directory.

git stash

Usage: git stash save

This command temporarily stores all the modified tracked files.

Usage: git stash pop

This command restores the most recently stashed files.

Usage: git stash list

This command lists all stashed changesets.

Usage: git stash drop

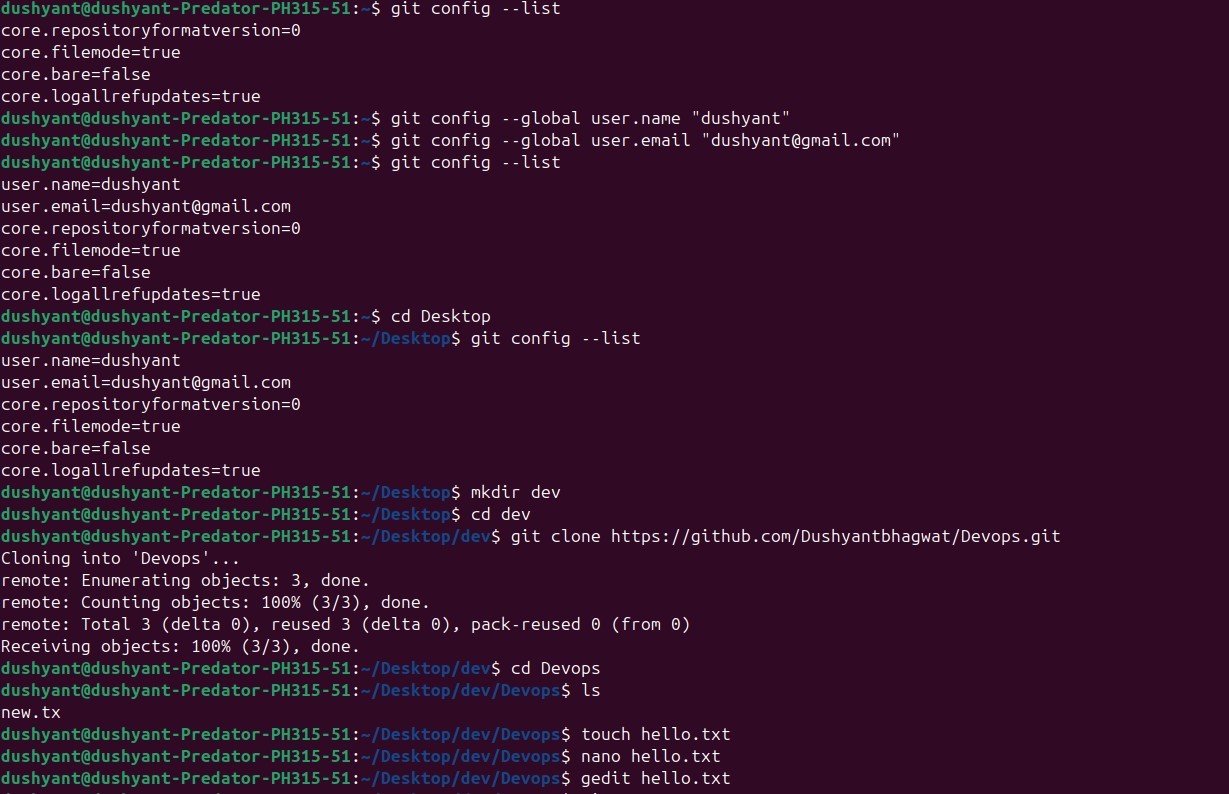
This command discards the most recently stash009564ed changeset.

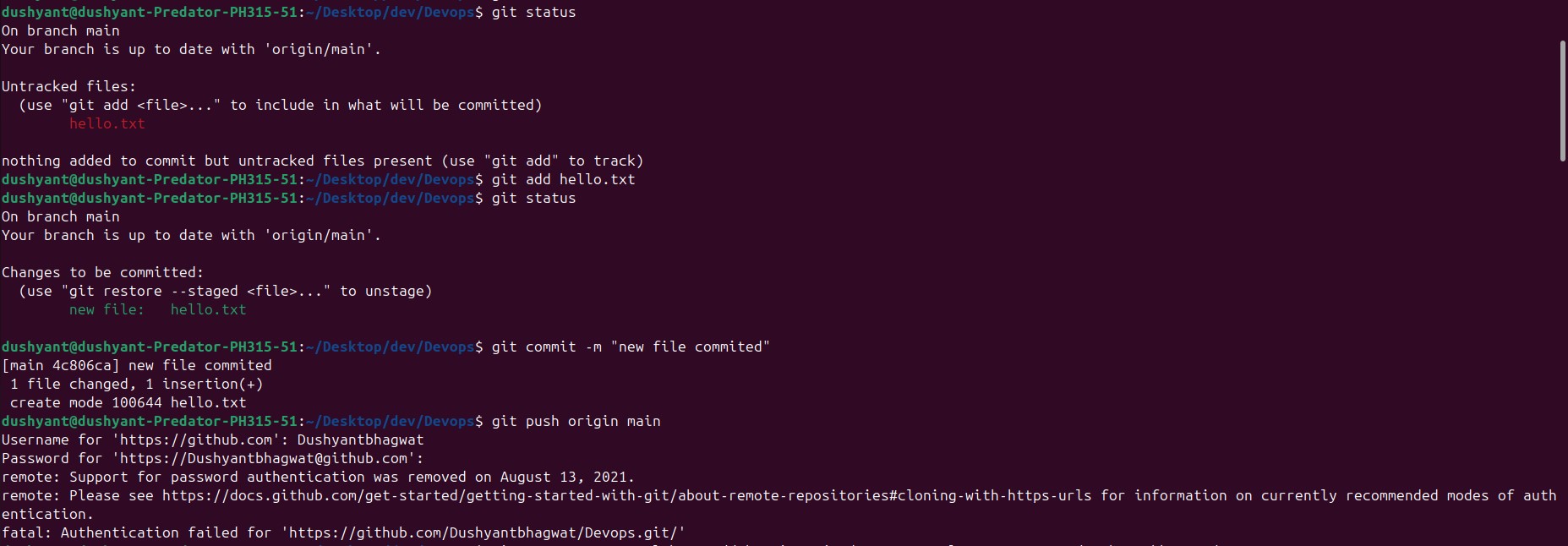
Get Token

1. Log into GitHub.
2. Click on your name / Avatar in the upper right corner and select Settings.
3. On the left, click Developer settings.
4. Select Personal access tokens and click Generate new token.
5. Give the token a description/name and select the scope of the token. ...
6. Click Generate token.
7. This configures the computer to remember the complex token by enable caching of the credentials.

git config --global credential.helper cache

1. If needed, you can later clear the token from the local computer by running









**Experiment 3: Jenkins Installation and Setup For CICD**

**Uninstall any version of java**

$java\_version=`java -version 2>&1 | head -n 1 | awk -F"\"" '{print $2}'`

-Remove all the Java related packages (Sun, Oracle, OpenJDK, IcedTea plugins, GIJ): $ sudo apt-get update

$ apt-cache search java | awk '{print($1)}' | grep -E -e '^(ia32-)?(sun|oracle)-java' -e '^openjdk-' -e

'^icedtea' -e '^(default|gcj)-j(re|dk)' -e '^gcj-(.\*)-j(re|dk)' -e 'java-common' | xargs sudo apt-get -y remove

$ sudo apt-get -y autoremove

-Purge config files:

$ dpkg -l | grep ^rc | awk '{print($2)}' | xargs sudo apt-get -y purge

-Remove Java config and cache directory:

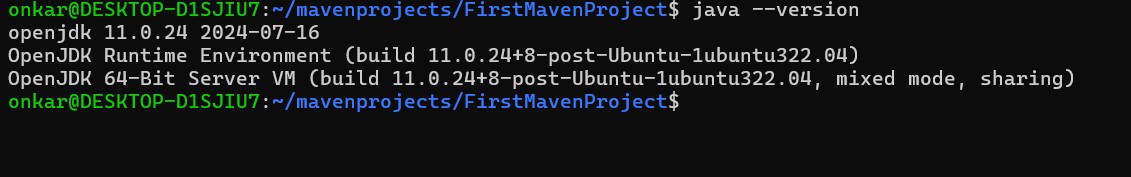
$ sudo bash -c 'ls -d /home/\*/.java' | xargs sudo rm -rf

-Remove manually installed JVMs: $ sudo rm -rf /usr/lib/jvm/\*

**Intall java : Jenkins requires Java to run. Install the OpenJDK package by running:**

sudo apt-get install openjdk-11-jdk or

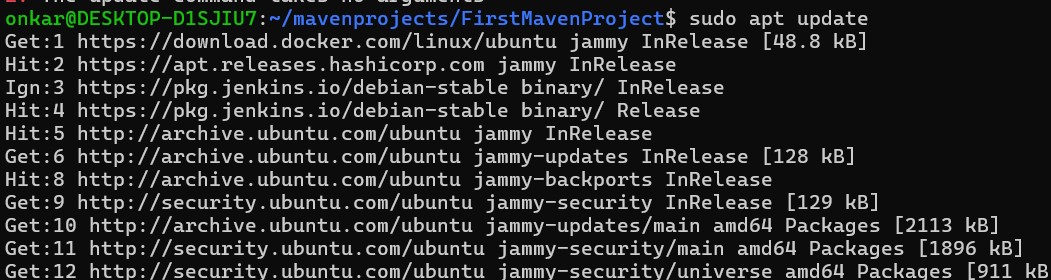
sudo apt install openjdk-11-jdk -y java -version



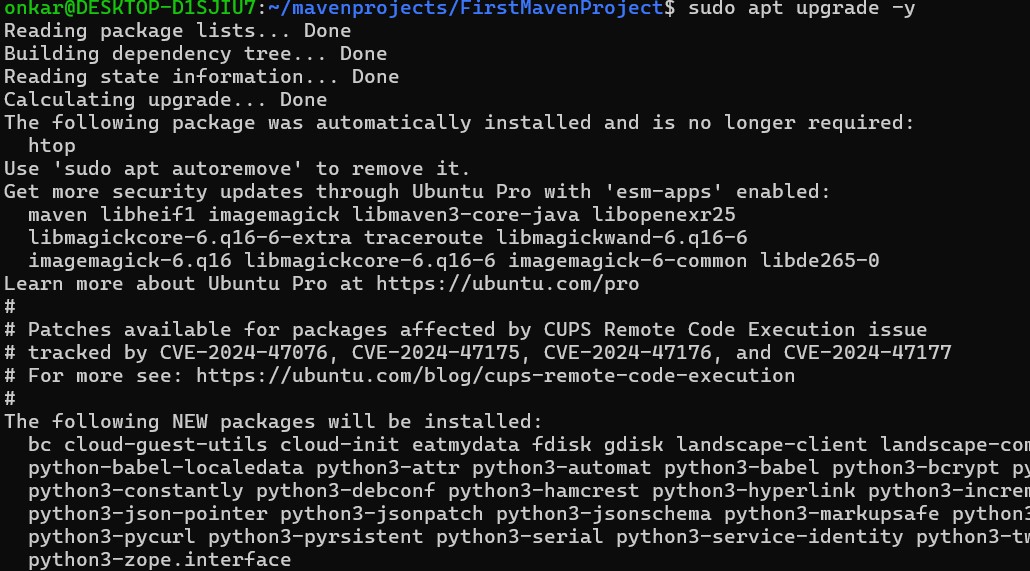
**Install Jenkins**

1. Before installing Jenkins, ensure your system package list is updated:

sudo apt update



sudo apt upgrade -y



1. As a prerequisite add the Jenkins repository to your system with:

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add –



1. Then, append the Jenkins repository to your system's sources list:

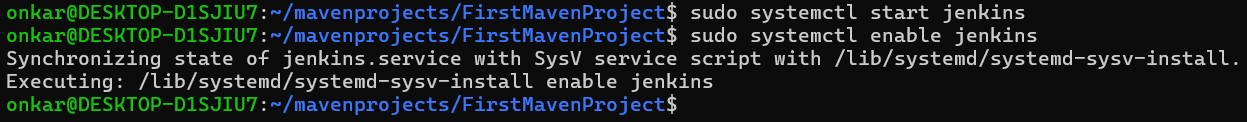
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

1. After adding the repository, install Jenkins:

sudo apt update sudo apt install jenkins -y

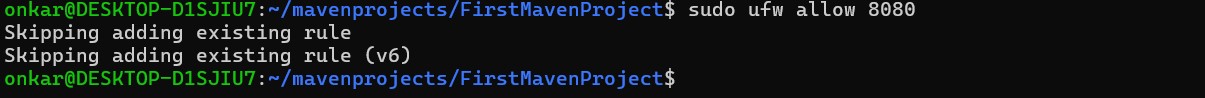
1. To start Jenkins and enable it to run at boot, use:

sudo systemctl start jenkins sudo systemctl enable Jenkins



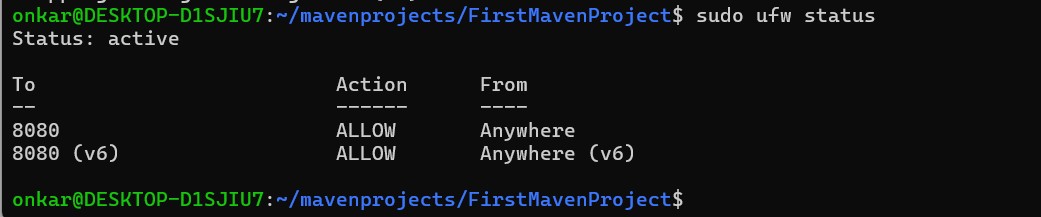
1. Adjust Firewall settings : If you have a firewall enabled, allow traffic on port 8080:

sudo ufw allow 8080



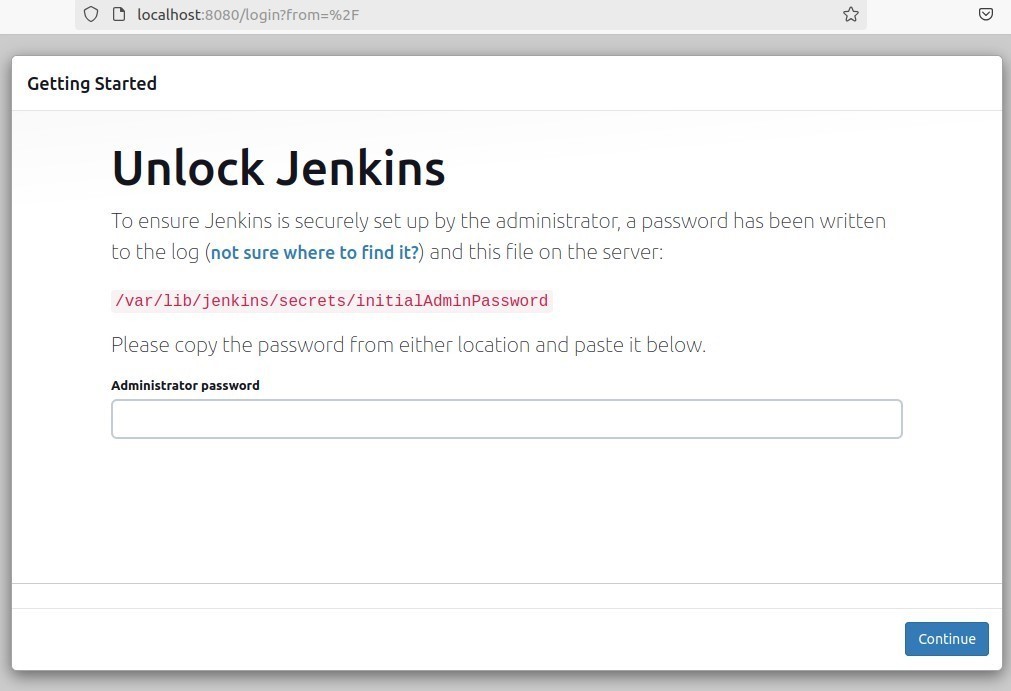
1. Check UFW status to confirm the change:

sudo ufw status



1. Configure Jenkins

To access Jenkins, navigate to http://localhost:8080 or http://localhost:8080 in your web browser. You'll be prompted to enter the Administrator password, which can be retrieved from:

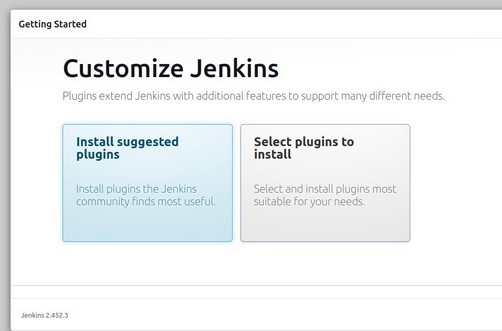


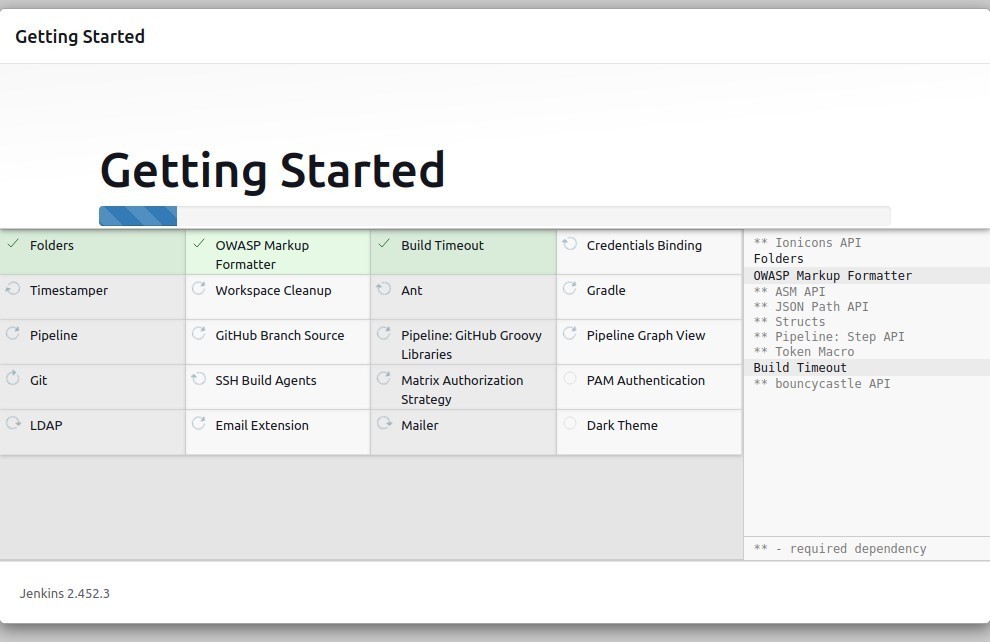
Get the password or the key to acess Jenkins using the path suggested along with sudo cat command

$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

1. Initial Setup Wizard

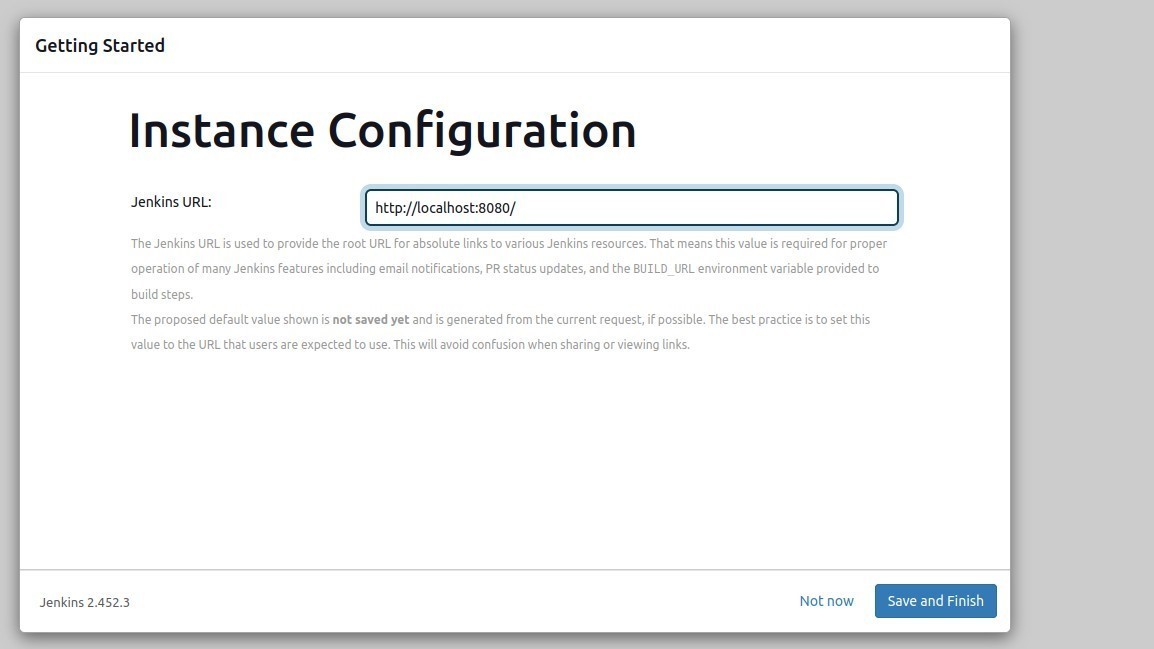
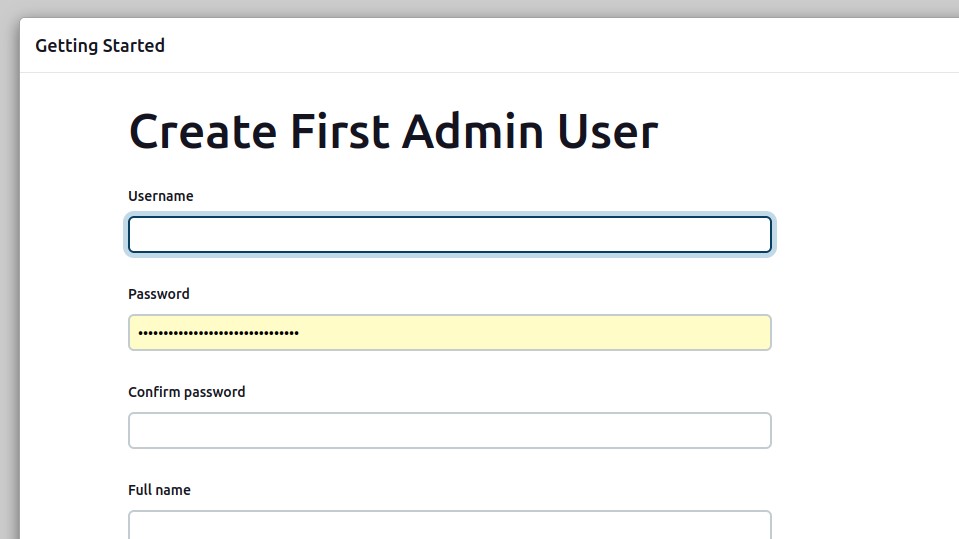
Upon entering the Administrator password, you'll be greeted by the Initial Setup Wizard. Here, you can install the suggested plugins or select specific ones according to your needs.





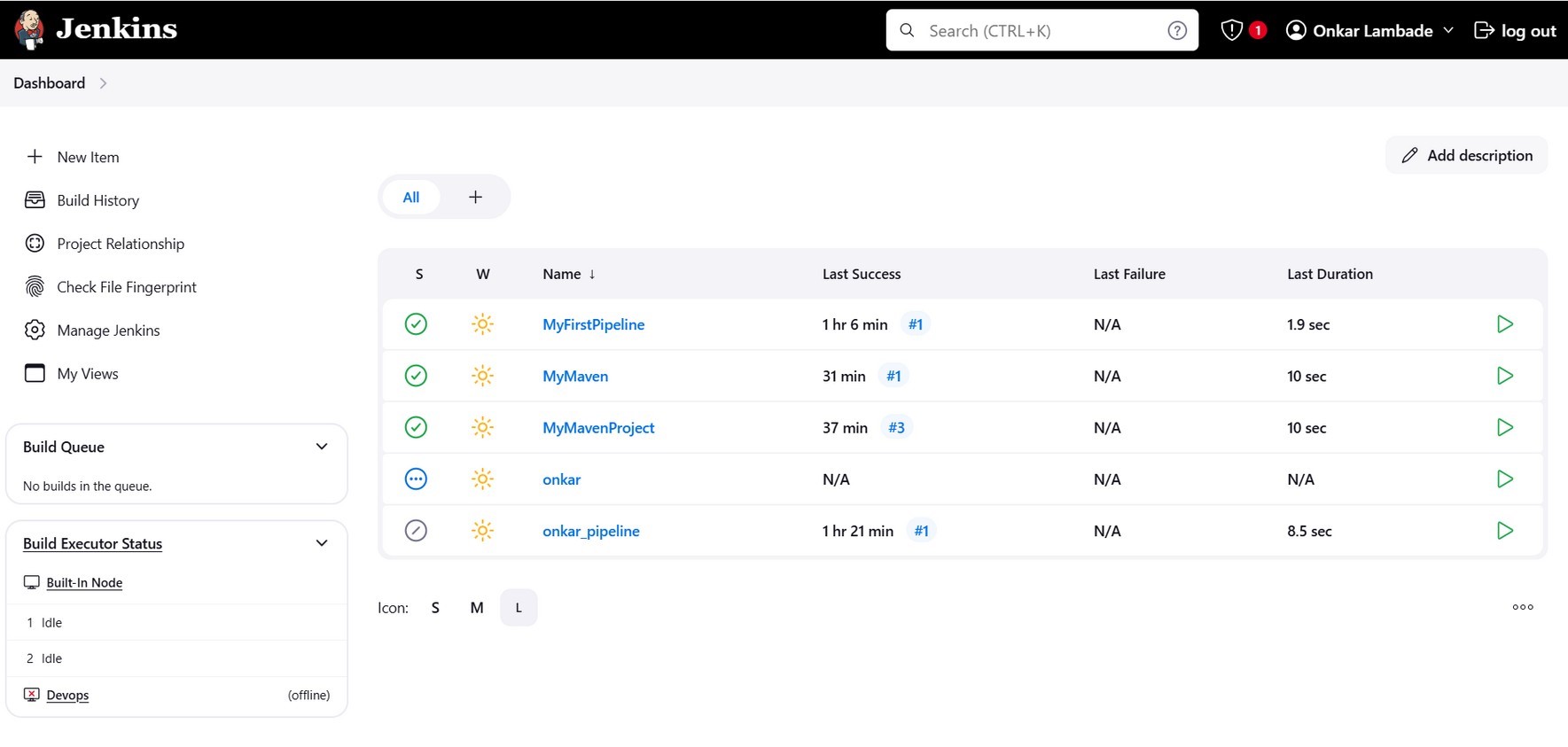
1. Create Admin User

After plugin installation, create an admin user with a username, password, and relevant details. ( always give user name as dbit, and password dbit and email as dbit@one.com)



1. Instance configuration

Finally, confirm the Jenkins URL and complete the setup. You're now ready to start creating your CI/CD pipelines!



**Conlusion :**

With Jenkins installed on your Ubuntu 22 system, you've taken a significant leap in automating your development processes.

**References :**

1. https://reintech.io/blog/installing-configuring-jenkins-ubuntu-22

**Experiment 4: Jenkin Pipeline and Marven Aim**: is to create pipeline adn maven project using jenkins

**Procedure :**

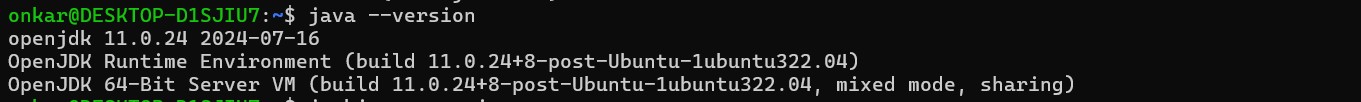
Steps to write here are

# Part A - Snapshots of your project creation and execution with output generated for pipeline

1. **Jenkins installed and running** on your local machine or server. If not, you can follow the official Jenkins installation guide.



2.**Java installed** on your machine.



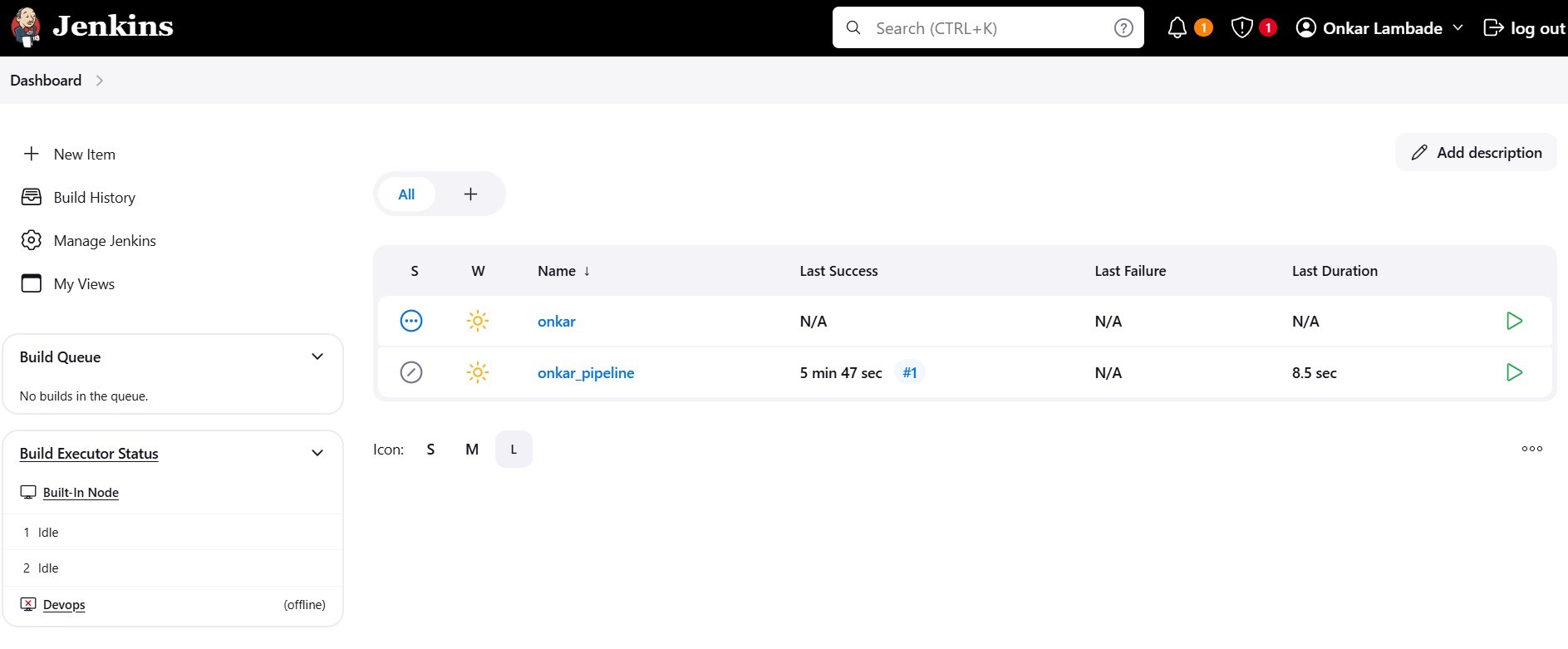
**Step-by-Step Guide for Creating and Executing a Pipeline in Jenkins**

**Step 1: Log in to Jenkins**

* Open a web browser and go to your Jenkins instance URL (typically [http://localhost:8080](http://localhost:8080/)).

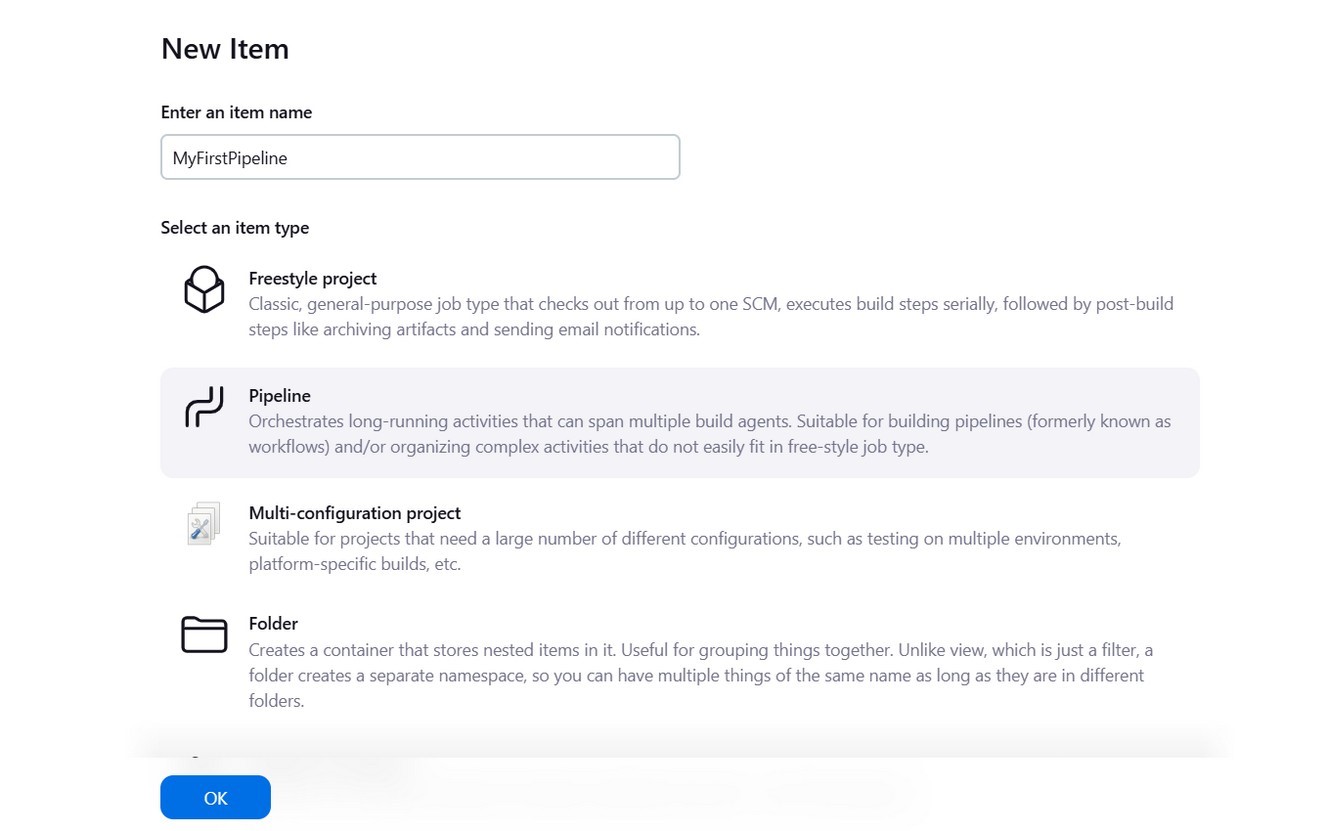


* Log in using your Jenkins credentials.



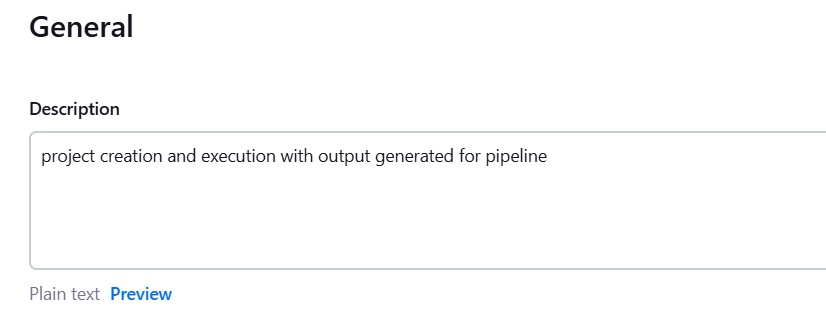
**Step 2: Create a New Pipeline Project**

1. From the Jenkins dashboard, click **“New Item”** on the left-hand menu.
2. Give your project a name, for example, MyFirstPipeline.
3. Select **“Pipeline”** as the project type.
4. Click **"OK"** to proceed.

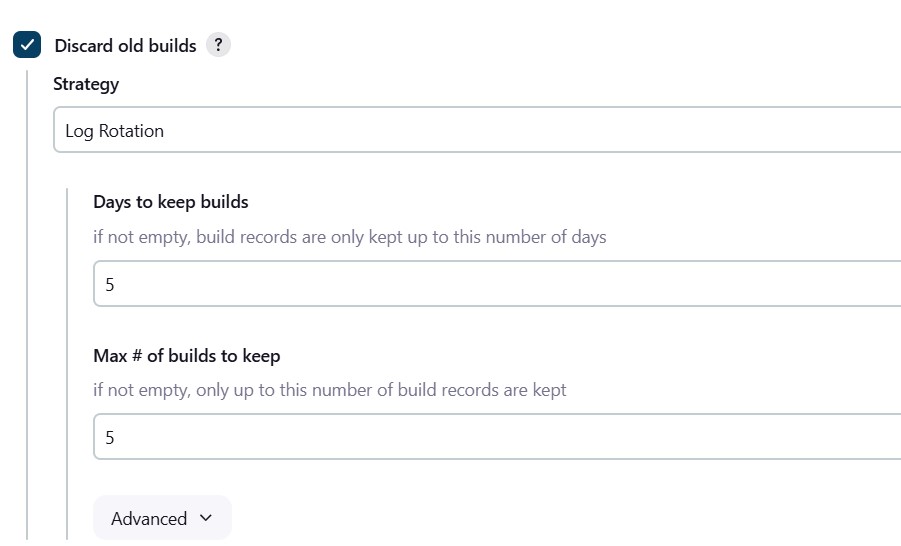


**Step 3: Configure the Pipeline**

1. On the project configuration page, you can fill in the following fields:
   * + **Description**: Describe what the pipeline does (optional).



* + - **Discard old builds**: You can check this option to limit the number of builds to keep.



1. **Scroll down to the "Pipeline" section**:
   * **Definition**: Choose **“Pipeline script”** from the dropdown menu.



* + **Script**: In this field, you will define your pipeline script.

pipeline { agent any

stages { stage('Checkout') {

steps {

echo 'Checking out code from SCM...'

} } stage('Build') { steps {

echo 'Building the project...'

} } stage('Test') { steps {

echo 'Running tests...'

}

}

stage('Package') {

steps {

echo 'Packaging the application...'

}

}

stage('Deploy') {

steps {

echo 'Deploying the application...'

}

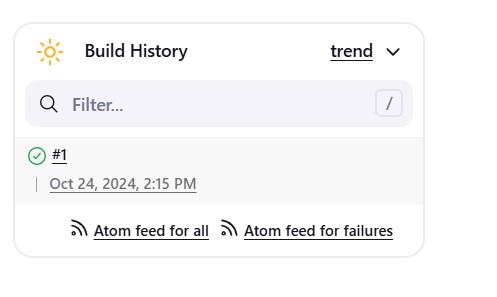
}

}

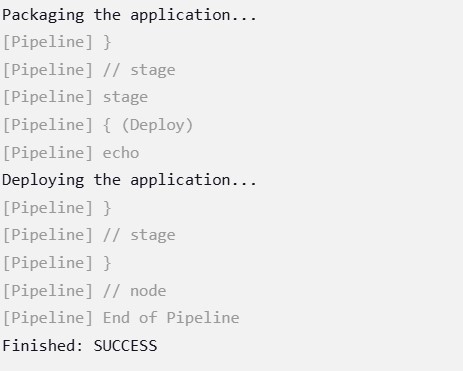
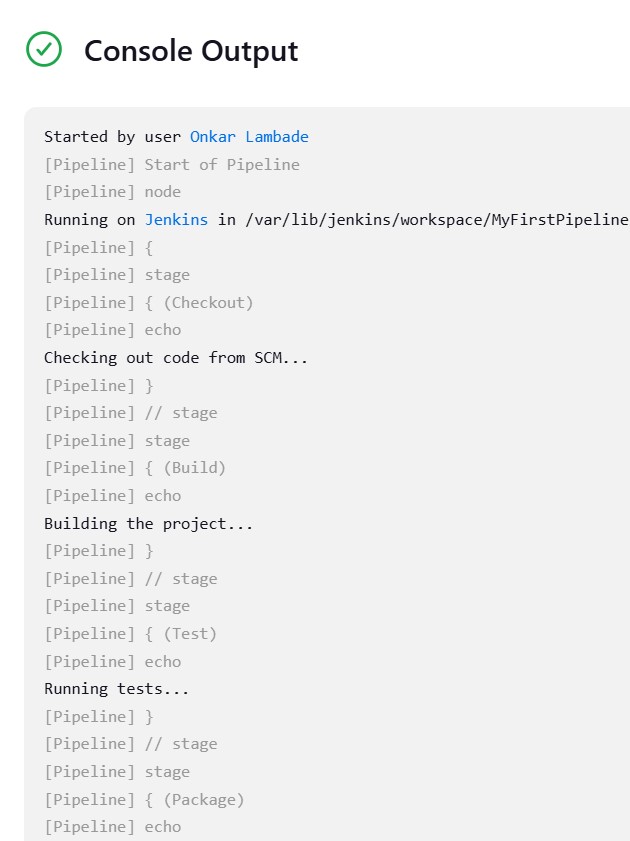
}

**Step 5: Save and Build the Pipeline**

1. **Click on “Save”** at the bottom of the configuration page.
2. On the project’s main page, you should see a **“Build Now”** option on the left sidebar. Click on it to trigger the pipeline. **Step 6: Check the Output**
3. Once the build is triggered, you can click on the build number under the **“Build History”** section to see the build details.



1. Click **“Console Output”** to see the real-time logs of your pipeline execution. You should see the output messages:



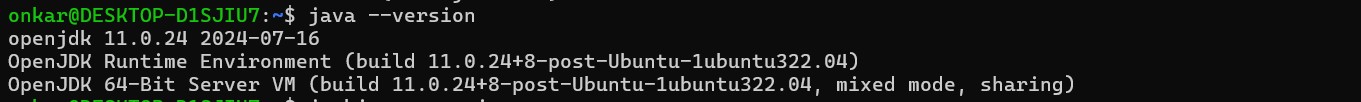
1. If you see this output, congratulations! Your pipeline is working.

# Part B -Snapshots of your project creation and execution with output generated for marven

**1.Jenkins installed and running** on your local machine or server. If not, you can follow the official Jenkins installation guide.



2.**Java installed** on your machine.



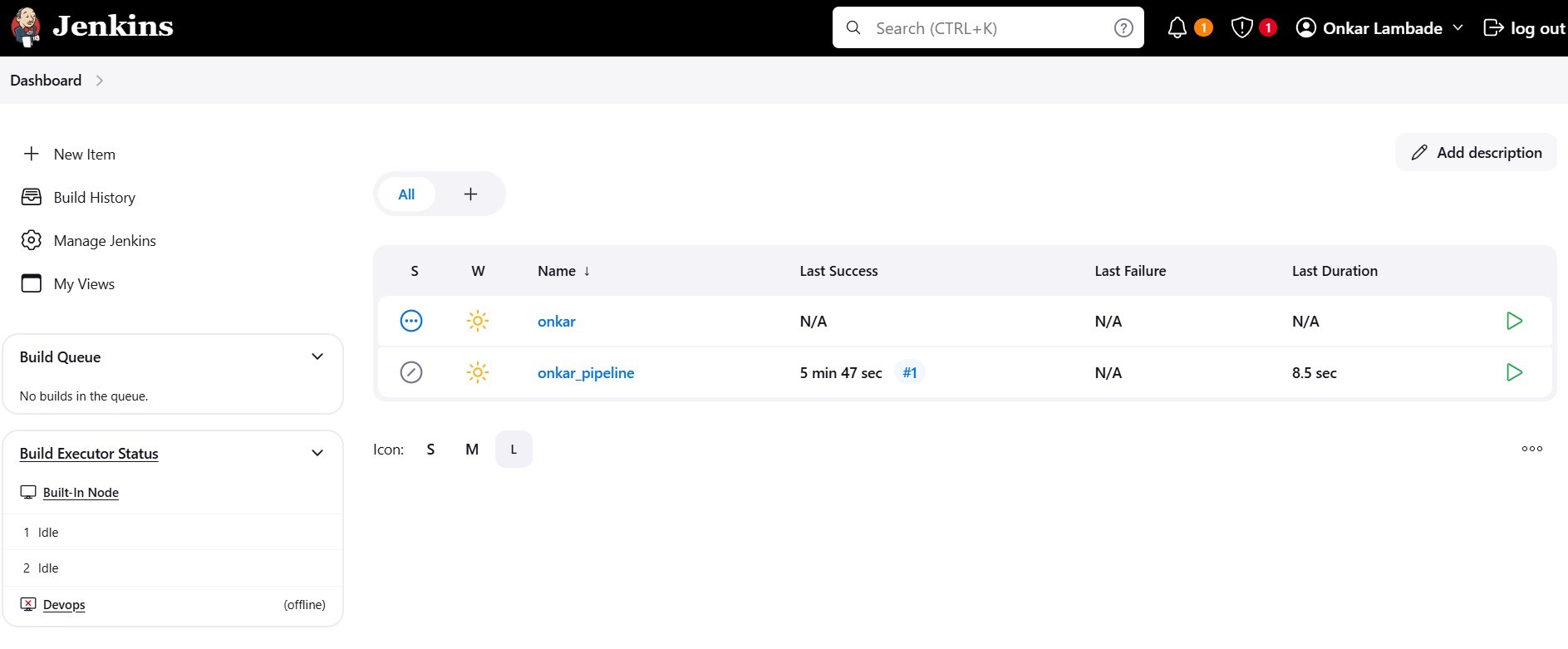
**Step-by-Step Guide for Creating and Executing a Pipeline in Jenkins:-**

**Step 1: Log in to Jenkins**

* Open a web browser and go to your Jenkins instance URL (typically [http://localhost:8080](http://localhost:8080/)).

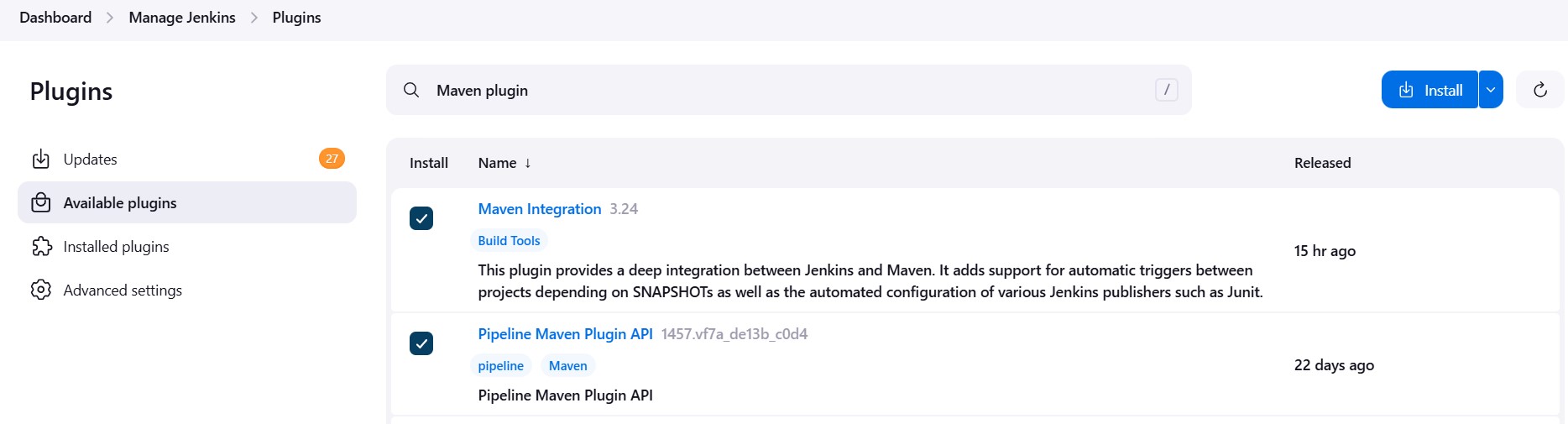


* Log in using your Jenkins credentials.



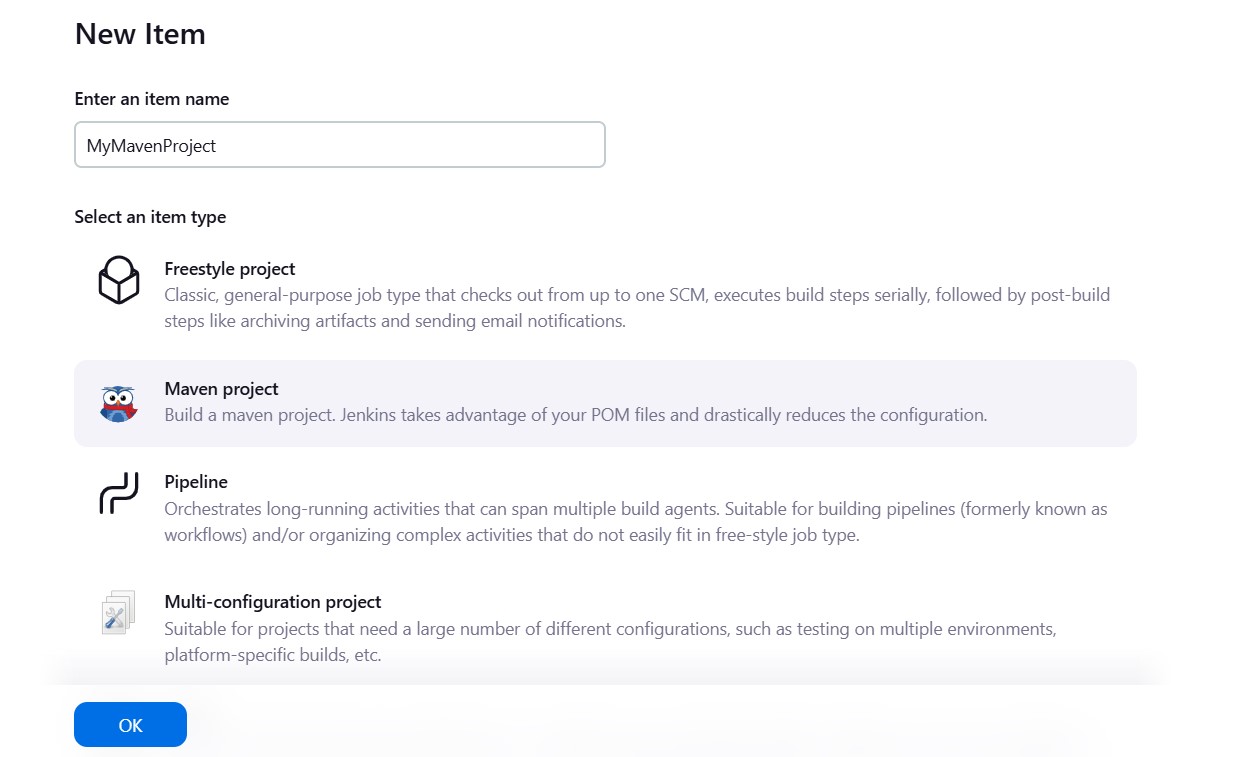
## Step 2: Install the Maven Plugin (if not already installed)

1. Go to **"Manage Jenkins"** from the dashboard.
2. Click on **"Manage Plugins"**.
3. In the **"Available"** tab, search for **"Maven Integration"** or **"Maven Plugin"**.
4. Install the plugin if it's not already installed. Restart Jenkins if prompted.



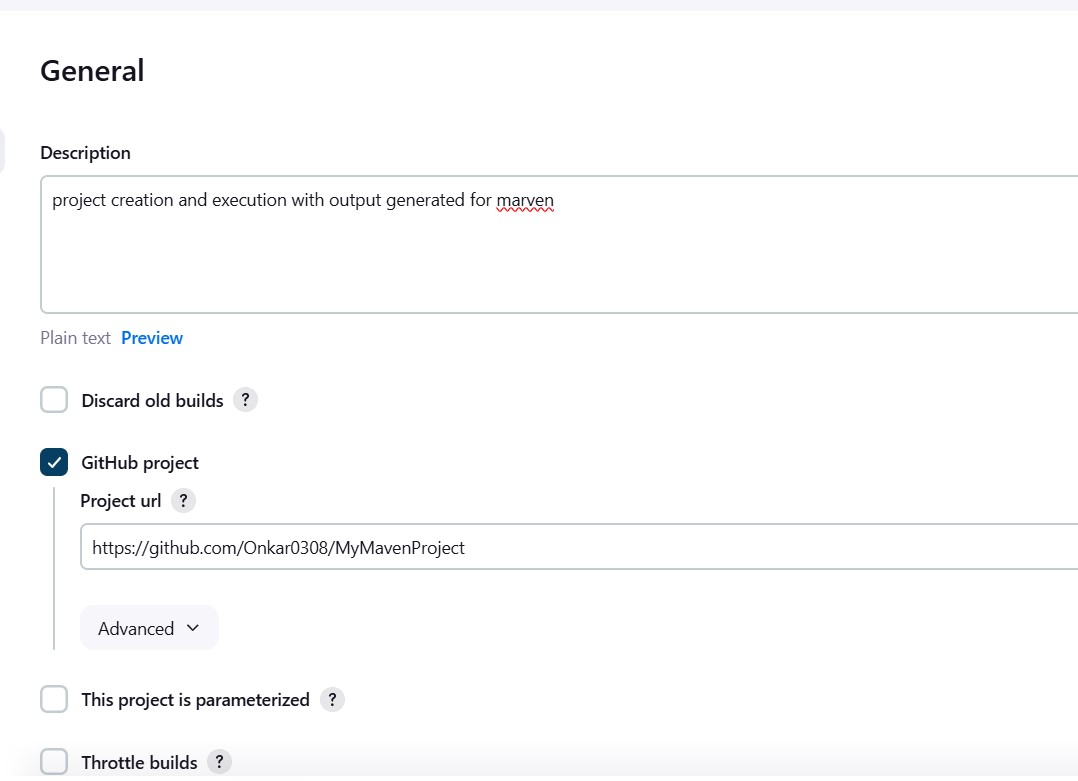
**Step 3: Create a New Maven Project**

1. **From the Jenkins dashboard, click on “New Item”.**
2. **Enter a name for your project, e.g., MyMavenProject.**
3. **Select “Maven Project” and click “OK”.**



**Step 4: Configure the Maven Project**

1. **Description: Optionally provide a description of your project.**



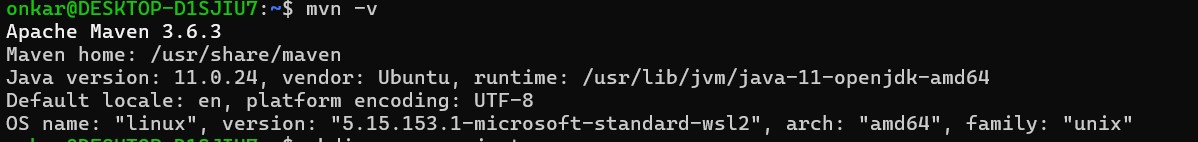
1. **GitHub project: If your Maven project is hosted on GitHub, you can provide the URL here.**

## IF you don’t Have Github repository follow this steps:-

**Step 1: Set Up Your Local Maven Project**

1. **Install Maven (if you haven’t already):**

**o Ensure that you have Maven installed on your machine. You can check by running mvn -v in your command line or terminal.**



1. **Create a New Maven Project:**
   * **Open your command line or terminal. o Navigate to the directory where you want to create your project:**

**o**



**Use the following Maven command to create a new project:**

**bash Copy code mvn archetype:generate -DgroupId=com.example -DartifactId=MyMavenProject -**

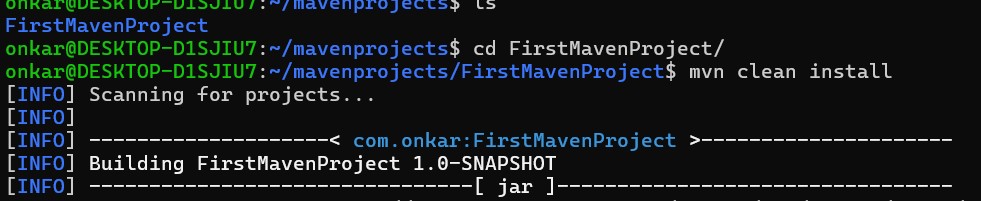
**DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**  **Replace com.example with your desired group ID.**

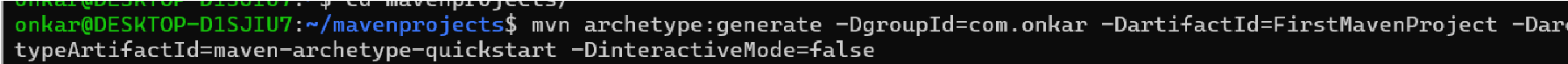
 **Replace MyMavenProject with your desired artifact ID.**

1. **Navigate to Your Project Directory:**

**bash**

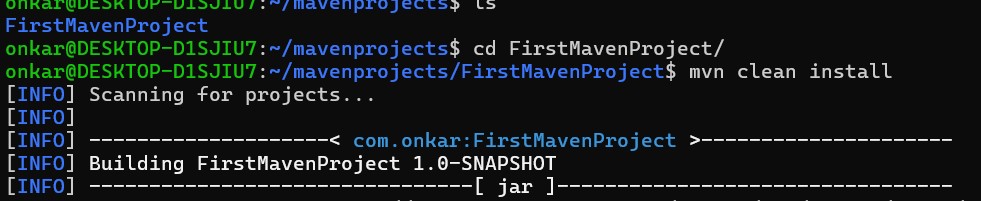
**4.**



**Build Your Project (optional):**

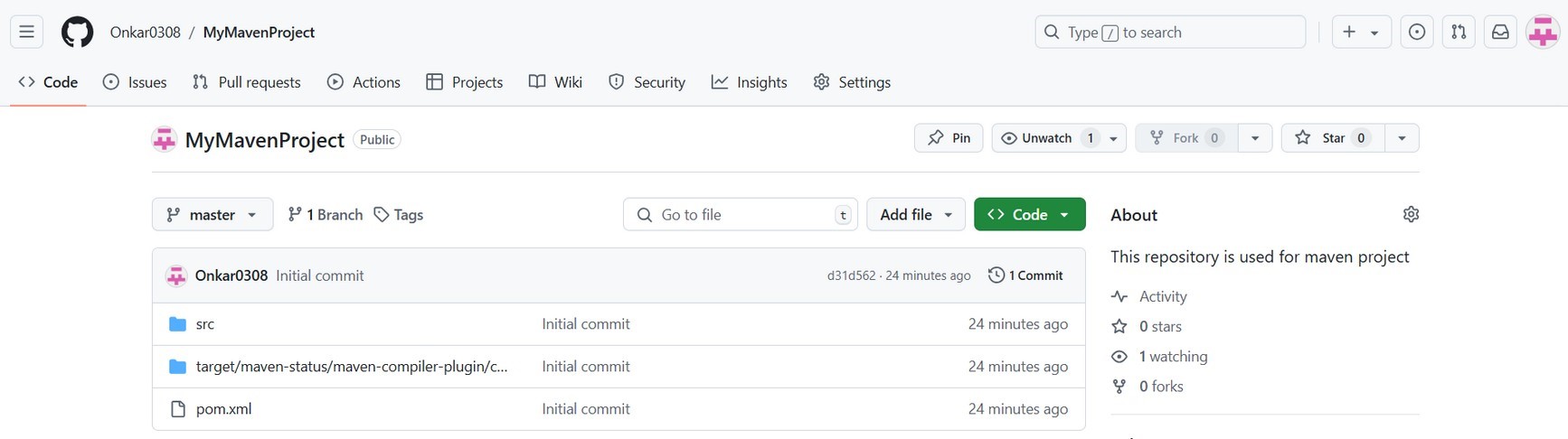
* + **You can build your Maven project to ensure it’s set up correctly:**

**bash Copy code mvn clean install**



**Step 2: Create a GitHub Repository**

1. **Log in to GitHub:**
   * **Open a web browser and go to** [**GitHub**](https://github.com/)**.**
   * **Log in to your account (or create one if you don’t have an account).**
2. **Create a New Repository:**
   * **Click on the "+" icon in the top right corner and select "New repository".**
   * **Fill in the details:**
     + **Repository name: Enter a name for your repository, e.g., MyMavenProject.**
     + **Description: Add an optional description.**
     + **Public/Private: Choose whether you want your repository to be public or private.**
   * **Do not initialize with a README: Since you will be pushing an existing project.**
   * **Click "Create repository".**

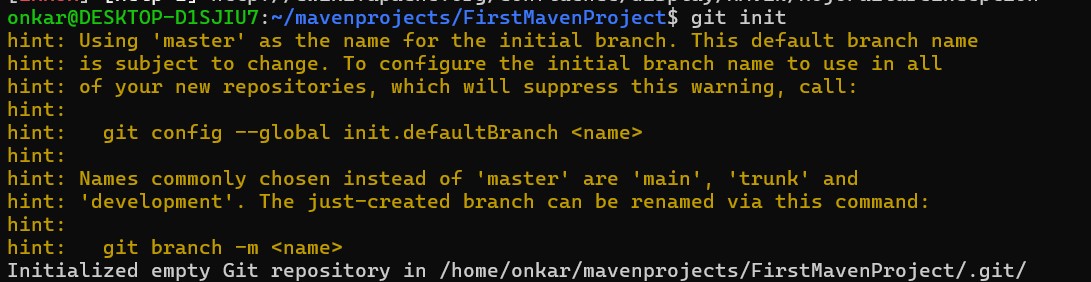


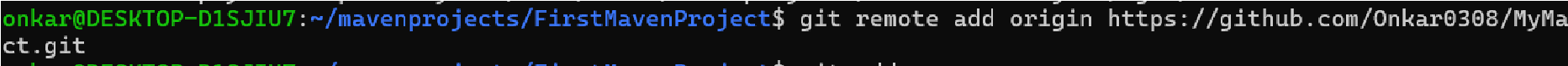
**Step 3: Initialize Git in Your Local Project**

**1. Initialize Git:**

**o In your command line or terminal, still within the project directory (MyMavenProject), run:**

**2.**



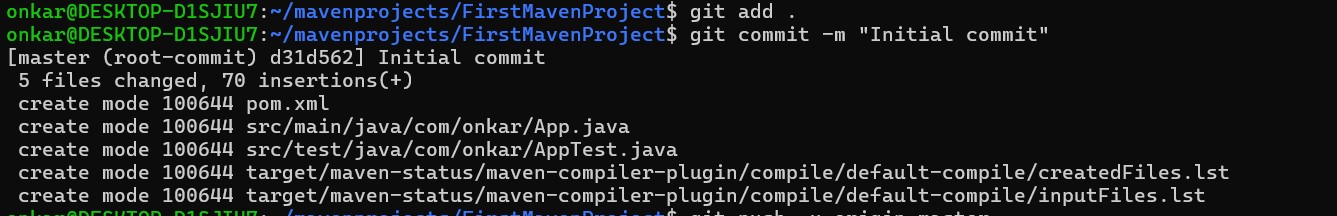
**Add Remote Repository: o Add the GitHub repository as a remote:**



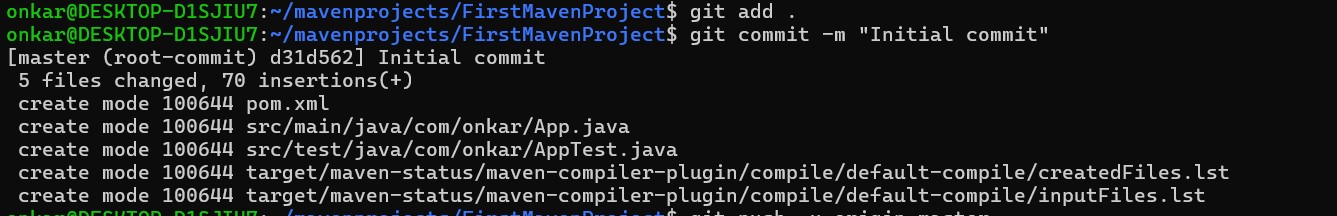
**Replace yourusername with your GitHub username and adjust the URL according to the repository name.**

**3. Stage Your Files:**

**o Stage all files in your project for commit: 4.**



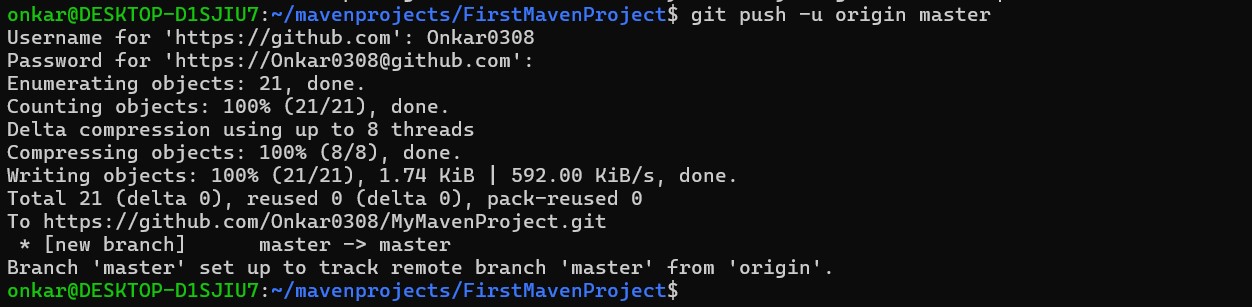
**Commit Your Changes: o Commit the staged files:**



**Step 4: Push Your Project to GitHub**

**1. Push to GitHub:**

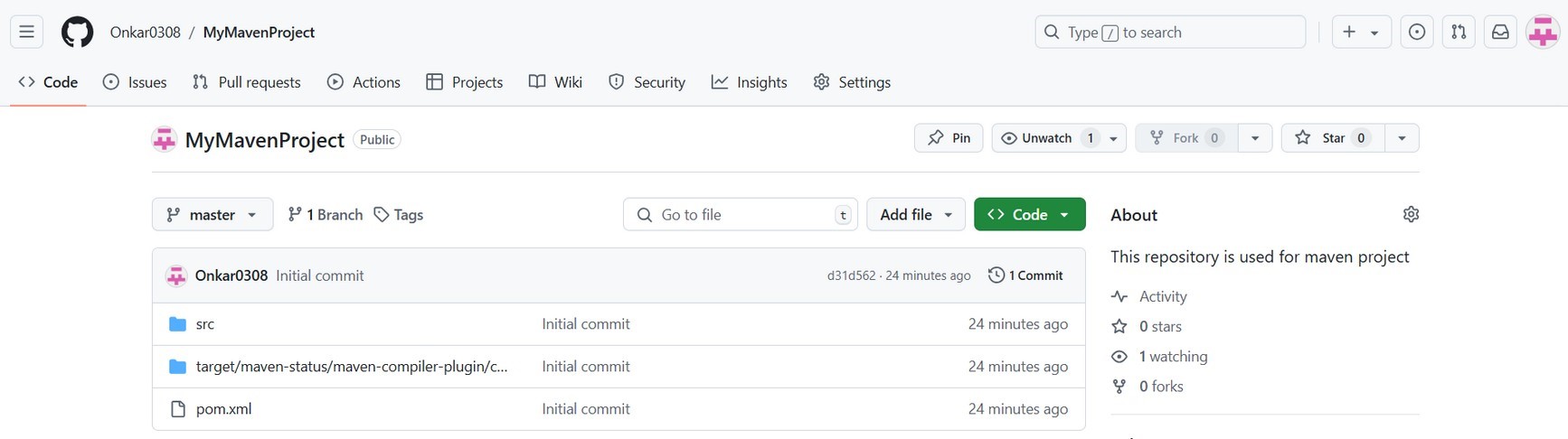
**o Push your local commits to the GitHub repository:**



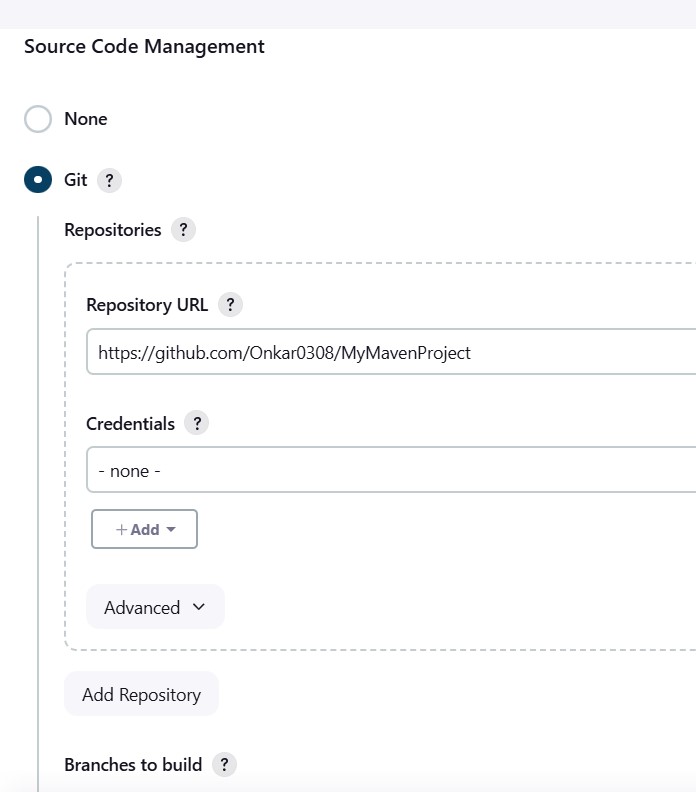
**Step 5: Verify Your Project on GitHub**

**1. Go back to GitHub:**

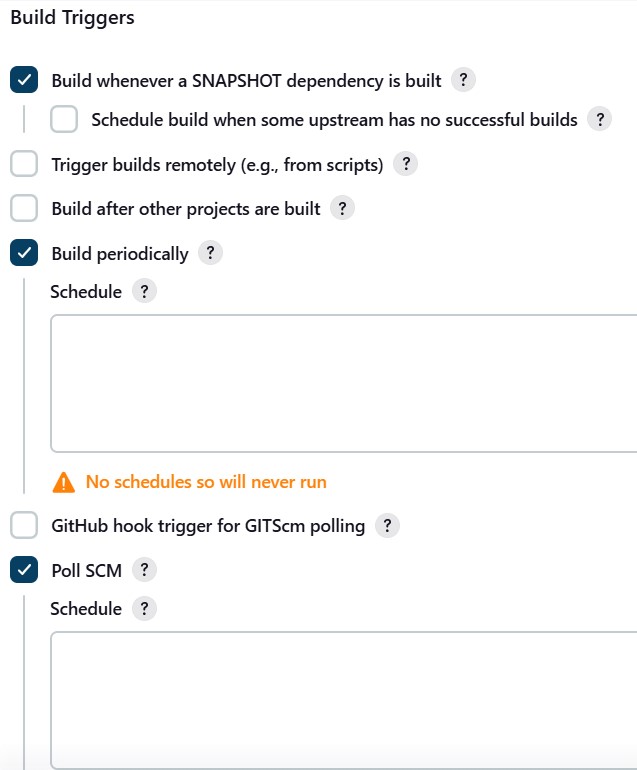
**o Refresh your repository page. You should see your Maven project files uploaded to GitHub.**

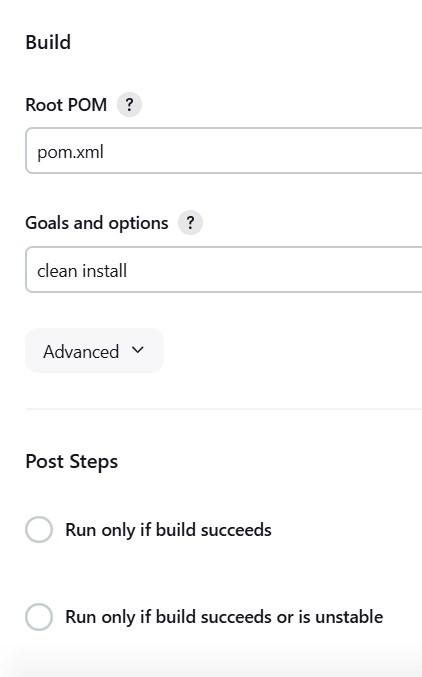
**ss**

1. **Source Code Management: o Choose “Git” if your project is in a Git repository.**
   * **Enter the repository URL and credentials if needed.**



1. **Build Triggers:**
   * **You can check options like “Poll SCM” or “Build periodically” depending on your needs.**



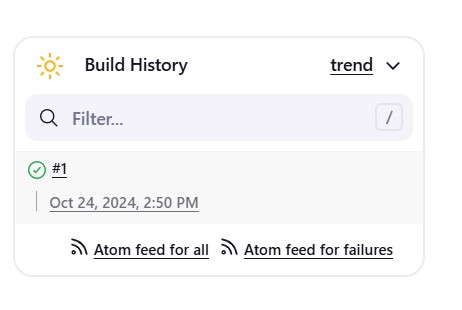
1. **Build:**
   * **In the “Goals and options” section, enter the Maven goals, for example, clean install to clean and build the project. o Optionally specify other parameters, such as -DskipTests to skip tests.**
2. **Advanced Settings: Click on “Advanced” if you need to set up specific Maven settings or profiles.**

**Step 5: Save the Configuration**

 **Click “Save” at the bottom of the page to store your project settings.**

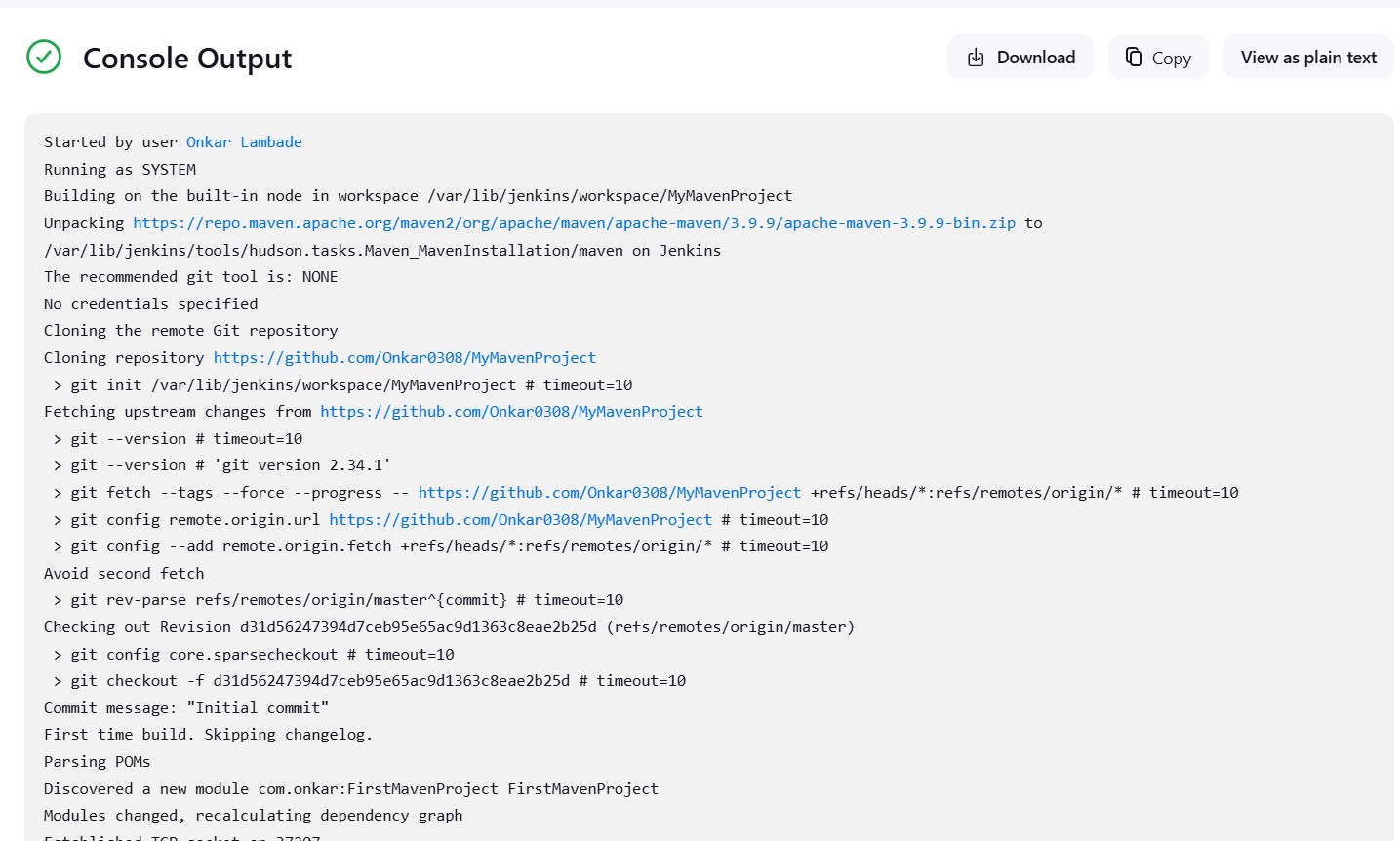
**Step 6: Build the Maven Project**

1. **On the project’s main page, you will see a “Build Now” option on the left sidebar. Click it to trigger the Maven build.**
2. **You will see a build history entry with a timestamp.**

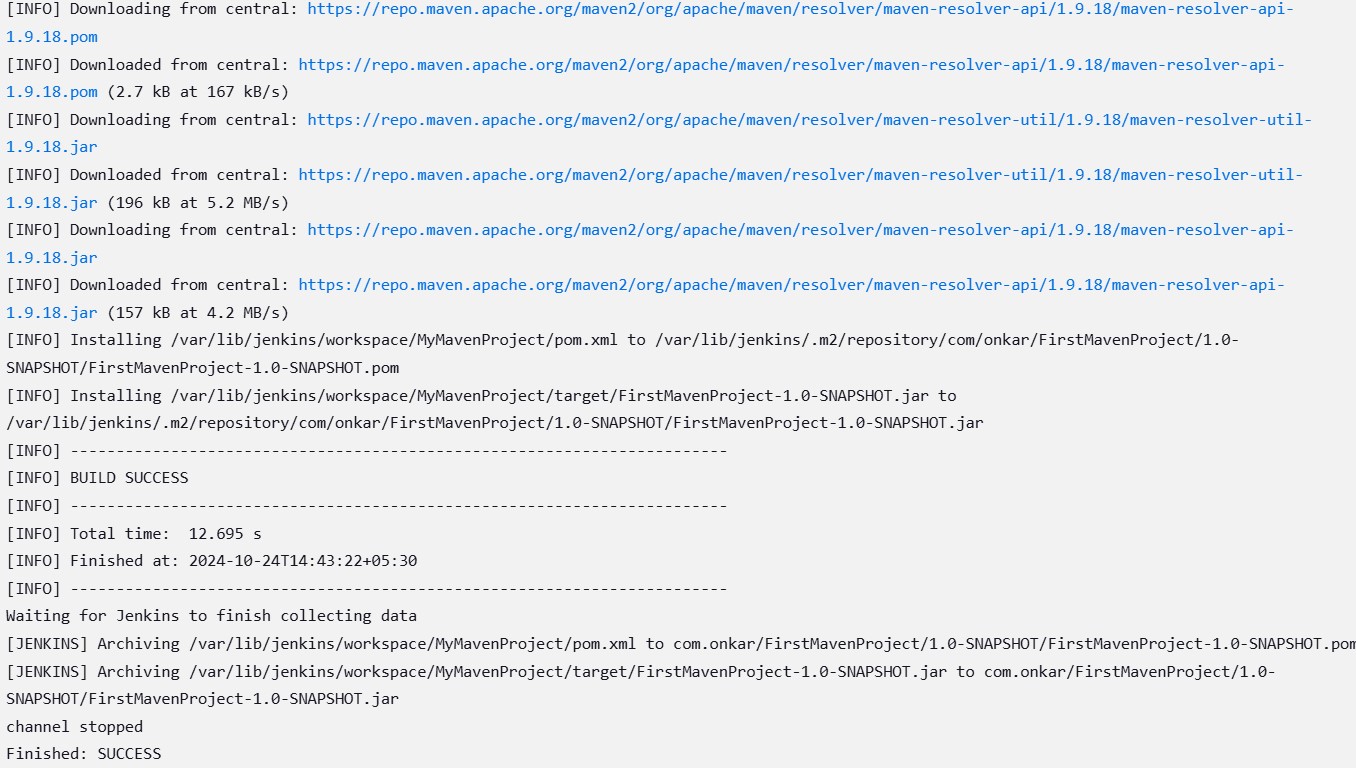


**Step 7: Check the Output**

1. **Click on the build number (usually labeled as #1 for the first build) under the “Build History” section to view build details.**
2. **Click on “Console Output” to see the logs of your Maven build process. Sample Output: You should see output logs similar to the following:**







**Conclusion :** Frame your conclusion here

**References:** Include your references here

**Rubrics**: 5 marks for 1st part and 5 mark for second part - 100- 80 % - 5, 80- 60%- 4 and so on **Experiment 4**

**Docker Installation & Basic Commands of Docker**

**Part A:**

Steps for Installing Docker:

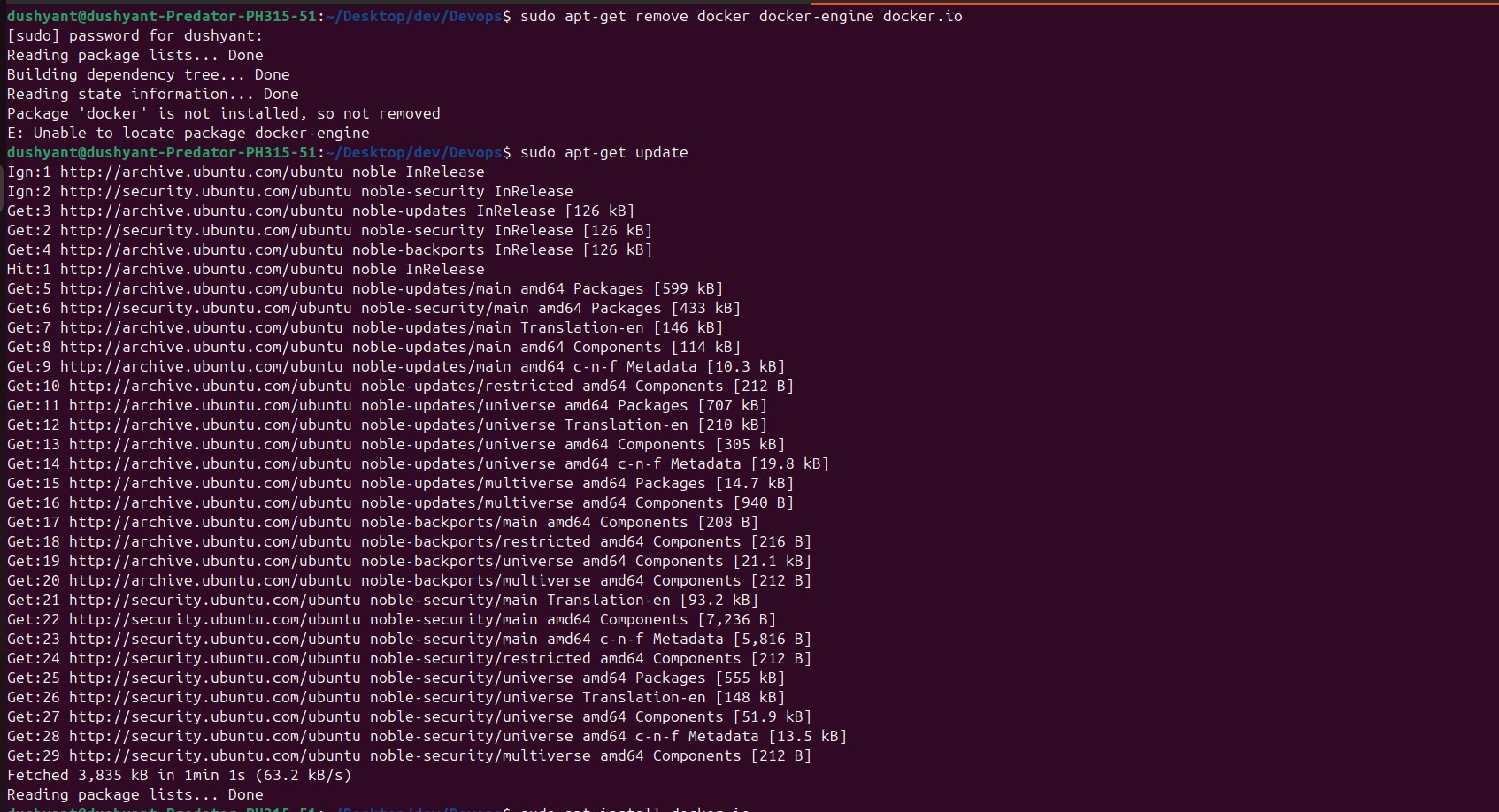
1. Open the terminal on Ubuntu.
2. Remove any Docker files that are running in the system, using the followingcommand:

$ sudo apt-get remove docker docker-engine docker.io

After entering the above command, you will need to enter the password of the root and press enter.

1. Check if the system is up-to-date using the following command:

$ sudo apt-get update



1. Install Docker using the following command:

$ sudo apt install docker.io

You’ll then get a prompt asking you to choose between y/n - choose y

1. Install all the dependency packages using the following command:

$ sudo snap install docker *alternate commands to install docker are*

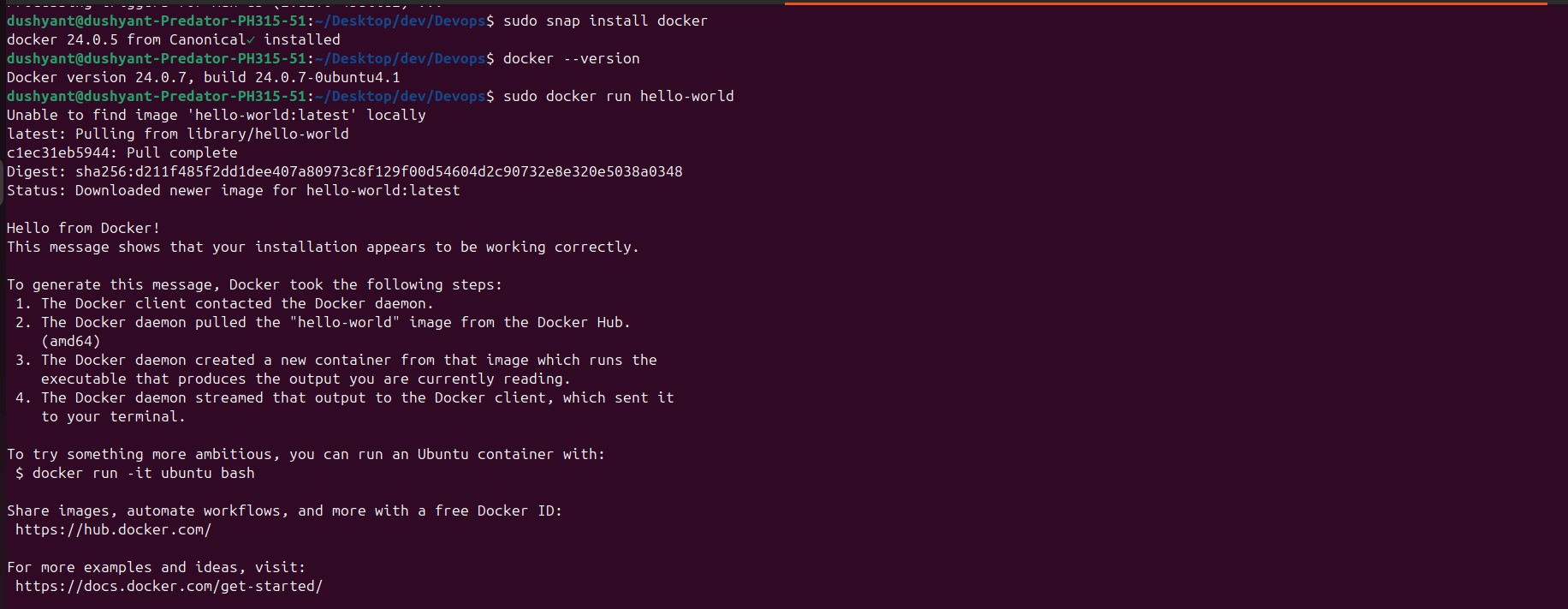
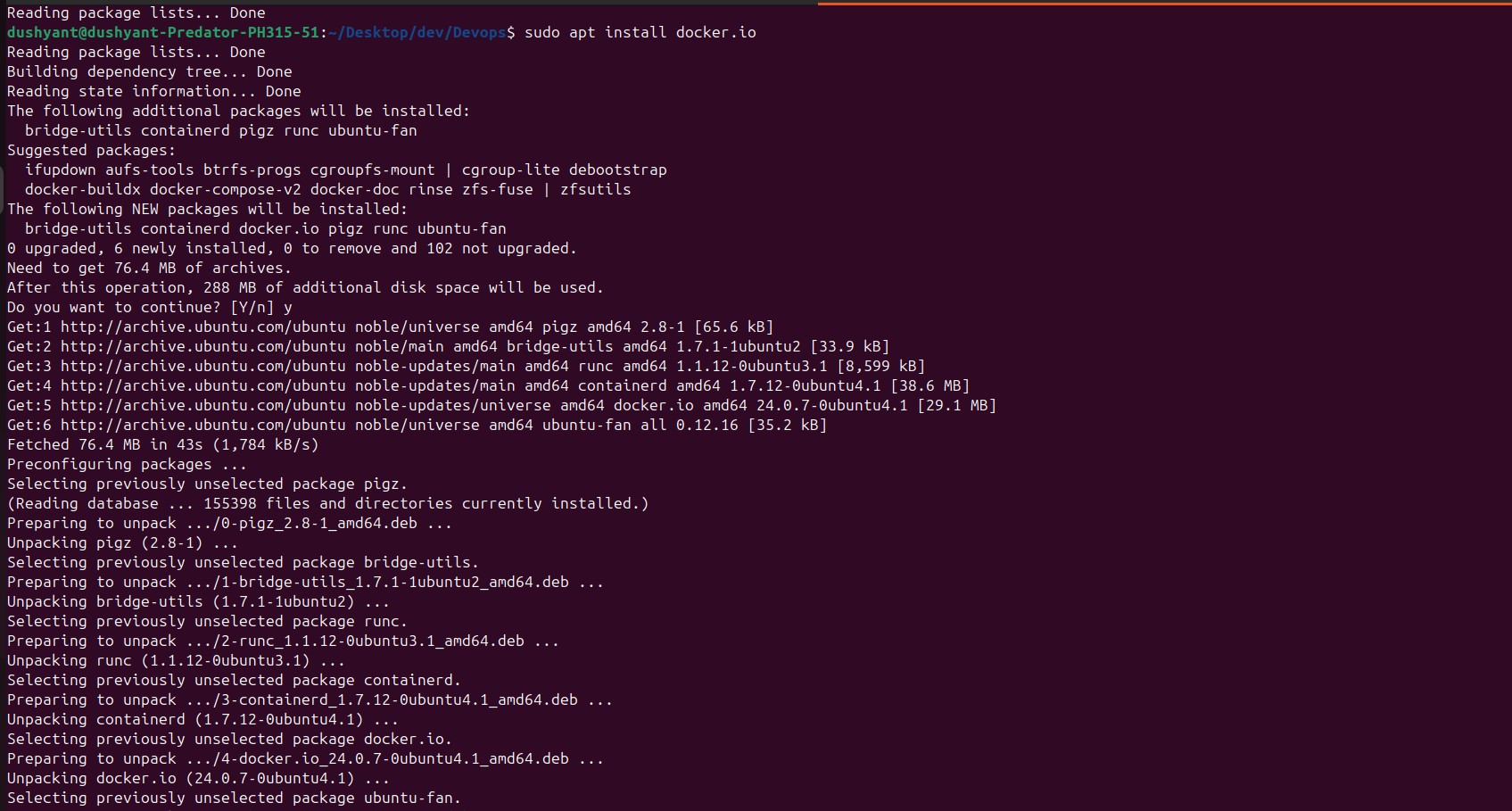
$ sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \ software-properties-common

To nstall packages to allow apt to use a repository over HTTPS

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add commad to add Docker’s official GPG key

$ sudo apt-key fingerprint 0EBFCD88

Verify that you now have the key with the fingerprint



1. Before testing Docker, check the version installed using the following command: $ docker –version
2. Pull an image from the Docker hub using the following command:

$ sudo docker run hello-world

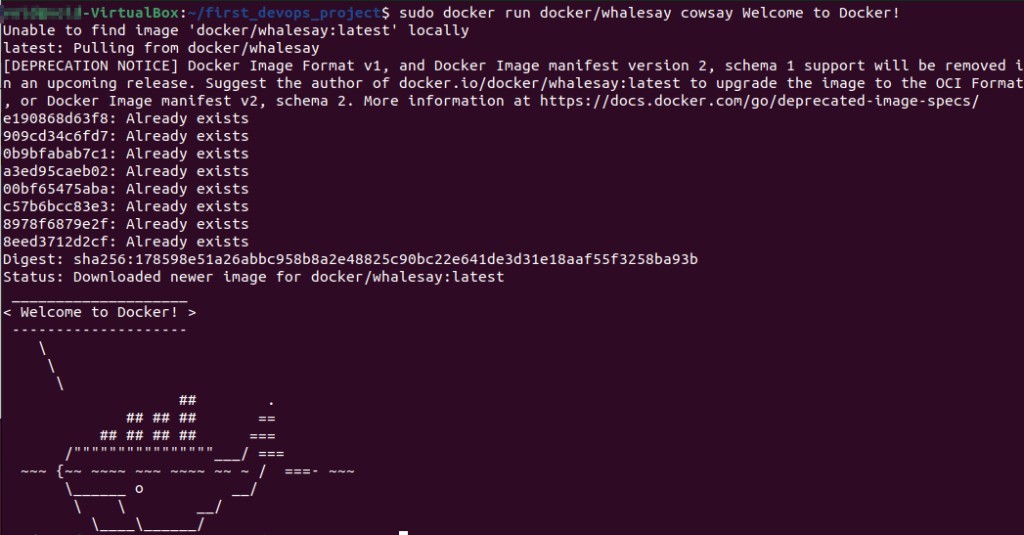
Here, hello-world is the docker image present on the Docker hub. Output will be like this as in figure.



1. The actual Hellow World command of docker is

$ docker run docker/whalesay cowsay boo

The default image of docker appears with the message boo.

Check

if the

docker image has been pulled and is present in your system using the

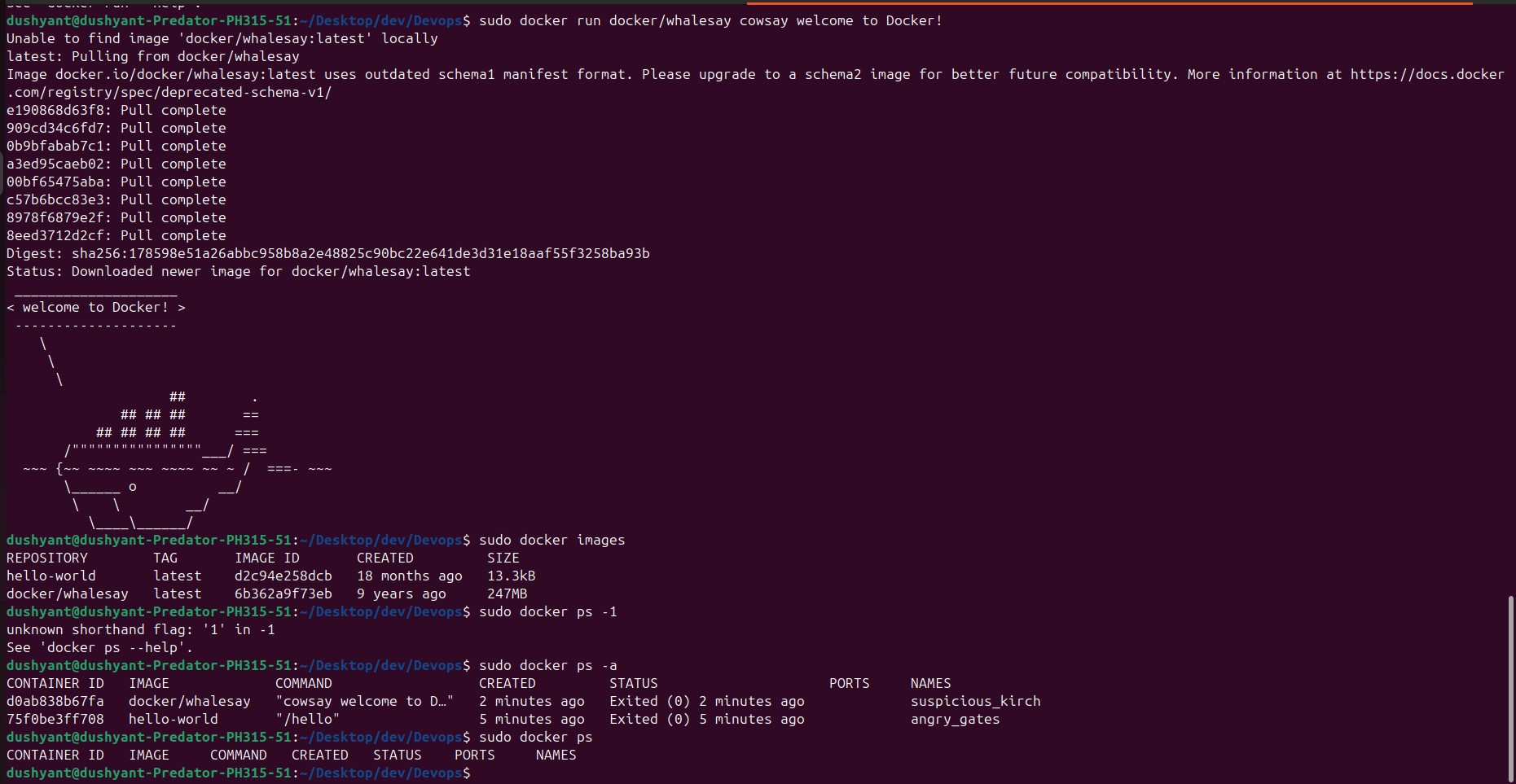
following command:

$ sudo docker images

To display all the containers pulled, use the following command:

$ sudo docker ps -a

To check for containers in a running state, use the following command: $ sudo docker ps



**Part B**: Docker search, docker Pull and docker run

Use the command docker search to search for public images on the Docker hub. It will return information about the image name, description, stars, official and automated.

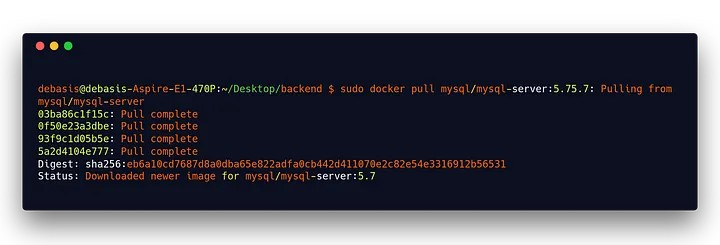
Now that we know the name of the image, we can pull that from the Docker hub using the command docker pull. Here, we are setting the platform option as well.

$ sudo docker search mysql

*or alternate*

$ sudo docker pull -platform linux/x86\_64 mysql *or alternate*

$sudo docker pull mysql/mysql-erver:tag



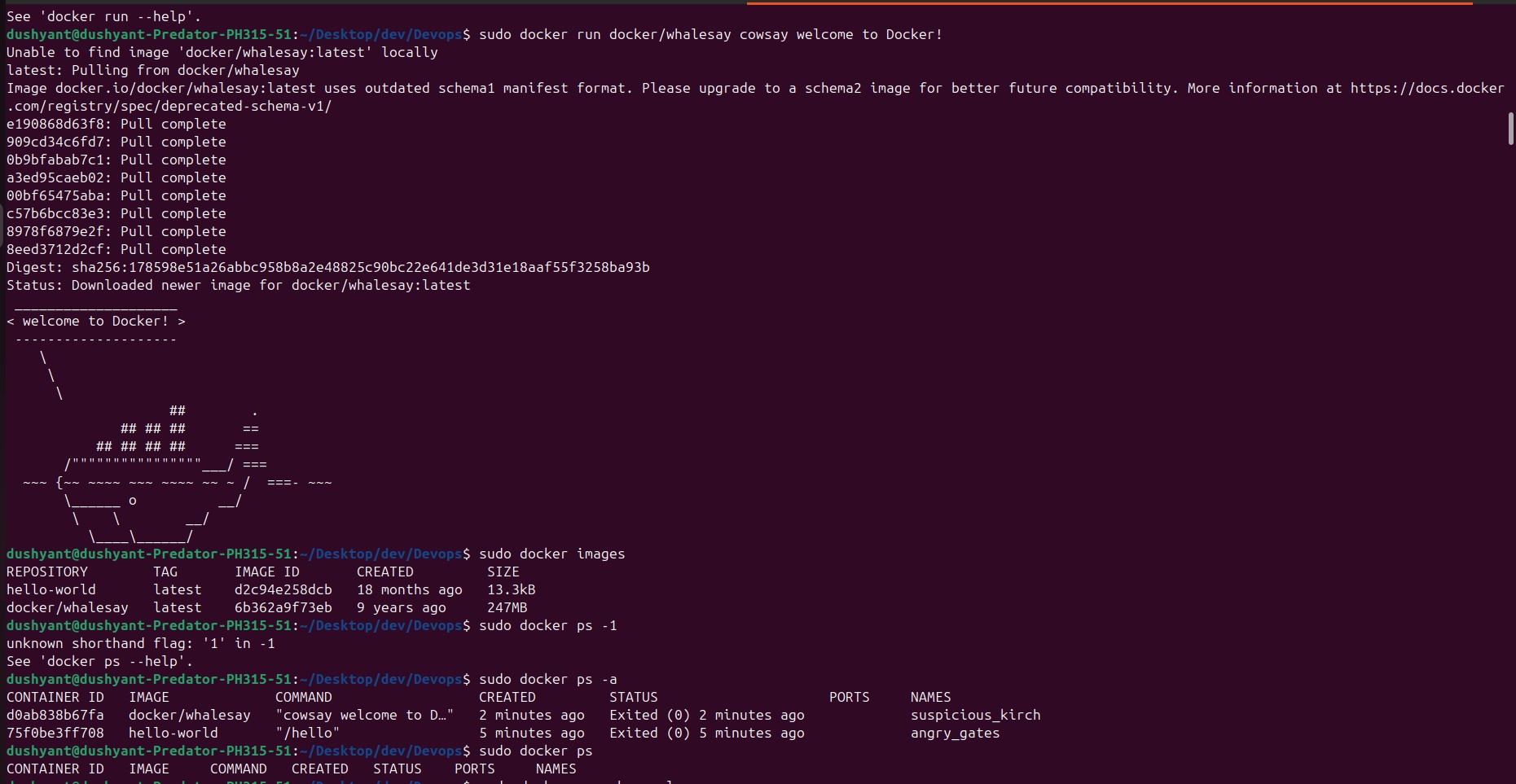
Log into MySQL within the docker container using the docker exec command: $ sudo docker exec -it mysql bash

Now run this command

$ sudo docker run --name mysql -p 3406:3306 -e MYSQL\_ROOT\_PASSWORD=anypassword d mysql/mysql-server:5.7

You can check it by running the following command…The first image as you can see in the snippet is the mysql-server image in anew terminal

$ sudo docker ps -a



Remember, when we created and ran the MySQL container, we provided

MYSQL\_ROOT\_PASSWORD=anypassword. Create a database and user, and grant privileges in MySQL (from within the container). og into MySQL within the docker container using the docker exec command, Log into MySQL if you haven’t already. After login, the mysql> prompt shows up:

$ mysql -uroot -panypassword

$ SHOW DATABASES ;

$ use database;

$ exit

$ exit

$docker restart

Let’s restart our stopped contained by using the following command. We may want to use this after we reboot our machine. docker restart f8c52bedeecc

$docker rename

Now, let’s change the container name from compassionate\_fermi to test\_db. We may want to change the name to keep track of our containers more easily. docker rename compassionate\_fermi test\_db

$docker exec

Access the running container test\_db by running the following command. It’s helpful, if we want to access the MySQL command line and execute MySQL queries.

docker exec -it test\_db bash mysql -uroot -pmy-secret-pw SHOW DATABASES;

The -i and -t options are used to access the container in an interactive mode.

Then we provide the name of the container we want to access, which in this case test\_db. Finally, the bash command is used to get a bash shell inside the container.

$docker logs

This command is helpful to debug our Docker containers. It will fetch logs from a specified container. $docker logs test\_db

If we want to continue to stream new output,use the option -follow. docker logs -follow test\_db $docker rm

To remove a container, we can use the following command. docker rm test\_db

You may encounter an error like

Error response from daemon: You cannot remove a running container ………

Stop the container before attempting removal or force remove

As it recommends, we can stop the container first and then remove it or use option -f to remove a running container forcefully.

$docker stop test\_db

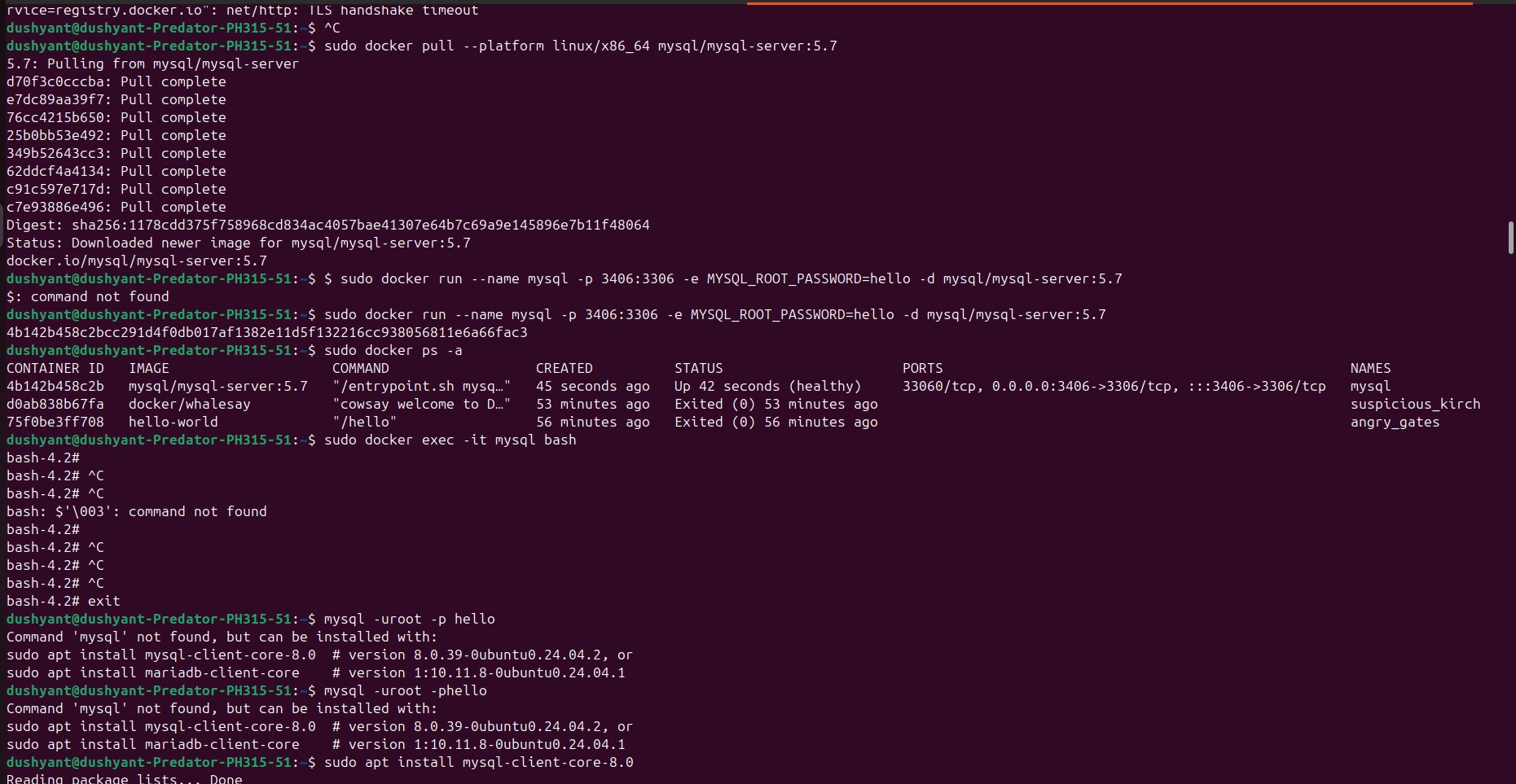
$docker rm test\_db# ordocker rm -f test\_db

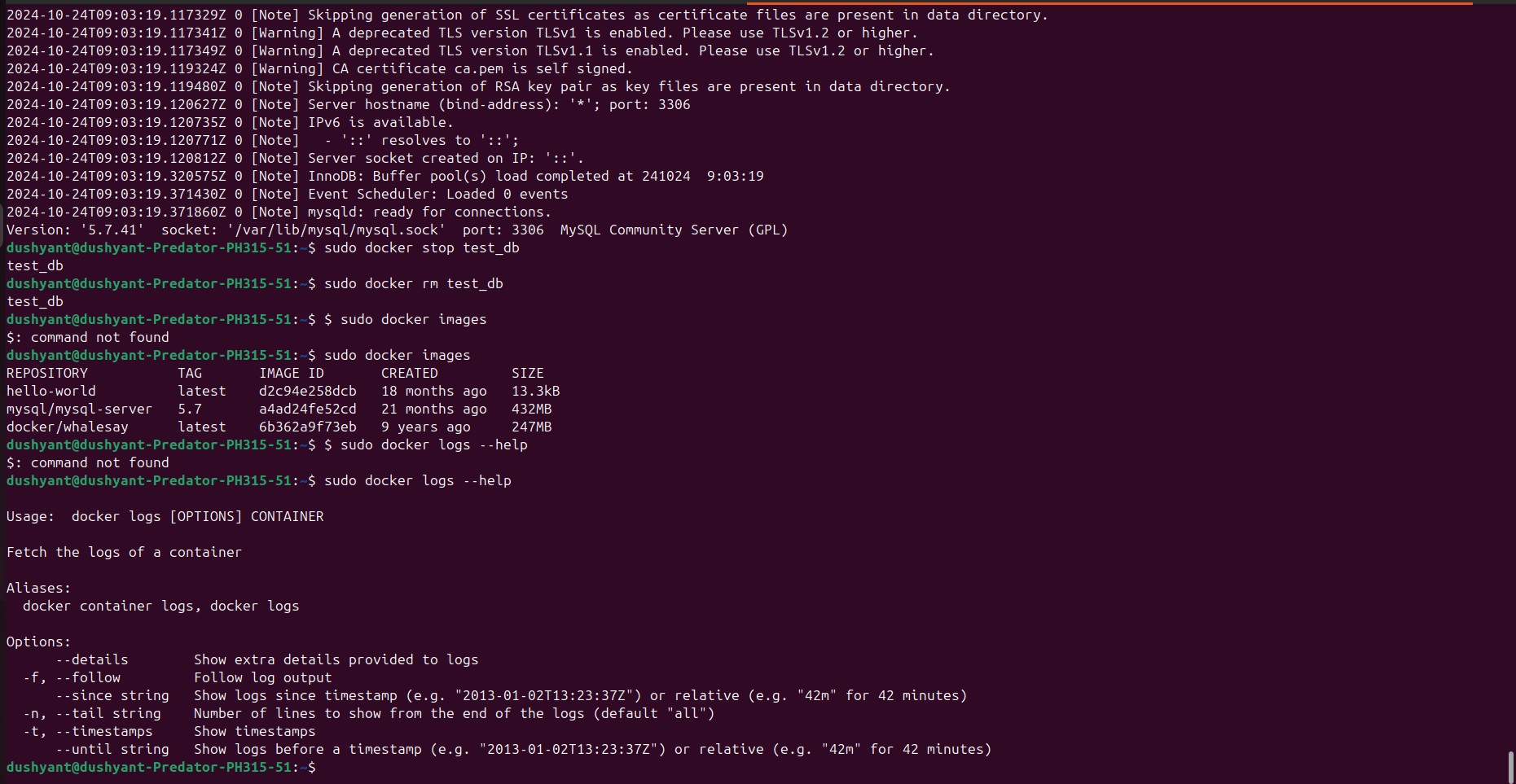
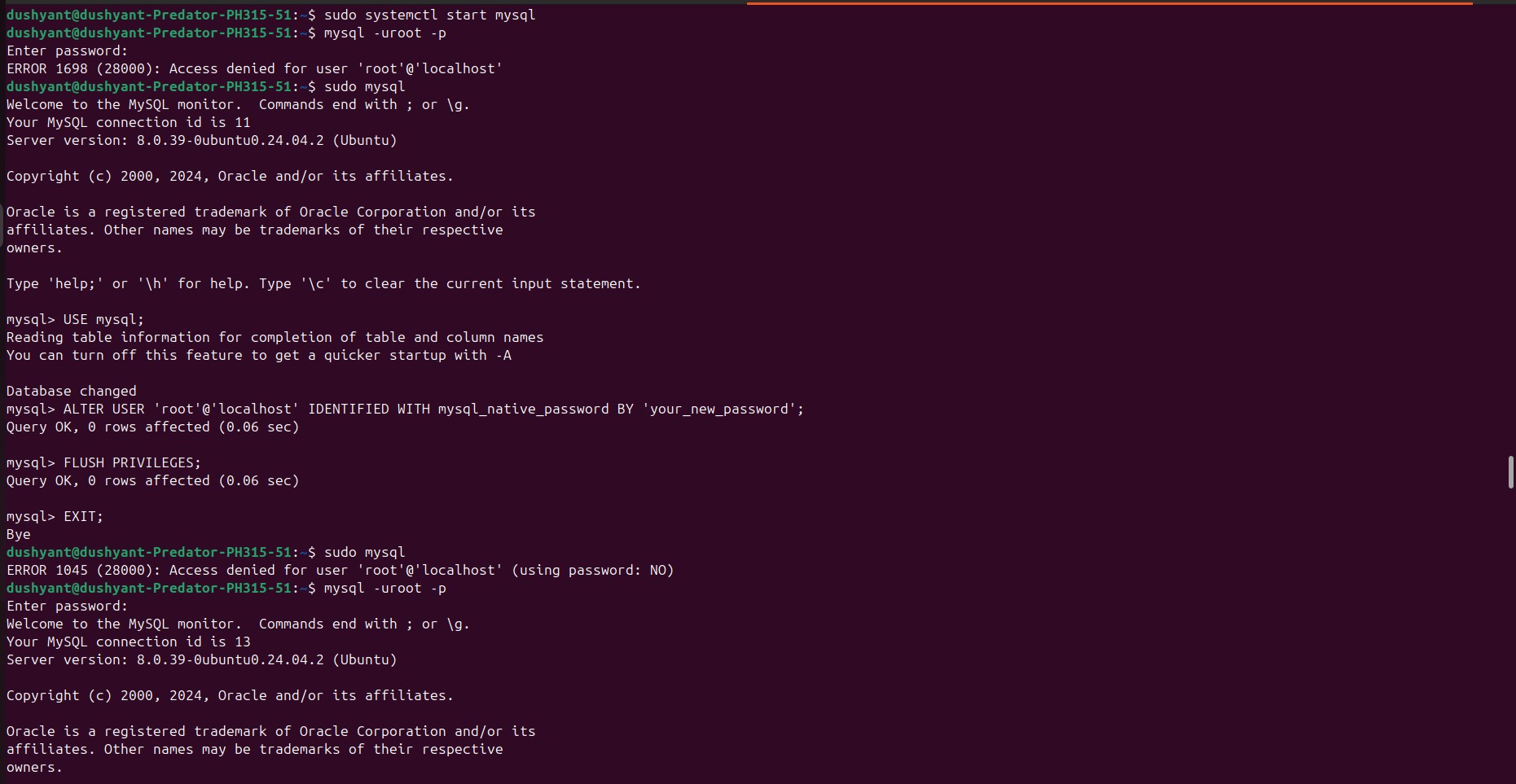
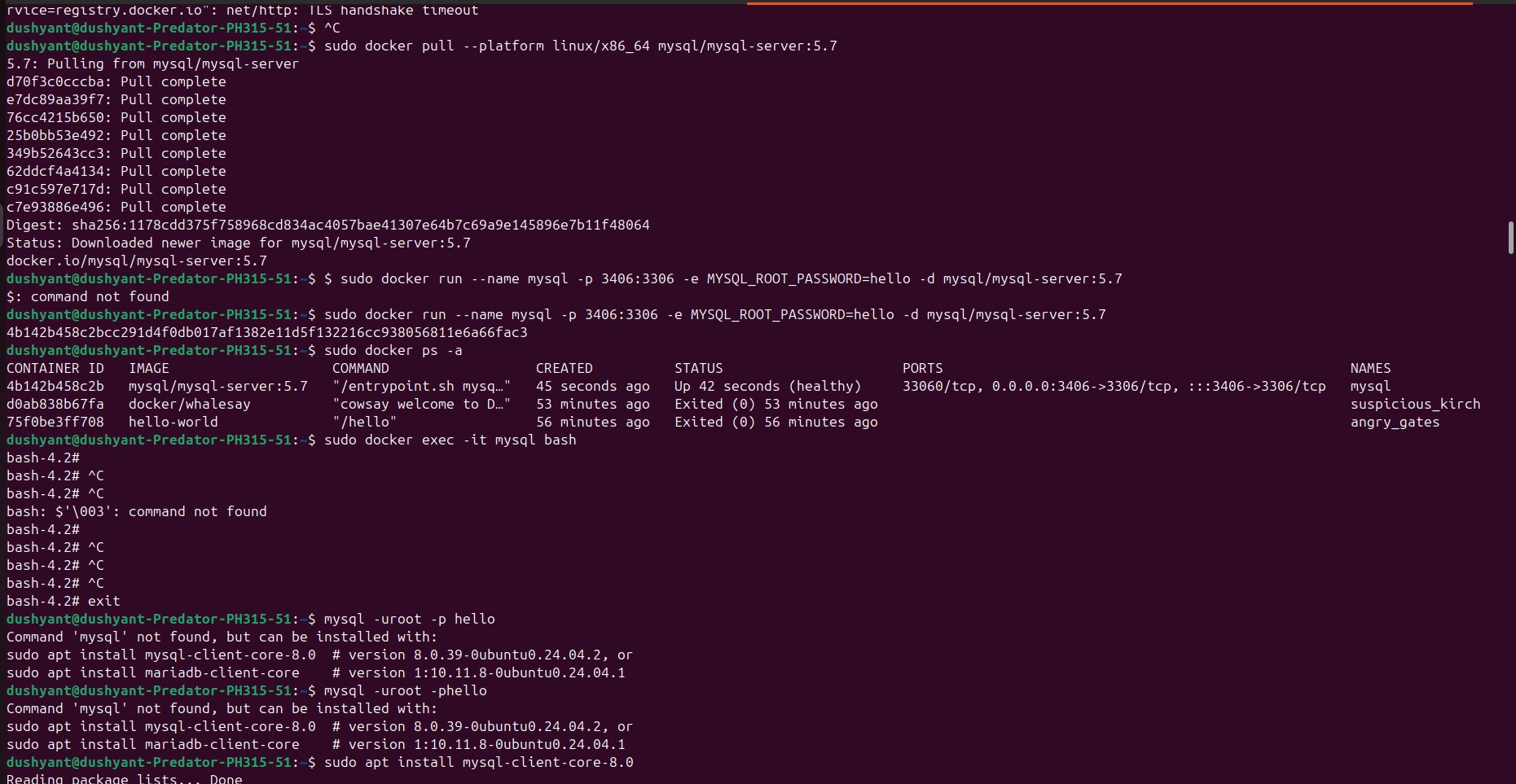
$docker rmi

To free some disk space, we can use the docker rmi command with the image id to remove an image. $docker rmi eb0e825dc3cf

These commands come with plenty of helpful options. If you want to know about other available options, run the docker command\_name --help command. For example:

docker logs –help





Reference:

1. https://www.simplilearn.com/tutorials/docker-tutorial/how-to-install-docker-

on-

ubuntu

1. https://towardsdatascience.com/12-essential-docker-commands-you-should-know-

c2d5a7751bb5

1. https://docs.docker.com/engine/reference/commandline/container/
2. https://towardsdatascience.com/15-docker-commands-you-should-know-

970ea5203421

# Mini-Project

**Aim: Installation and demonstration of Chef**

# About the Tutorial

Chef is a configuration management technology developed by Opscode to manage infrastructure on physical or virtual machines. It is an open source developed using Ruby, which helps in managing complex infrastructure on the fly.

This tutorial provides a basic understanding of the infrastructure and fundamental concepts of managing an infrastructure using Chef.

# Audience

This tutorial has been prepared for those who want to understand the features and functionality of Chef and how Chef can help in reducing the complexity of managing an infrastructure.

After completing this tutorial one would have moderate level understanding of Chef and its key building blocks. It will also give a fair idea on how to configure Chef in a preconfigured infrastructure and how to use it.

# Prerequisites

We assume anyone who wants to learn Chef should have an understanding of system administration, infrastructure and network protocol communication. To automate the infrastructure provisioning, one should have a command over basic Ruby script writing and the underlying system where one wants to use Chef.

### On Windows

**Step 1:** Download the Windows installer from [www.git-scm.org](http://www.git-scm.org/)and follow the installation steps.

**Step 2:** Sign up for a central repository on GitHub.

**Step 3:** Upload the ssh key to the GitHub account, so that one can interact with it easily. For details on ssh key visit the following link [https://help.github.com/articles/generating-sshkeys.](https://help.github.com/articles/generating-ssh-keys)

**Step 4:** Finally create a repo on the github account by visiting <https://github.com/new>with the name of chef-repo.

Before actually starting to write a cookbook, one can set up an initial GIT repository on the development box and clone the empty repository provided by Opscode.

**Step 1:** Download Opscode Chef repository empty structure.

$ wget https://github.com/opscode/chef-repo/tarball/master

**Step 2:** Extract the tar ball.

$ tar –xvf master

**Step 3:** Rename the directory.

$ mv opscode-chef-repo-2c42c6a/ chef-repo

**Step 4:** Change the current working directory to chef repo.

$ cd chef-repo

**Step 5:** Initialize a fresh get repo.

$ git init .

**Step 6:** Connect to your repo on the git hub.

$ git remote add origin git@github.com:vipin022/chef-

**Step 7:** Push the local repo to github.

$ git add .

$ git commit –m “empty repo structure added”

$ git push –u origin master

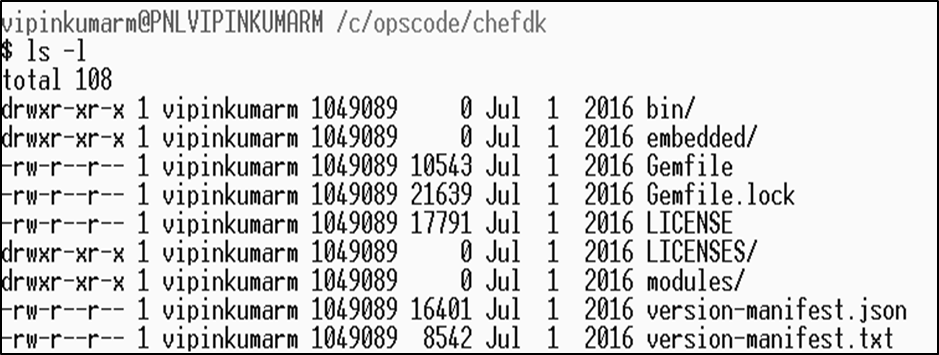
By using the above procedure, you will get an empty chef repo in place. You can then start working on developing the recipes and cookbooks. Once done, you can push the changes to the GitHub.

### On Windows Machine

**Step 1:** Download the setup .msi file of chefDK on the machine.

**Step 2:** Follow the installation steps and install it on the target location.

The setup will look as shown in the following screenshot.



#### ChefDK Path Variable

$ echo $PATH

/c/opscode/chef/bin:/c/opscode/chefdk/bin:

### On Linux Machine

In order to set up on the Linux machine, we need to first get curl on the machine

**Step 1:** Once curl is installed on the machine, we need to install Chef on the workstation using Opscode’s omnibus Chef installer.

$ curl –L <https://www.opscode.com/chef/install.sh>| sudo bash

**Step 2:** Install Ruby on the machine.

**Step 3:** Add Ruby to path variable.

$ echo ‘export PATH=”/opt/chef/embedded/bin:$PATH”’ >> ~/.bash\_profile && source ~/.bash\_profile

The Omnibus Chef will install Ruby and all the required Ruby gems into **/opt/chef/embedded** by adding **/opt/chef/embedded/bin** directory to the .bash\_profile file.

If Ruby is already installed, then install the Chef Ruby gem on the machine by running the following command.

$ gem install ch