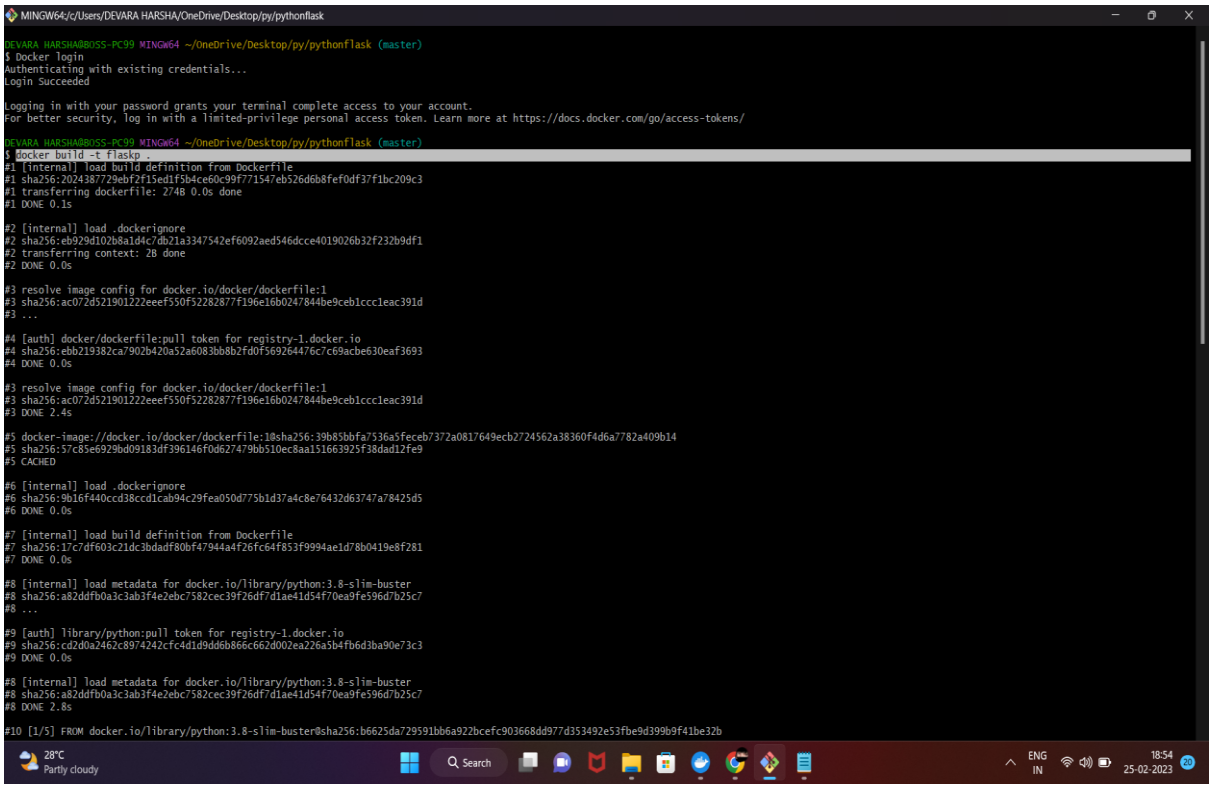
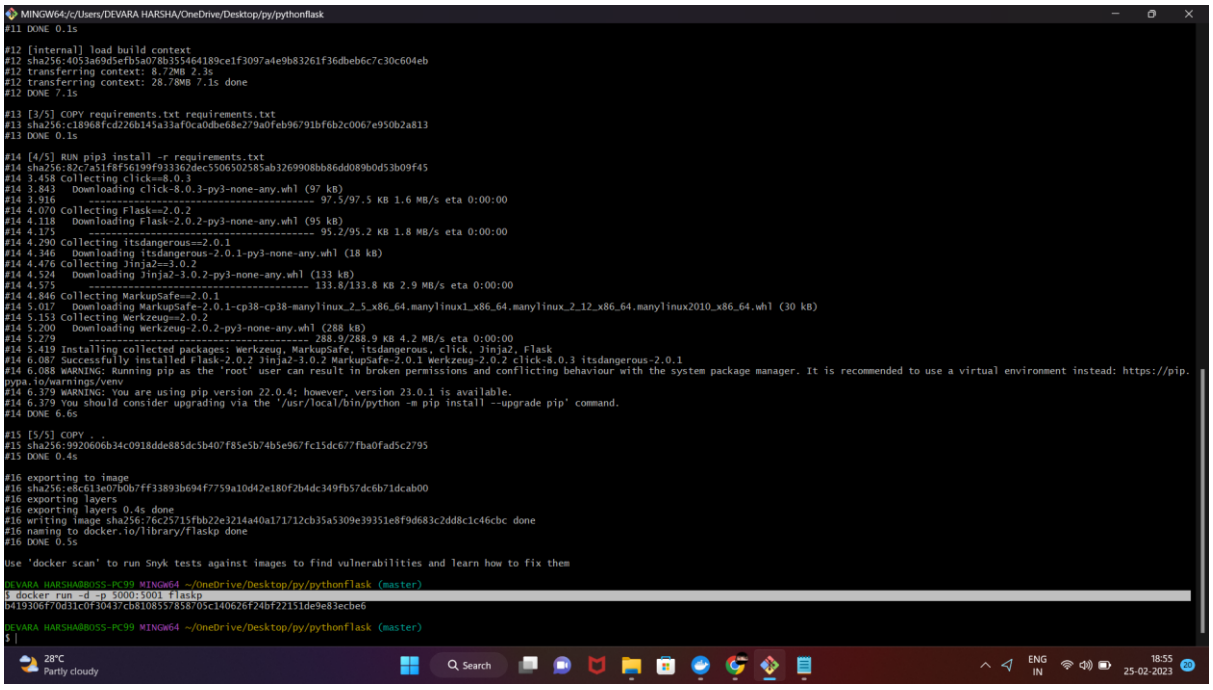
DEVARA HARSHA
20P31A04J4

Q1. Create a docker container and image out of Dockerfile for following application:

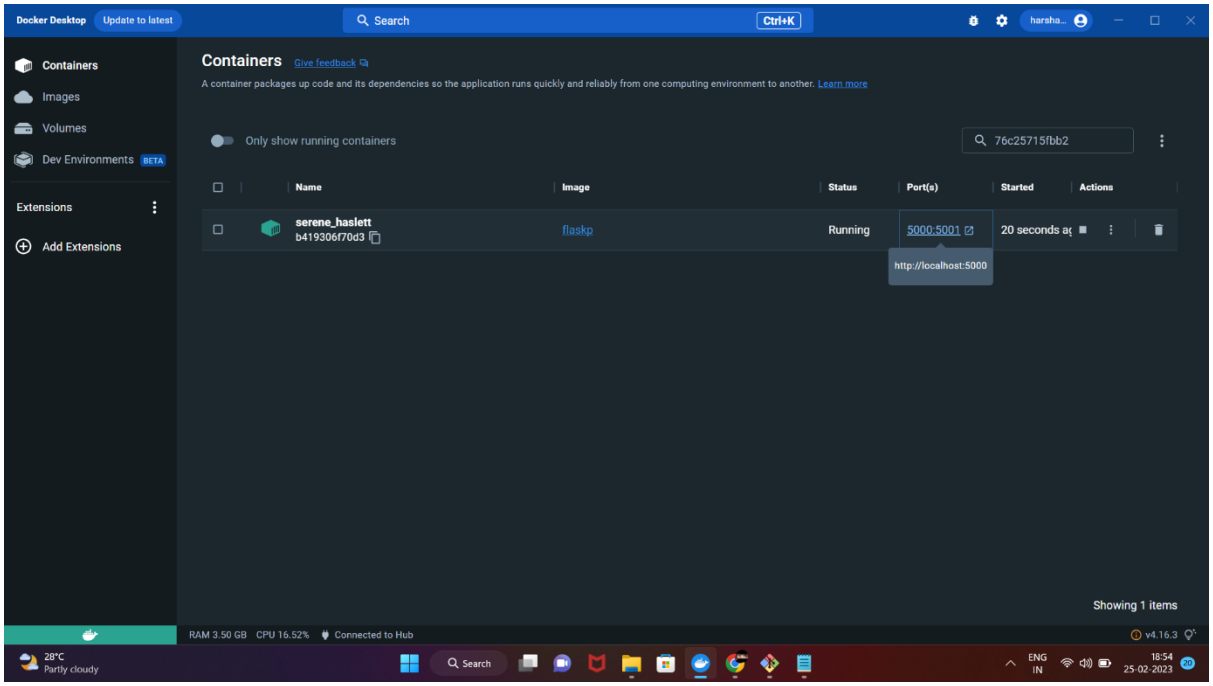
Python Flask





DevOps Graded Assignment-3

DEVARA HARSHA
20P31A04J4



welcome to the flask tutorials

DEVARA HARSHA
20P31A04J4

FROM : it defines which service image to be used for your application

WORKDIR : it defines the folder structure of the application inside the container and

where everything you download will be .

COPY file_which_contains_our_application_dependency.

example : python flask: requirements.txt

RUN : it tells docker to perform installation of whatever is there in your dependency file.

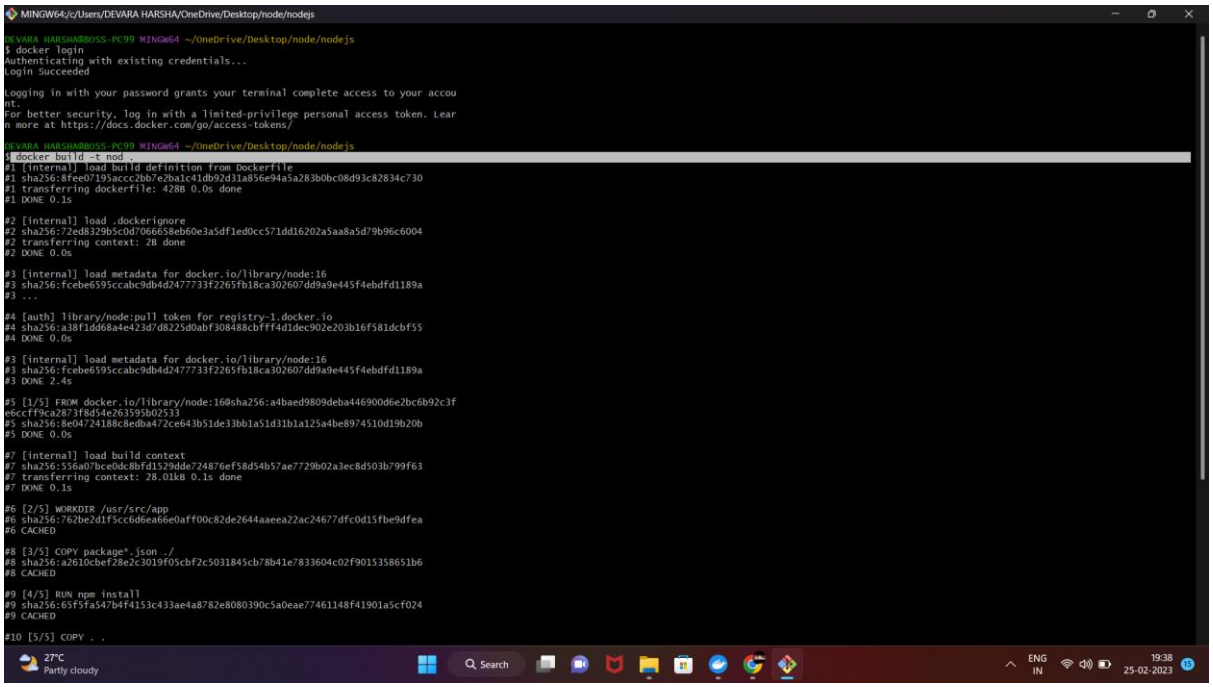
COPY . . : it tells to copy all the stuff just got downloaded and keep it in the desired folder structure

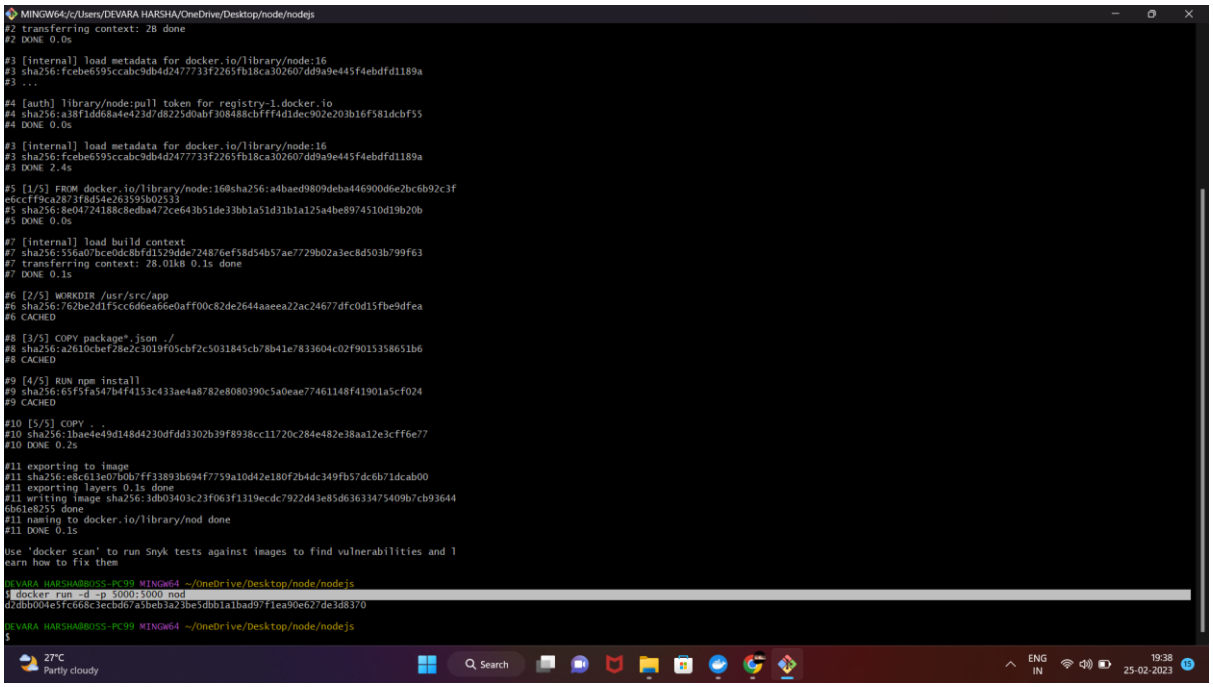
EXPOSE : it tells docker that our application will run on port 3000 inside the container.

CMD : it tells docker to run this command in command line environment

to start the application.

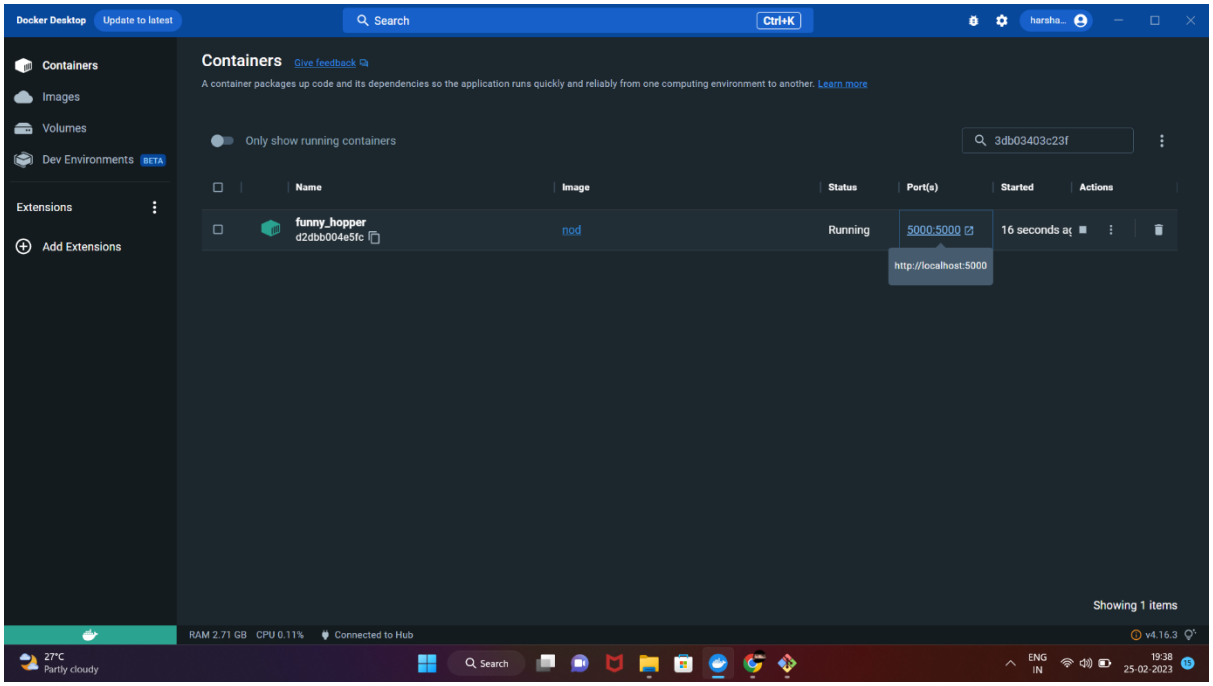
DEVARA HARSHA
20P31A04J4



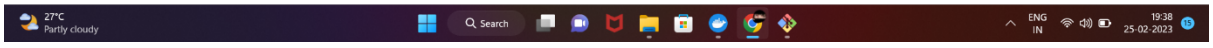


DevOps Graded Assignment-3

DEVARA HARSHA
20P31A04J4



hello docker



DEVARA HARSHA
20P31A04J4

FROM : it defines which service image to be used for your application

WORKDIR : it defines the folder structure of the application inside the container and

where everything you download will be .

COPY file_which_contains_our_application_dependency.

example : Nodejs : Package.Son ,: requirements.txt

RUN : it tells docker to perform installation of whatever is there in your dependency file.

COPY . . : it tells to copy all the stuff just got downloaded and keep it in the desired folder structure

EXPOSE : it tells docker that our application will run on port 3000 inside the container.

CMD : ["node","index.js"] : it tells docker to run this command in command line environment

to start the application

DEVARA HARSHA
20P31A04J4

Q2.Create a CI-CD pipeline for a Nodejs Application in jenkins and all the steps involved in it should be given in a screenshot and your jenkins username must be visible in the screenshot.

Sol: CI-CD PIPELINE USING JENKINS

Step 1 : Push all files of your application into the git hub repository.

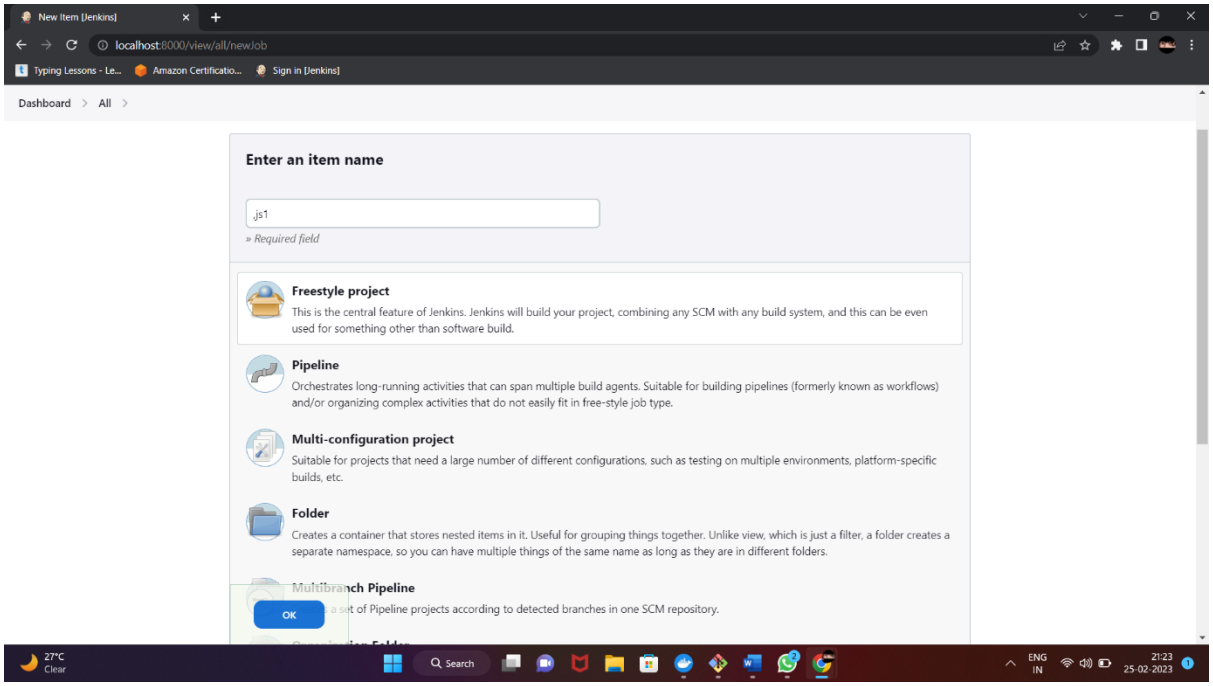
Step 2 : Create a docker image for that, where the docker file is present and push into the docker hub.

Step 3 : Now you can go to the Jenkins dashboard.

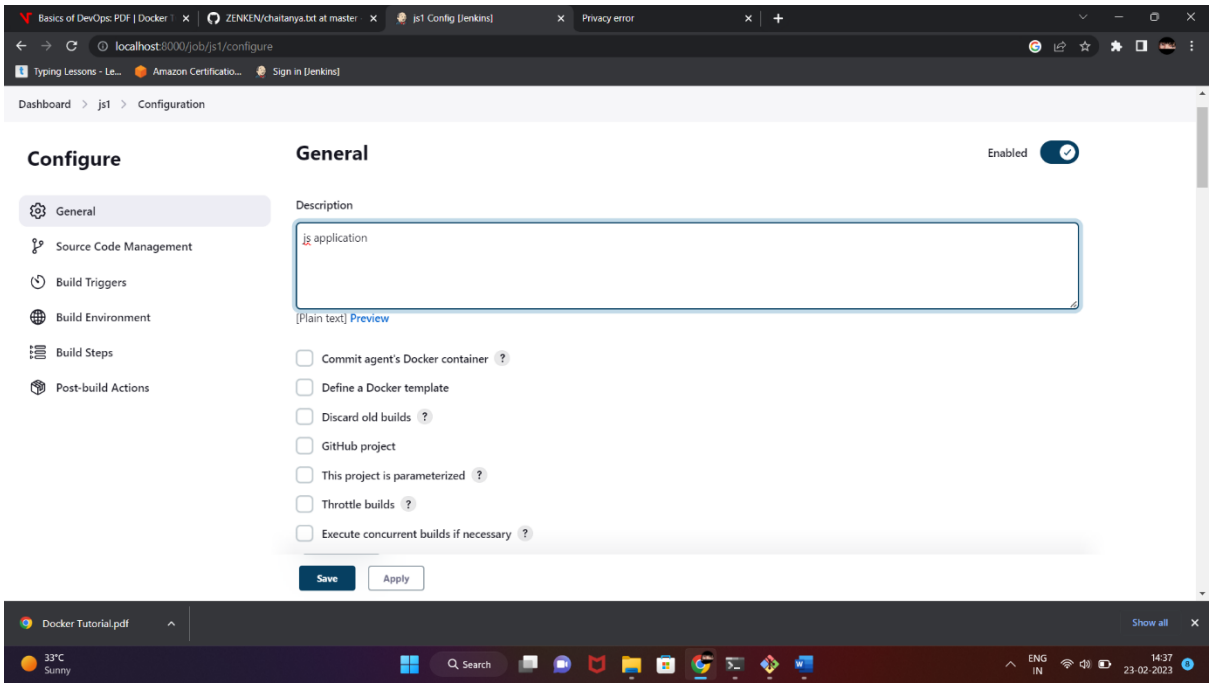
Step 4: create a new item and apply suitable configure settings based on your application.

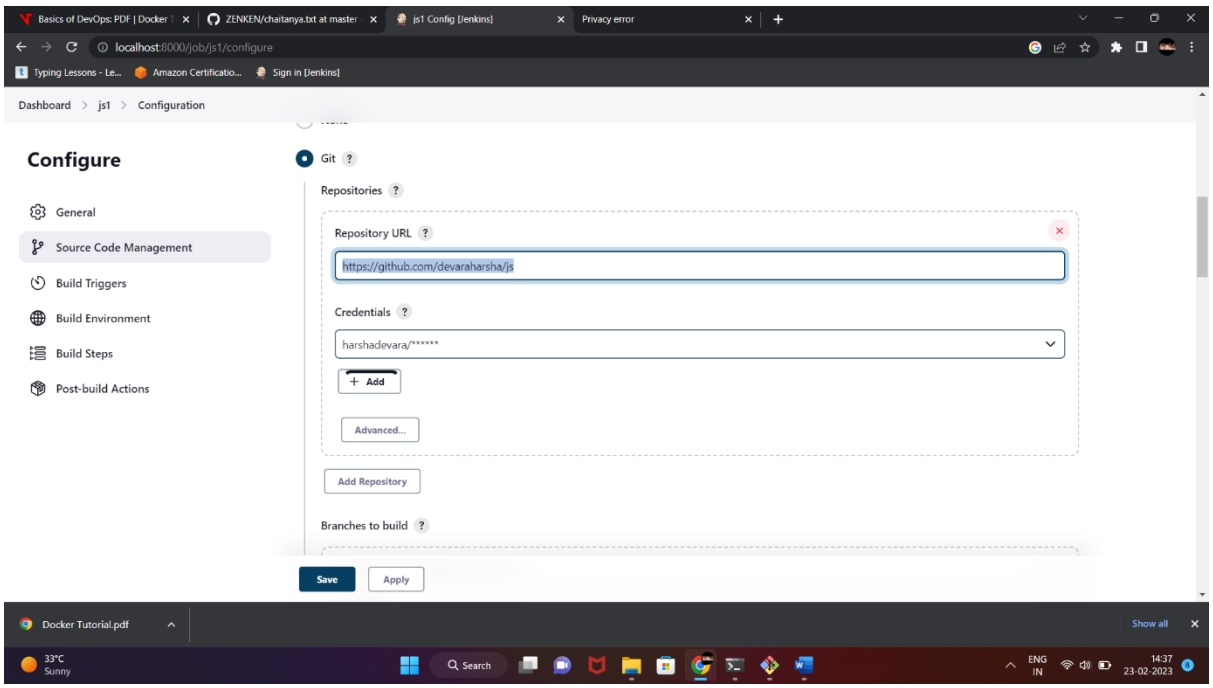
Step 5: click on save and apply.

Step 6 :click on build now, then see the console output.



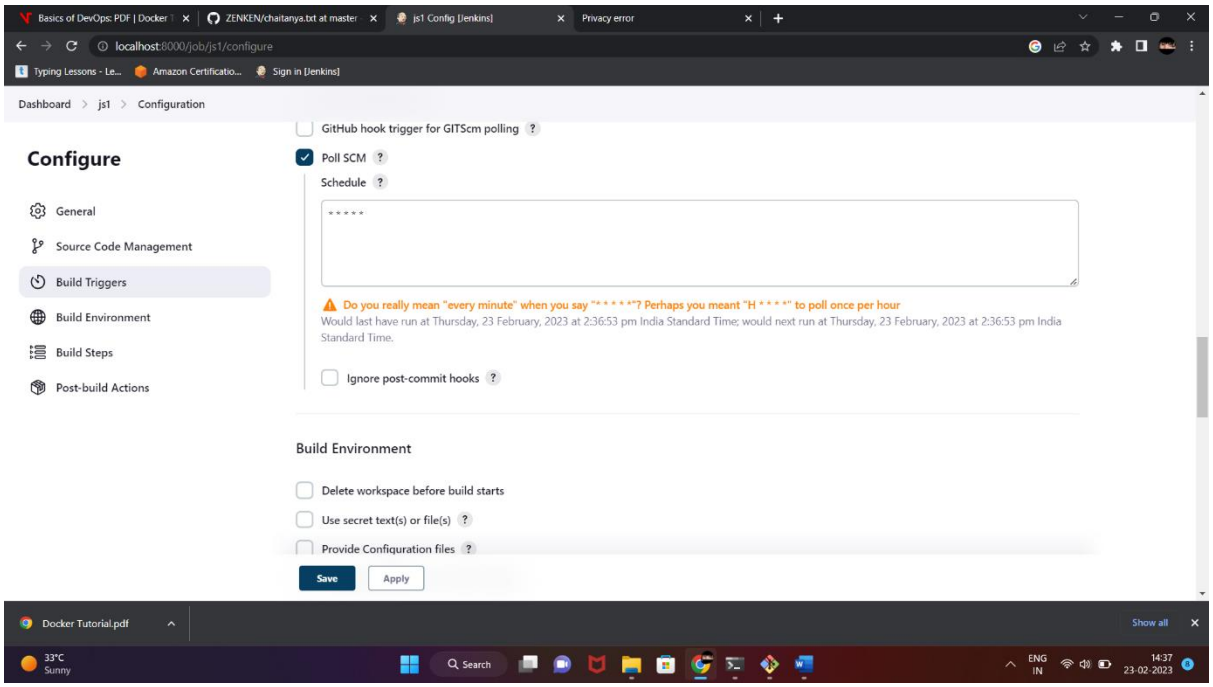
DEVARA HARSHA
20P31A04J4

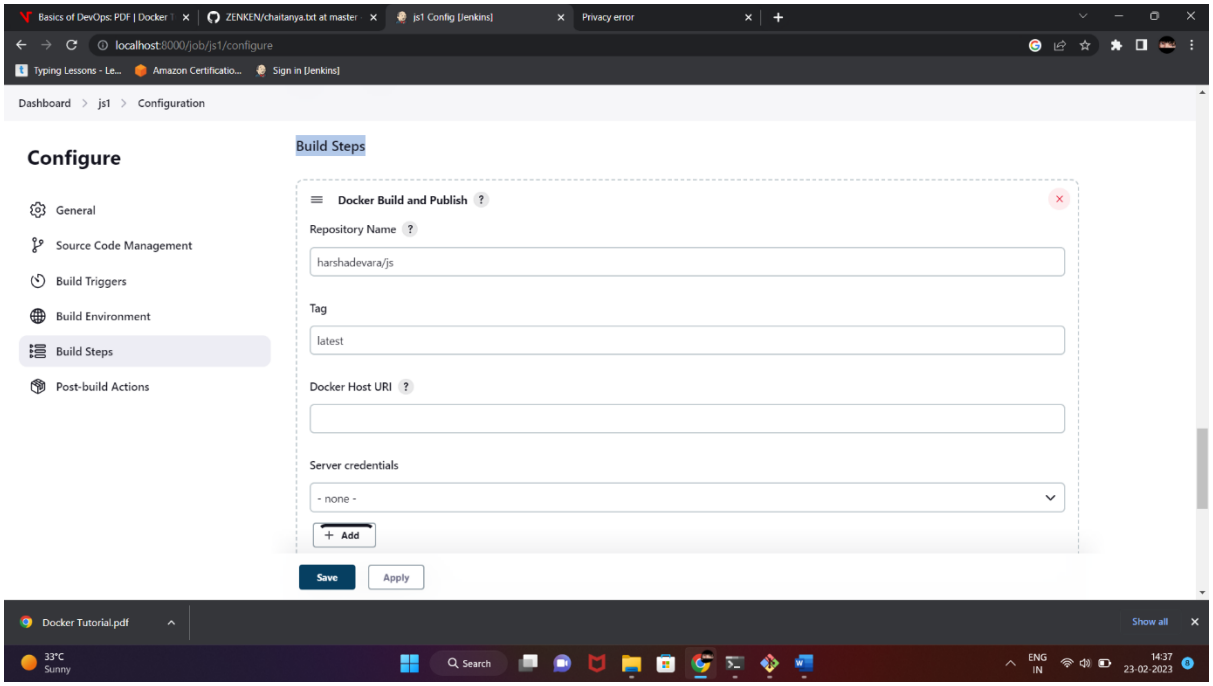




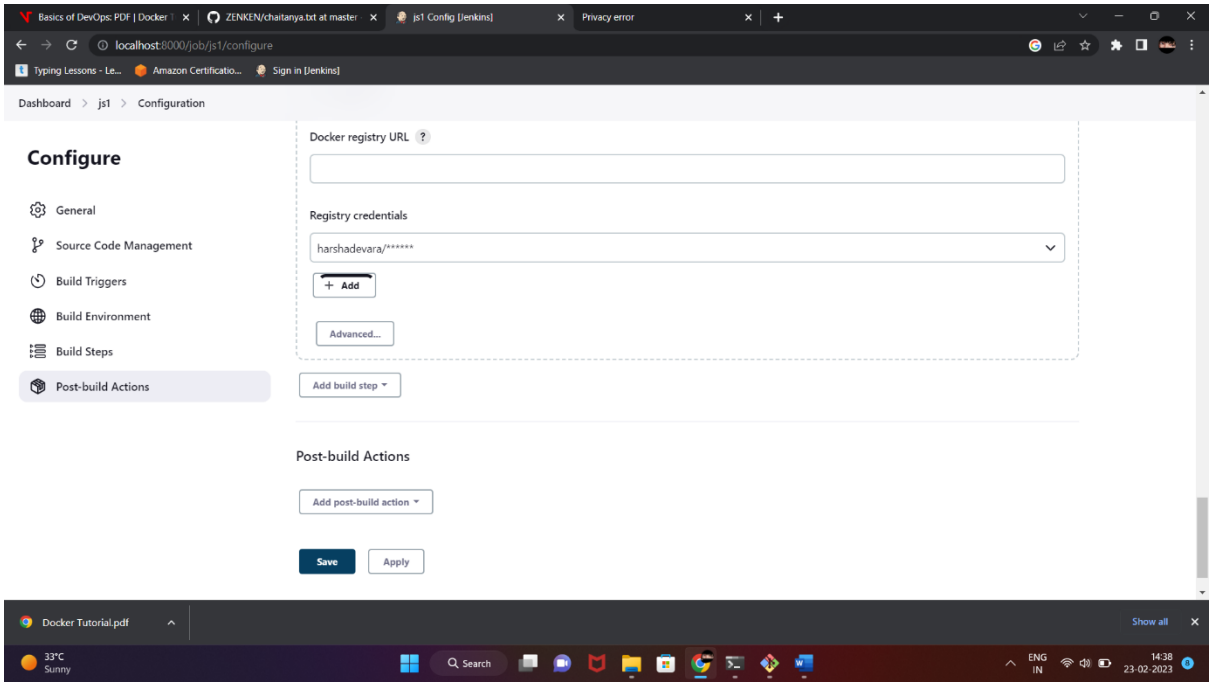
DevOps Graded Assignment-3

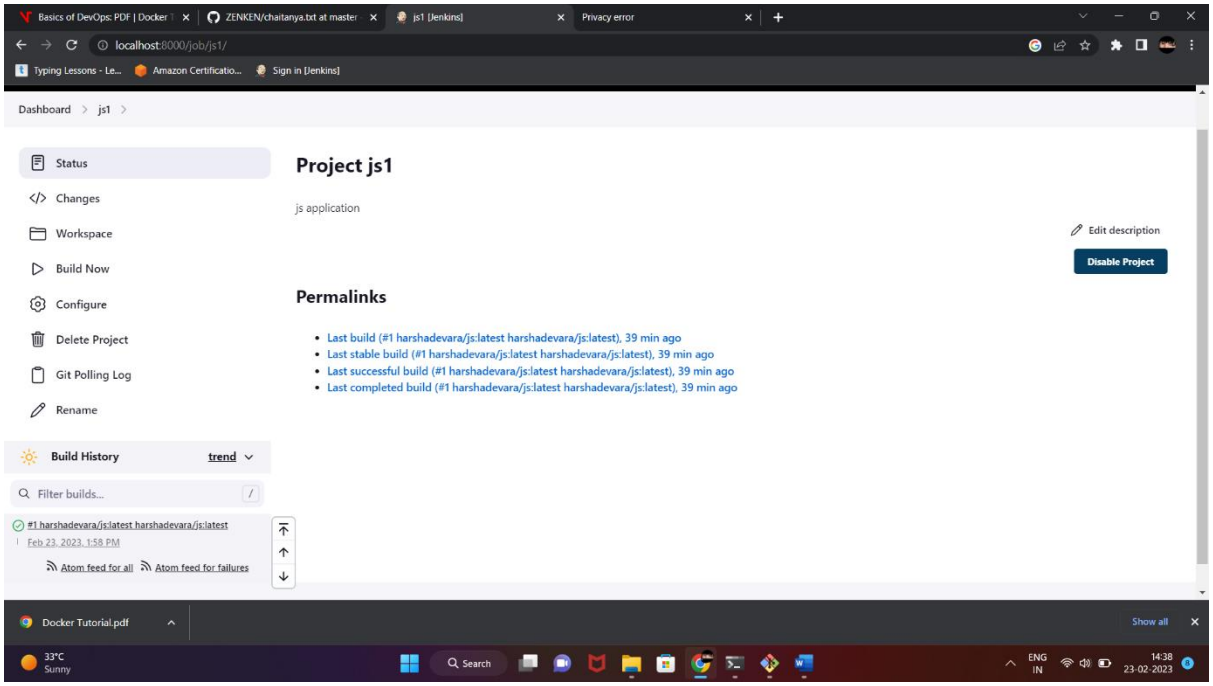
DEVARA HARSHA
20P31A04J4





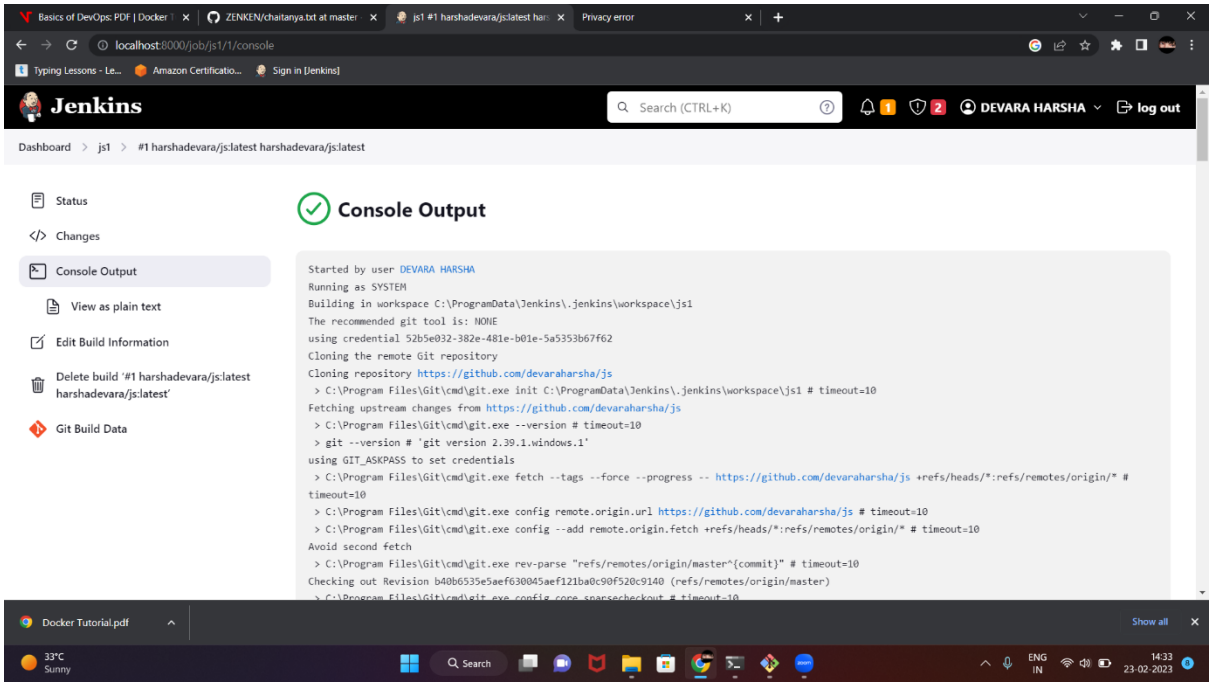
DEVARA HARSHA
20P31A04J4

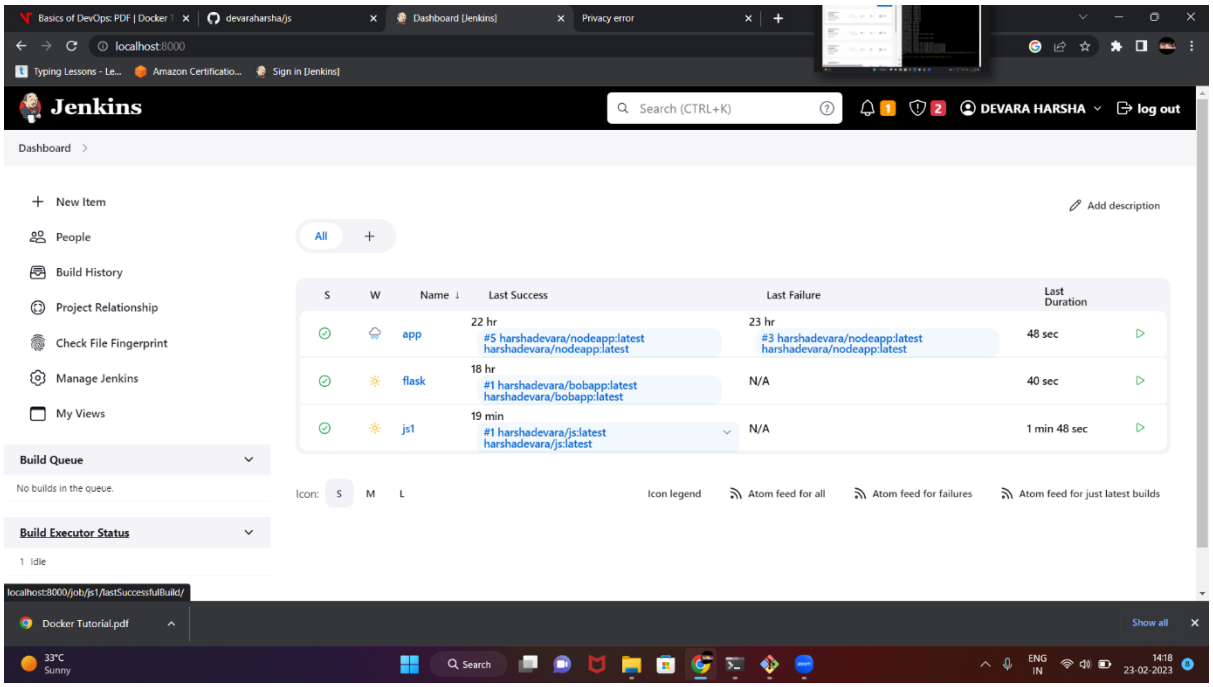




DevOps Graded Assignment-3

DEVARA HARSHA
20P31A04J4





DEVARA HARSHA
20P31A04J4

Q3. Create documentation of how you are going to create a CI-CD pipeline for python applications.
In the documentation you have mentioned each step which should be taken to configure the CI-CD pipeline for a python application including the plugins you are using and the global tool configuration.

Creating a CI/CD pipeline for Python application typically involves the following steps:

Version Control System: First, you need to have your code in a version control system (VCS) such as Git. This allows you to track changes to your code and collaborate with others.

Automated Testing: You should create a suite of automated tests to ensure that your code is working as expected. You can use testing frameworks such as pytest or unittest to write these tests.

Continuous Integration: Next, you can set up a continuous integration (CI) server, such as Jenkins, to automatically build and test your code whenever changes are pushed to the VCS. The CI server should also report any failures in the tests to the developers.

Here are the more detailed steps:

Step 1: Version Control System (VCS)

1.1. Create a repository in GitHub, Bitbucket or GitLab.

1.2. Clone the repository to your local machine.

1.3. Write your Python code in the cloned directory.

Step 2: Automated Testing

2.1. Install testing libraries, such as pytest or unittest.

2.2. Write unit tests and integration tests for your Python code.

2.3. Run the tests locally to ensure they all pass before committing any changes.

2.4. As well as same applications files push into the docker hub as image.

DEVARA HARSHA
20P31A04J4

Step 3: Continuous Integration

3.1. Sign to the Jenkins with your existing credentials.

3.2. Now go to the Jenkins Dashboard.

3.3. Go to the manage Jenkins install plugins in the available plugins like pyhton3,git,docker etc..

3.4. create a new item(pythonflask) and ok ,apply suitable configure settings based on your application.

3.5. In the source code management give your git hub credentials for accessing your whole application files.

3.6. Now the build triggers having a Poll scm then giva a five stars like this * * * * *.

3.7. In the build steps give your docker hub credentials(registry).

3.8. click on save and apply.

3.9. click on build now, then see the console output.