

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

Python Flask

```
MINGW64/c/Users/DEVARA HARSHA/OneDrive/Desktop/py/pythonflask
DEVARA HARSHA@DESKTOP-PC99 MINGW64 ~/OneDrive/Desktop/py/pythonflask (master)
$ docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/

DEVARA HARSHA@DESKTOP-PC99 MINGW64 ~/OneDrive/Desktop/py/pythonflask (master)
$ docker build -t flask .
#1 [internal] load build definition from Dockerfile
#1 sha256:202387729eb27f15ed1f304ce0c99771547eb526db68fef0df37f1bc209c3
#1 transferring dockerfile: 2748 0.0s done
#1 DONE 0.1s

#2 [internal] load dockerignore
#2 sha256:eb928d102b8a1d4c7db21a3347542ef6092aed546dcce4019026b32f232b9df1
#2 transferring context: 28 done
#2 DONE 0.0s

#3 resolve image config for docker.io/docker/dockerfile:1
#3 sha256:ac072d52190122eeef550f5228287f196e1b0247844be9ceblccc1eac391d
#3 ...

#4 [auth] docker/dockerfile:pull token for registry-1.docker.io
#4 sha256:ebb219382ca7902b420a52a6083bb8b2fd0f569264476c7c69ace630eaf3693
#4 DONE 0.0s

#5 resolve image config for docker.io/docker/dockerfile:1
#5 sha256:ac072d52190122eeef550f5228287f196e1b0247844be9ceblccc1eac391d
#5 DONE 2.4s

#6 docker-image://docker.io/docker/dockerfile:1bsha256:29b85bfa7536a5fceb7372a0817649ecb2724562a38360f4d6a7782a409b14
#6 sha256:57c85e929db09183df396146f0d627479bb510ec8aa151663925f38dad12fe9
#6 CACHED

#6 [internal] load dockerignore
#6 sha256:8b16f440ccd38ccd1cab94c29fa050d775b1d37a4c8e76432d63747a8425d5
#6 DONE 0.0s

#7 [internal] load build definition from Dockerfile
#7 sha256:17c7df603c21dc3bdad780bf47944a4f26fc64f853f9994ae1d78b0419e8f281
#7 DONE 0.0s

#8 [internal] load metadata for docker.io/library/python:3.8-slim-buster
#8 sha256:as2ddf0a3c3ab3f4e2ebc7582cec39f26df7d1ae41d54f70ea9fe596d7b25c7
#8 ...

#9 [auth] library/python:pull token for registry-1.docker.io
#9 sha256:cd20a2462c8974242cf4d109dd6b86c6e2002ea226a5b4fb6d3ba90e73c3
#9 DONE 0.0s

#8 [internal] load metadata for docker.io/library/python:3.8-slim-buster
#8 sha256:as2ddf0a3c3ab3f4e2ebc7582cec39f26df7d1ae41d54f70ea9fe596d7b25c7
#8 DONE 2.8s

#10 [1/5] FROM docker.io/library/python:3.8-slim-buster@sha256:b6625da729591bb6a922bcef903668dd977d353492e53f9e9d399b9f41be32b
```

```
MINGW64/c/Users/DEVARA HARSHA/OneDrive/Desktop/py/pythonflask
#11 DONE 0.1s

#12 [internal] load build context
#12 sha256:4053a69d5fb5a078b355464189ce1f3097a4e9b83261f36d8eb6c7c30c604eb
#12 transferring context: 8.72MB 2.3s
#12 transferring context: 28.78MB 7.1s done
#12 DONE 7.1s

#13 [3/5] COPY requirements.txt requirements.txt
#13 sha256:c18968fcd226b145a33af0ca0db6e8e279a0f696791bf6b2c0067e950b2a813
#13 DONE 0.1s

#14 [4/5] RUN pip3 install -r requirements.txt
#14 sha256:82c7a51f8f56199f933362dec5506502585ab3269908bb86dd089b0d53b09f45
#14 3.458 collecting click==8.0.3
#14 3.843 Downloading click-8.0.3-py3-none-any.whl (97 kB)
#14 3.916 ----- 97.5/97.5 KB 1.6 MB/s eta 0:00:00
#14 4.070 Downloading Flask==2.0.2-py3-none-any.whl (95 kB)
#14 4.118 ----- 95.2/95.2 KB 1.8 MB/s eta 0:00:00
#14 4.175 ----- 133.8/133.8 KB 2.9 MB/s eta 0:00:00
#14 4.290 collecting itsdangerous==2.0.1
#14 4.346 Downloading itsdangerous-2.0.1-py3-none-any.whl (18 kB)
#14 4.476 collecting Jinja2==3.0.2
#14 4.524 Downloading Jinja2-3.0.2-py3-none-any.whl (133 kB)
#14 4.575 ----- 288.9/288.9 KB 4.2 MB/s eta 0:00:00
#14 4.846 collecting MarkupSafe==2.0.1
#14 5.017 Downloading MarkupSafe-2.0.1-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux2_12_x86_64.manylinux2010_x86_64.whl (30 kB)
#14 5.153 collecting Werkzeug==2.0.2
#14 5.200 Downloading Werkzeug-2.0.2-py3-none-any.whl (288 kB)
#14 5.279 ----- 288.9/288.9 KB 4.2 MB/s eta 0:00:00
#14 5.419 Installing collected packages: Werkzeug, MarkupSafe, itsdangerous, click, Jinja2, Flask
#14 6.087 Successfully installed Flask-2.0.2 Jinja2-3.0.2 MarkupSafe-2.0.1 Werkzeug-2.0.2 click-8.0.3 itsdangerous-2.0.1
#14 6.088 WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv.
#14 6.379 WARNING: You are using pip version 22.0.4; however, version 23.0.1 is available.
#14 6.379 You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
#14 DONE 6.6s

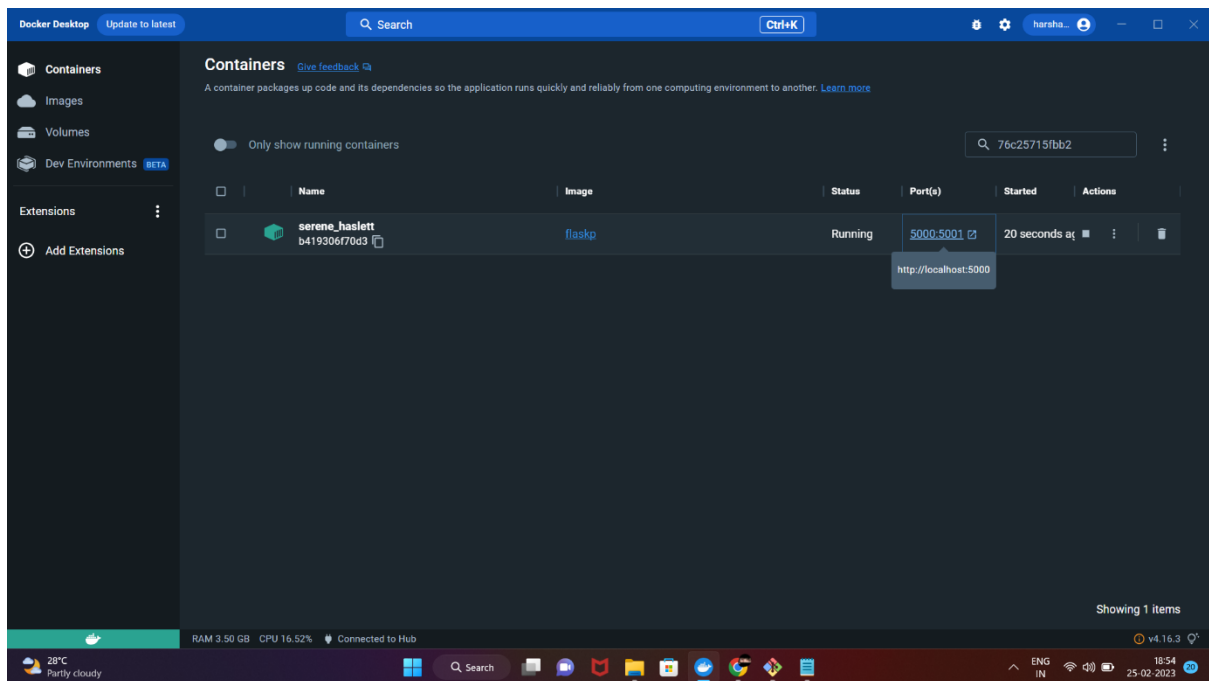
#15 [5/5] COPY
#15 sha256:9920606b34c0918dde885dc5b407f85e5b74b5e967f15dc677fba0fad5c2795
#15 DONE 0.4s

#16 exporting to image
#16 sha256:e8c613e0b6b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#16 exporting layers
#16 exporting layers 0.4s done
#16 writing image sha256:76c25715fbb22e3214a0a171712cb35a5309e39351e8f9d683c2d8c1c46cbc done
#16 naming to docker.io/library/flask done
#16 DONE 0.5s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

DEVARA HARSHA@DESKTOP-PC99 MINGW64 ~/OneDrive/Desktop/py/pythonflask (master)
$ docker run --rm --p 5000:5000 flask
0419306f70d31c0f3043cb8108557858705c140626f240f22151de9e83ecbe8

DEVARA HARSHA@DESKTOP-PC99 MINGW64 ~/OneDrive/Desktop/py/pythonflask (master)
$
```



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.

```
MINGW64/c:/Users/DEVARA HARSHA/OneDrive/Desktop/nodejs
DEVARA HARSHA@PC99: ~/OneDrive/Desktop/nodejs$ docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn
more at https://docs.docker.com/go/access-tokens/

DEVARA HARSHA@PC99: ~/OneDrive/Desktop/nodejs$ docker build -t c5-mod
#1 [internal] load build definition from Dockerfile
#1 sha256:8fee07195acc2bb7e2ba1c1db92d31a856e94a5a283b0c08d93c82834c730
#1 transferring Dockerfile: 4288 B [0.0s done]
#1 DONE 0.1s
#2 [internal] load .dockerignore
#2 sha256:72ed8329b5c0d7066e58eb6e3a5df1ed0cc571dd16202a5aa8a5d79b96c6004
#2 transferring context: 2B [0.0s done]
#2 DONE 0.0s
#3 [internal] load metadata for docker.io/library/node:16
#3 sha256:fcebe595ccabc9db4d247733f2265fb18ca302607dd9a9e445f4ebdf1189a
#3 ...
#4 [auth] library/node:pull token for registry-1.docker.io
#4 sha256:a38f1dd68a4e423d7d8225dbabf308488cbfff4d1dec902e203b16f581dcbf55
#4 DONE 0.0s
#5 [internal] load metadata for docker.io/library/node:16
#5 sha256:fcebe595ccabc9db4d247733f2265fb18ca302607dd9a9e445f4ebdf1189a
#5 DONE 2.4s
#6 [1/5] FROM docker.io/library/node:16@sha256:a4baed9809deba446900d6e2bc6b92c3f
e6ccff9ca2873f8d4e263595b02533
#6 sha256:8e04724188c8edba472ce643b51de33bb1a51d31b1a125a4be8974510d19b20b
#6 DONE 0.0s
#7 [internal] load build context
#7 sha256:556a07bce0dc8bf1529dde724876ef58d54b57ae7729b02a3ec8d503b799f63
#7 transferring context: 28.01k B [0.1s done]
#7 DONE 0.1s
#8 [2/5] WORKDIR /usr/src/app
#8 sha256:762be2d1f5cc6d6eab6e0aff00c82e2644aeea22ac24677dfc0d15fbedf5ea
#8 CACHED
#9 [3/5] COPY package*.json ./
#9 sha256:a2610cbef28e2c3019f05cbf2c5031845cb78b41e7833604c02f901358651b6
#9 CACHED
#10 [4/5] RUN npm install
#10 sha256:65f5fa47b4f4153c433ae4a8782e8080390c5a0eae77461148f41901a5cf024
#10 CACHED
#11 [5/5] COPY . .
#11 sha256:1b0e4e49d148d4230dfdd3302b39f8938cc11720c284e482e38aa12e3cffe77
#11 DONE 0.2s
#12 exporting to image
#12 sha256:e8c613e07b0b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#12 exporting layers 0.1s done
#12 writing image sha256:3db03401c23f063f1319ecd7922d43e85d63633475409b7cb93644
6b61e8255 done
#12 naming to docker.io/library/node done
#12 DONE 0.1s

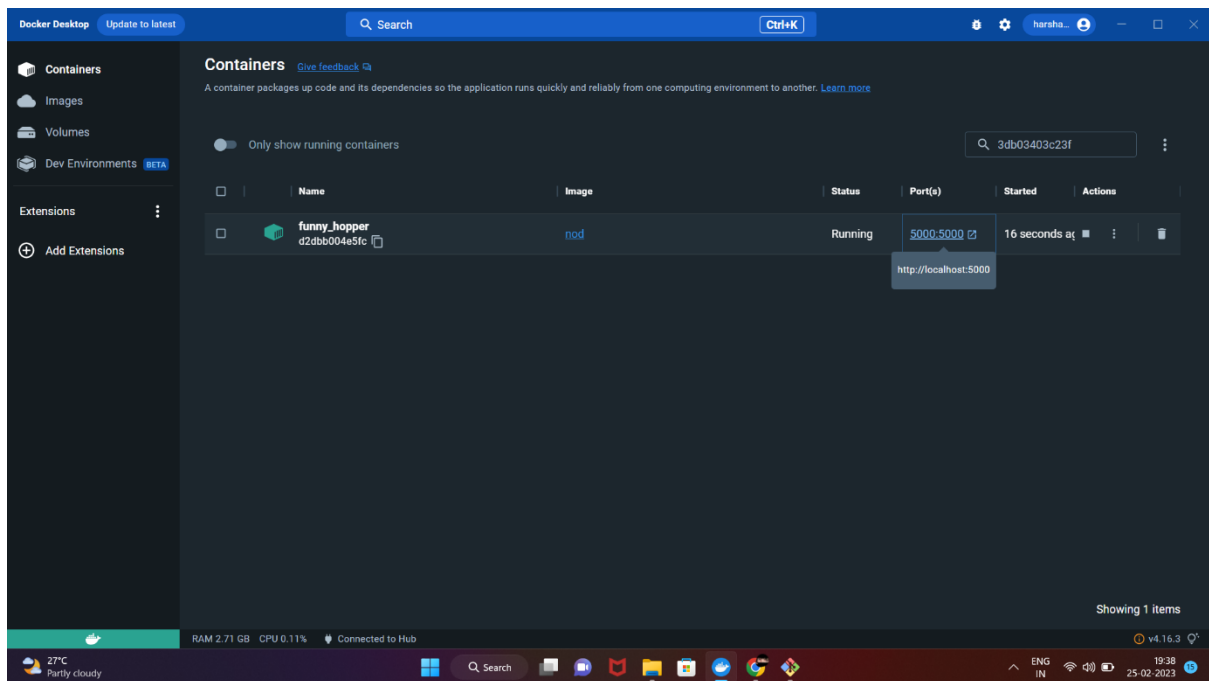
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and l
earn how to fix them

DEVARA HARSHA@PC99: ~/OneDrive/Desktop/nodejs$ docker build --push -t 20p31a04j4
d2dbb004e5f9c6883ecbd07a3beb3a23bde3dbb1bad97f1ea90e627de3d83170
DEVARA HARSHA@PC99: ~/OneDrive/Desktop/nodejs$
```

```
MINGW64/c:/Users/DEVARA HARSHA/OneDrive/Desktop/nodejs
#2 transferring context: 2B [0.0s done]
#2 DONE 0.0s
#3 [internal] load metadata for docker.io/library/node:16
#3 sha256:fcebe595ccabc9db4d247733f2265fb18ca302607dd9a9e445f4ebdf1189a
#3 ...
#4 [auth] library/node:pull token for registry-1.docker.io
#4 sha256:a38f1dd68a4e423d7d8225dbabf308488cbfff4d1dec902e203b16f581dcbf55
#4 DONE 0.0s
#5 [internal] load metadata for docker.io/library/node:16
#5 sha256:fcebe595ccabc9db4d247733f2265fb18ca302607dd9a9e445f4ebdf1189a
#5 DONE 2.4s
#6 [1/5] FROM docker.io/library/node:16@sha256:a4baed9809deba446900d6e2bc6b92c3f
e6ccff9ca2873f8d4e263595b02533
#6 sha256:8e04724188c8edba472ce643b51de33bb1a51d31b1a125a4be8974510d19b20b
#6 DONE 0.0s
#7 [internal] load build context
#7 sha256:556a07bce0dc8bf1529dde724876ef58d54b57ae7729b02a3ec8d503b799f63
#7 transferring context: 28.01k B [0.1s done]
#7 DONE 0.1s
#8 [2/5] WORKDIR /usr/src/app
#8 sha256:762be2d1f5cc6d6eab6e0aff00c82e2644aeea22ac24677dfc0d15fbedf5ea
#8 CACHED
#9 [3/5] COPY package*.json ./
#9 sha256:a2610cbef28e2c3019f05cbf2c5031845cb78b41e7833604c02f901358651b6
#9 CACHED
#10 [4/5] RUN npm install
#10 sha256:65f5fa47b4f4153c433ae4a8782e8080390c5a0eae77461148f41901a5cf024
#10 CACHED
#11 [5/5] COPY . .
#11 sha256:1b0e4e49d148d4230dfdd3302b39f8938cc11720c284e482e38aa12e3cffe77
#11 DONE 0.2s
#12 exporting to image
#12 sha256:e8c613e07b0b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#12 exporting layers 0.1s done
#12 writing image sha256:3db03401c23f063f1319ecd7922d43e85d63633475409b7cb93644
6b61e8255 done
#12 naming to docker.io/library/node done
#12 DONE 0.1s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and l
earn how to fix them

DEVARA HARSHA@PC99: ~/OneDrive/Desktop/nodejs$ docker build --push -t 20p31a04j4
d2dbb004e5f9c6883ecbd07a3beb3a23bde3dbb1bad97f1ea90e627de3d83170
DEVARA HARSHA@PC99: ~/OneDrive/Desktop/nodejs$
```



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

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.

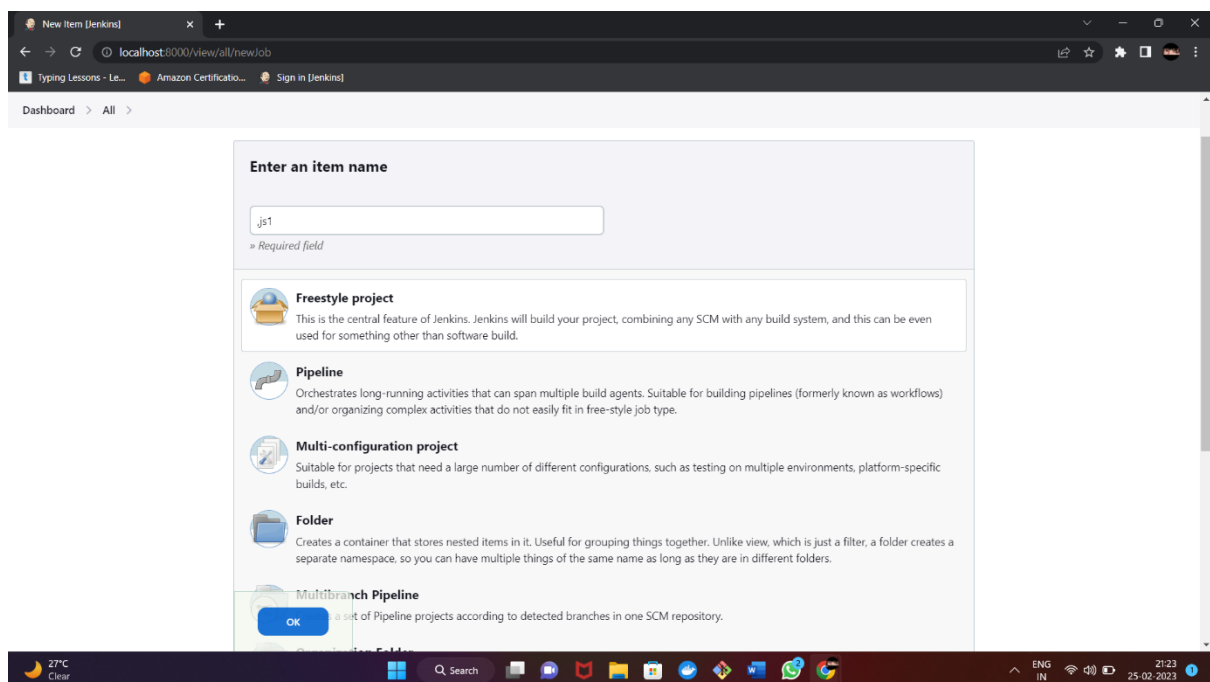
Step 2 : Create a docker image for that, where the docker file is present and push into the docker hub.

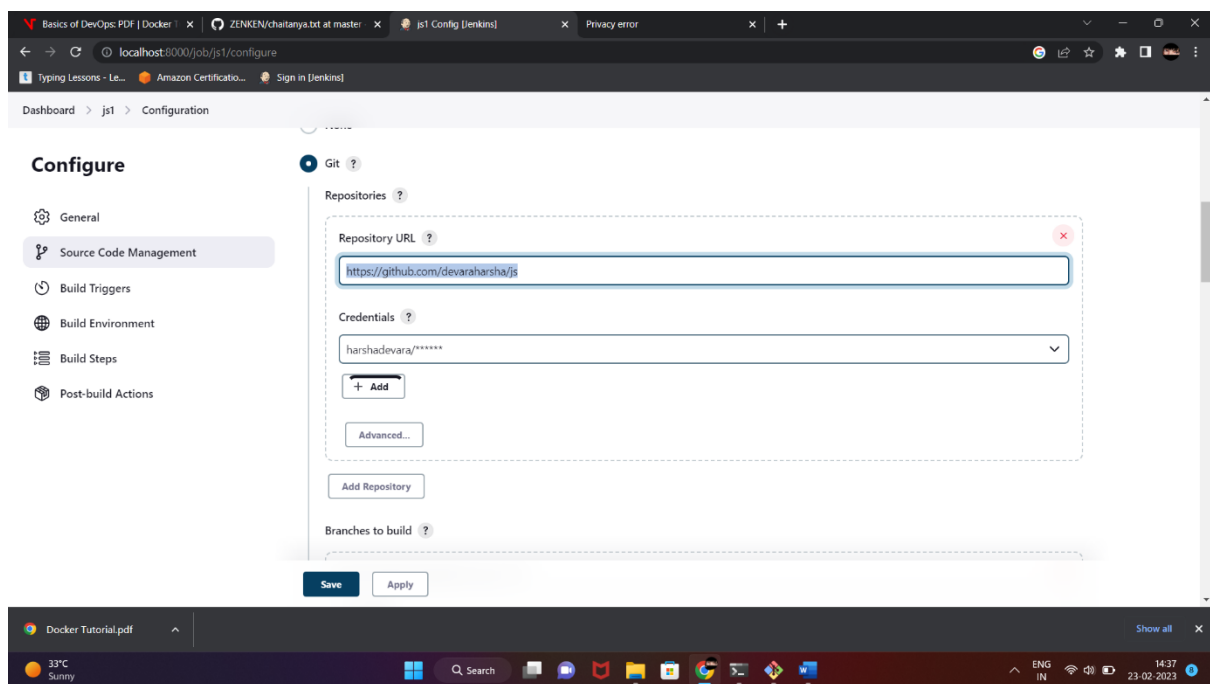
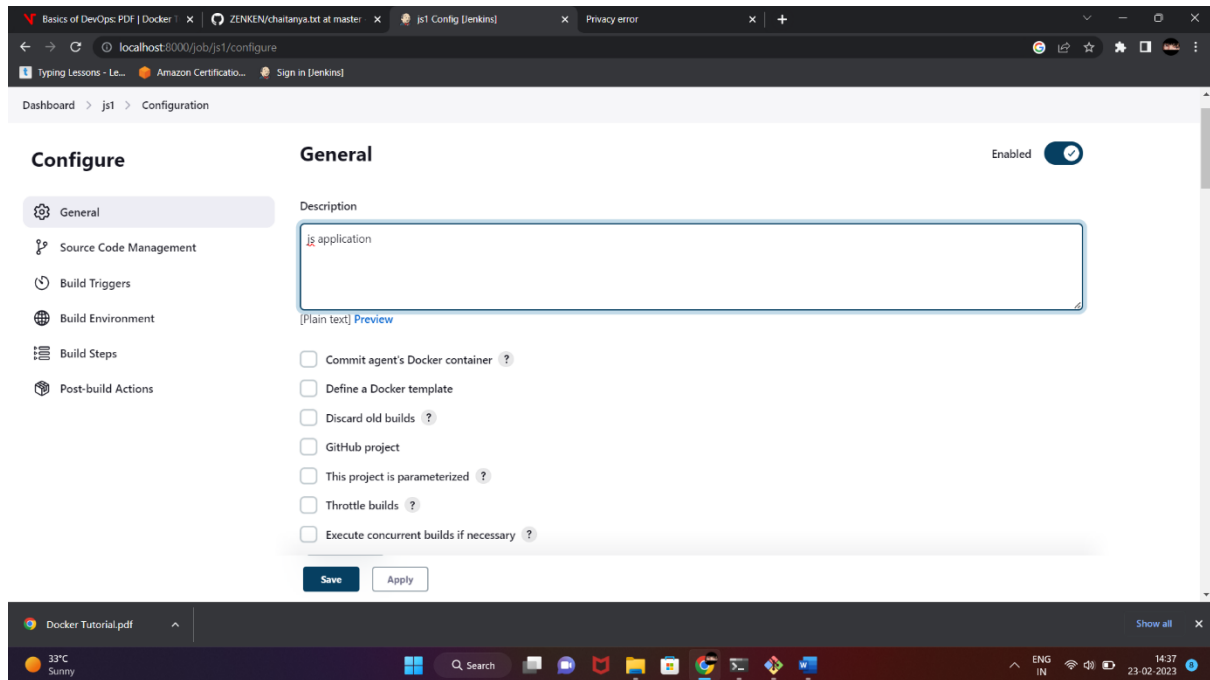
Step 3 : Now you can go to the Jenkins dashboard.

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

Step 5: click on save and apply.

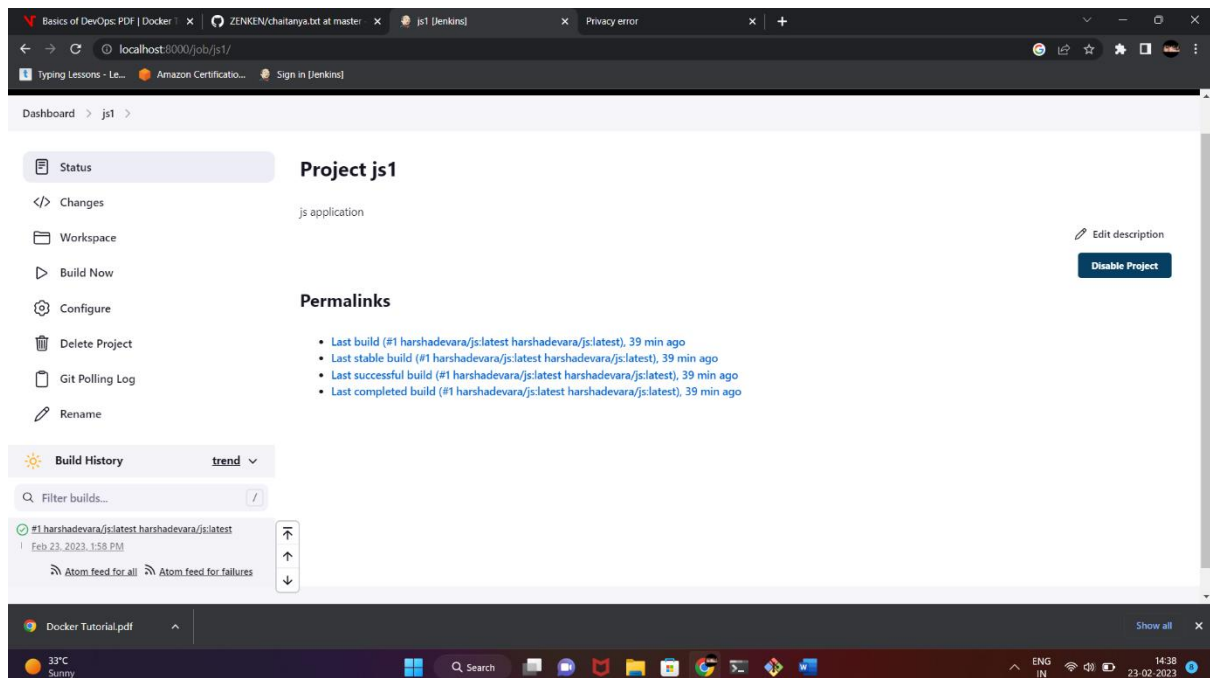
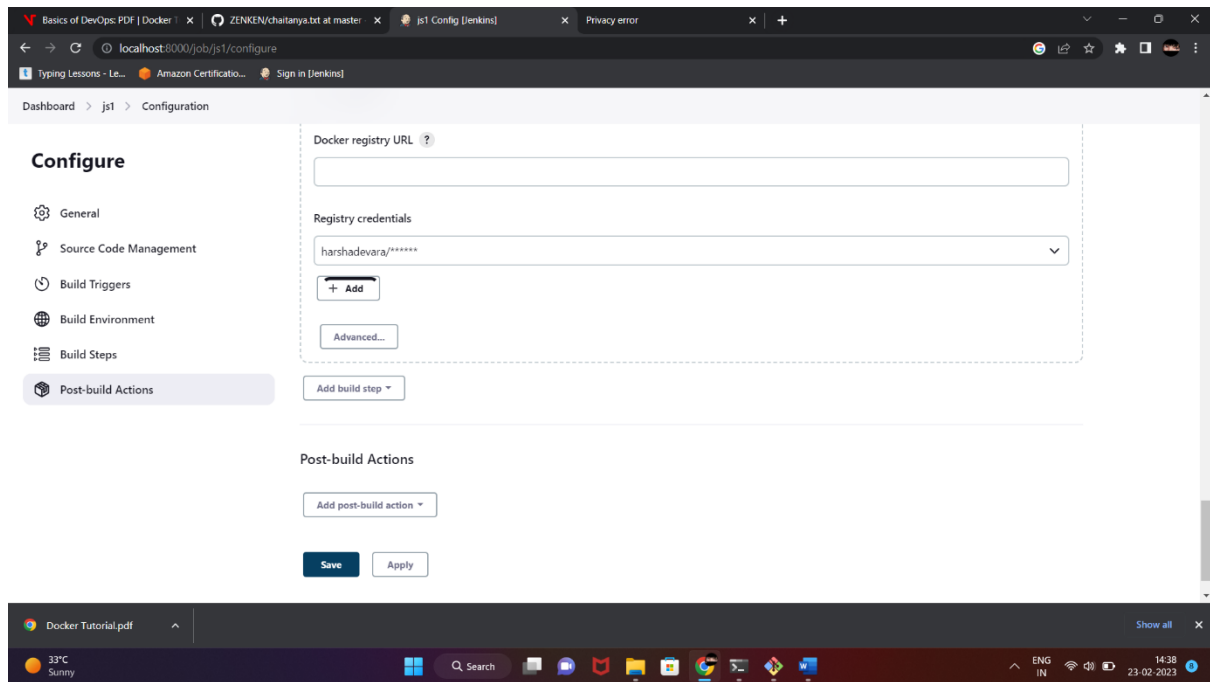
Step 6 :click on build now, then see the console output.





The screenshot shows the Jenkins Configuration page for a job named 'js1'. The 'Build Triggers' tab is selected in the left sidebar. The 'Poll SCM' checkbox is checked. The 'Schedule' field contains '*****'. A warning message states: 'Do you really mean "every minute" when you say "*****"? Perhaps you meant "H * * * *"' to poll once per hour. Below the schedule field, it says 'Would last have run at Thursday, 23 February, 2023 at 2:36:53 pm India Standard Time; would next run at Thursday, 23 February, 2023 at 2:36:53 pm India Standard Time.' The 'Ignore post-commit hooks' checkbox is unchecked. The 'Build Environment' section has three unchecked options: 'Delete workspace before build starts', 'Use secret text(s) or file(s)', and 'Provide Configuration files'. 'Save' and 'Apply' buttons are at the bottom.

The screenshot shows the Jenkins Configuration page for the same 'js1' job, but with the 'Build Steps' tab selected. A new build step named 'Docker Build and Publish' has been added. The 'Repository Name' field contains 'harshadevara/js'. The 'Tag' field contains 'latest'. The 'Docker Host URI' field is empty. The 'Server credentials' dropdown menu is set to '- none -'. There is an '+ Add' button below the dropdown. 'Save' and 'Apply' buttons are at the bottom.



The screenshot shows the Jenkins web interface for build #1 of job 'js1'. The 'Console Output' tab is selected, displaying the following log:

```
Started by user: DEVARA HARSHA
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\workspace\js1
The recommended git tool is: NONE
using credential 52b5e032-382e-481e-b01e-5a5353b67f62
Cloning the remote Git repository
Cloning repository https://github.com/devaraharsha/js
> C:\Program Files\Git\cmd\git.exe init C:\ProgramData\Jenkins\workspace\js1 # timeout=10
Fetching upstream changes from https://github.com/devaraharsha/js
> C:\Program Files\Git\cmd\git.exe --version # timeout=10
> git --version # 'git version 2.39.1.windows.1'
using GIT_ASKPASS to set credentials
> C:\Program Files\Git\cmd\git.exe fetch --tags --force --progress -- https://github.com/devaraharsha/js +refs/heads/*:refs/remotes/origin/* # timeout=10
> C:\Program Files\Git\cmd\git.exe config remote.origin.url https://github.com/devaraharsha/js # timeout=10
> C:\Program Files\Git\cmd\git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> C:\Program Files\Git\cmd\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision b40b6535e5aef630045aef121ba0c90f920c9140 (refs/remotes/origin/master)
> C:\Program Files\Git\cmd\git.exe config core.sparsecheckout # timeout=10
```

The screenshot shows the Jenkins Dashboard for job 'js1'. The 'Build History' tab is selected, displaying a table of recent builds:

S	W	Name	Last Success	Last Failure	Last Duration
✓	☁	app	22 hr #5 harshadevara/nodeapp:latest harshadevara/nodeapp:latest	23 hr #3 harshadevara/nodeapp:latest harshadevara/nodeapp:latest	48 sec
✓	☀	flask	18 hr #1 harshadevara/bobapp:latest harshadevara/bobapp:latest	N/A	40 sec
✓	☀	js1	19 min #1 harshadevara/js:latest harshadevara/js:latest	N/A	1 min 48 sec

Below the table, there is a section for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle). The bottom of the screenshot shows the Windows taskbar with the date 23-02-2023 and time 14:18.

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.

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 python3,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 give 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.