

ASSIGNMENT -3

Q1.CREATION OF DOCKER CONTAINER AND IMAGE:

*PYTHON FLASK:

- Open the folder of python flask and go to the command prompt.
- Login to the docker desktop and dockerhub.com in the background.
- In command prompt enter the command docker login, then it will authenticate with our credentials.
- Now, enter “docker images” it will list all our images like python,ubuntu etc. if we have any images otherwise, there will be no images.
- Inorder to build our own image,enter“docker build -timage_name .”
- If we check docker images again then it will show the image which we have created before.
- Create a repository in dockerhub.com.
- Now enter “docker tag image_name:latest username/repository_name” and push the image into our repository using “docker push username/image_name.”
- Now run the container and like docker run -d -p 5000:4000 imagename.
- Here 4000 indicates EXPOSE ->It tells about docker that our application will run on the port 4000.
- If we open the docker desktop, we can see that a container is in running state.
- Then it will show that ‘the site is not working’ then change the expose and again build the image and run the container.
- Now, it will show the output as ‘welcome to python flask tutorials’.

```
C:\Windows\System32\cmd.exe + -> C:\Windows\System32\cmd.exe
0e9a917feaab: Mounted from chaitanya8416/ac8416
637061849f21: Mounted from chaitanya8416/ac8416
af7d57f541a7: Mounted from chaitanya8416/ac8416
0fc22002fc74: Mounted from chaitanya8416/ac8416
0a68b19e97e0: Mounted from chaitanya8416/ac8416
8d60832b730a: Mounted from chaitanya8416/ac8416
63b3cf45ecce8: Mounted from chaitanya8416/ac8416
latest: digest: sha256:4ba9d9e6ab67d57ecd4e7a8931557fcfa0a0ae1a0b79ed7e61ae949470217151 size: 2206

C:\Users\chait\OneDrive\Desktop\folder\LearningGIt\pythonflask>docker run -d -p 5000:5000 acc
f484138639c56071e966a73ab7cb226e711ca084e4503dcba6633a701a096784

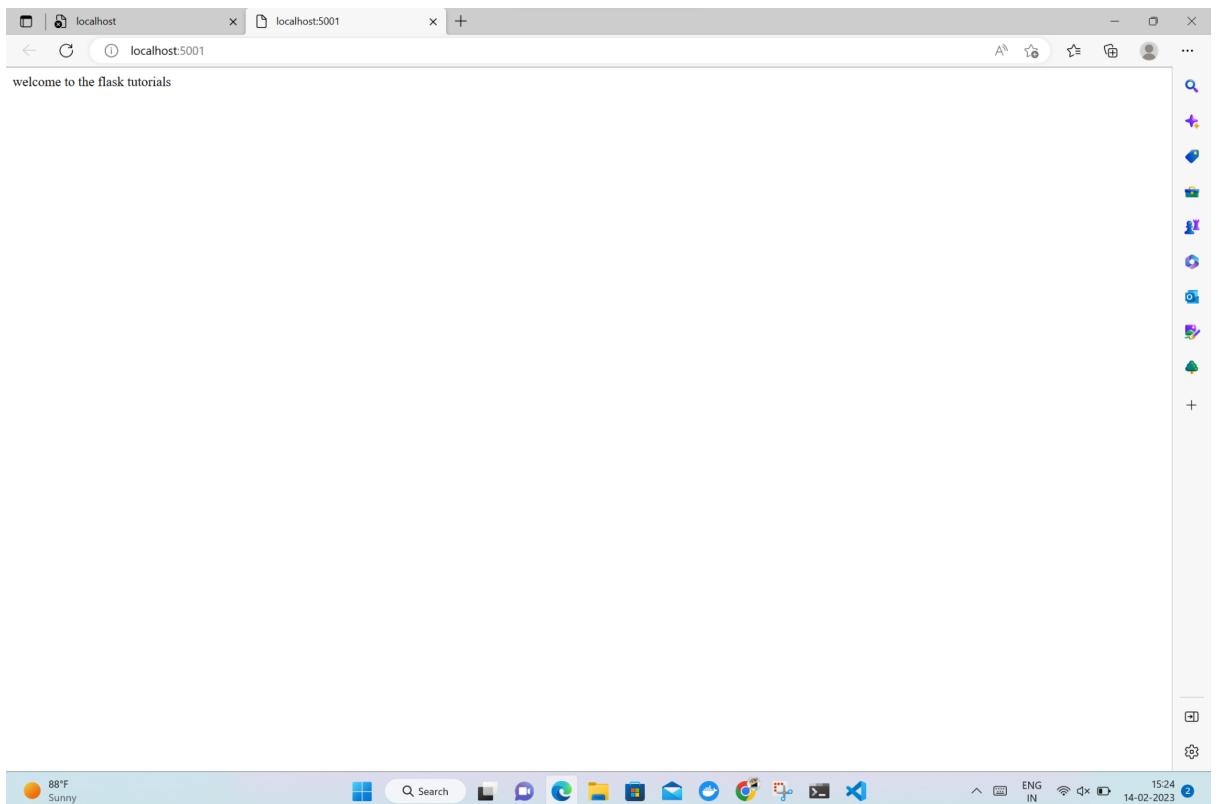
C:\Users\chait\OneDrive\Desktop\folder\LearningGIt\pythonflask>docker build -t acc .
[+] Building 5.3s (16/16) FINISHED
--> [internal] load build definition from Dockerfile 0.0s
--> => transferring dockerfile: 274B 0.0s
--> [internal] load .dockerignore 0.0s
--> => transferring context: 2B 0.0s
--> [internal] resolve image config for docker.io/docker/dockerfile:1 3.3s
--> [auth] docker/dockerfile:pull token for registry-1.docker.io 0.0s
--> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:39b85bbfa7536a5feceb7372a0817649ecb2724562a38360f4 0.0s
--> [internal] load build definition from Dockerfile 0.0s
--> [internal] load .dockerignore 0.0s
--> [internal] load metadata for docker.io/library/python:3.8-slim-buster 1.5s
--> [auth] library/python:pull token for registry-1.docker.io 0.0s
--> [1/5] FROM docker.io/library/python:3.8-slim-buster@sha256:2ce8031b678a8de21815a760313707f145f69ffc80f8d411b2 0.0s
--> [internal] load build context 0.2s
--> => transferring context: 173.87kB 0.1s
--> CACHED [2/5] WORKDIR /python-docker 0.0s
--> CACHED [3/5] COPY requirements.txt requirements.txt 0.0s
--> CACHED [4/5] RUN pip3 install -r requirements.txt 0.0s
--> CACHED [5/5] COPY . . 0.0s
--> => exporting image 0.0s
--> => exporting layers 0.0s
--> => writing image sha256:66497502bd0bc8684cafcc7733caf9af3f054a5c789b62c368e5f0a8eb07257 0.0s
--> => naming to docker.io/library/acc 0.0s

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

C:\Users\chait\OneDrive\Desktop\folder\LearningGIt\pythonflask>docker run -d -p 5001:5001 acc
f1c726d03d575002683b8e34606ebdc10f2219700c82190883b600d78a92f1d4a

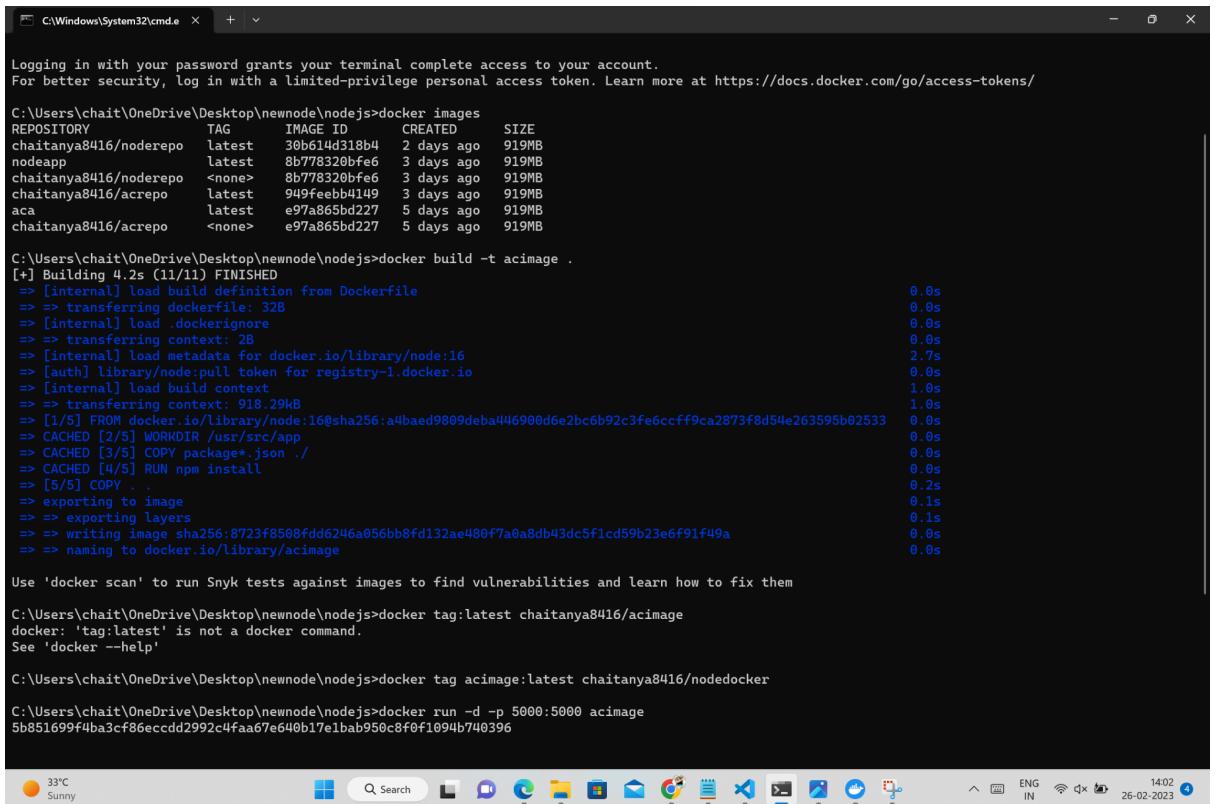
C:\Users\chait\OneDrive\Desktop\folder\LearningGIt\pythonflask>
```

OUTPUT:



NODEJS:

- Repeat the same process which we did for python flask.
- But instead of the python flask folder, open nodejs folder and enter all the commands required to run a container.



```
C:\Users\chait\OneDrive\Desktop\newnode\nodejs>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
chaitanya8416/noderepo latest 30b614d318b4 2 days ago 919MB
nodeapp latest 8b778320bfe6 3 days ago 919MB
chaitanya8416/noderepo <none> 8b778320bfe6 3 days ago 919MB
chaitanya8416/acrepo latest 949feeb4149 3 days ago 919MB
aca latest e97a865bd227 5 days ago 919MB
chaitanya8416/acrepo <none> e97a865bd227 5 days ago 919MB

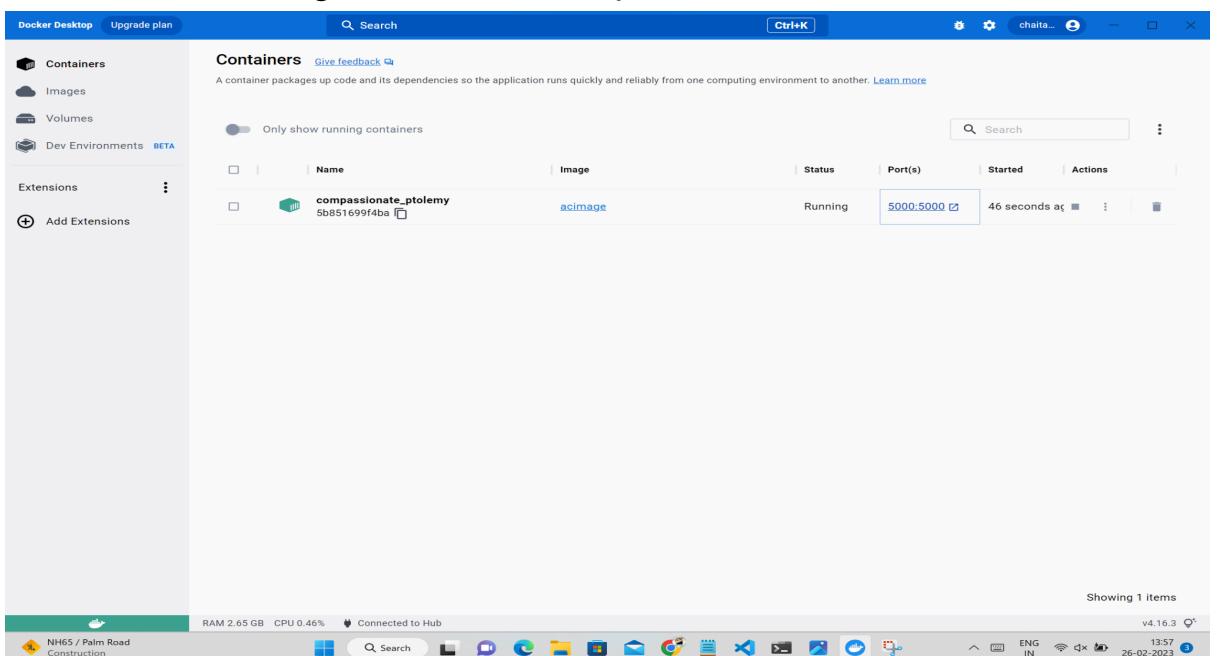
C:\Users\chait\OneDrive\Desktop\newnode\nodejs>docker build -t acimage .
[+] Building 4.2s (11/11) FINISHED
--> [internal] load build definition from Dockerfile
--> => transferring dockerfile: 32B
--> [internal] load .dockerignore
--> => transferring context: 2B
--> [internal] load metadata for docker.io/library/node:16
--> [auth] library/node:pull token for registry-1.docker.io
--> [internal] load build context
--> => transferring context: 918.29kB
--> [1/5] FROM docker.io/library/node:16@sha256:a4baed9809deba446900d6e2bc6b92c3fe6ccff9ca2873f8d54e263595b02533
--> => CACHED [2/5] WORKDIR /usr/src/app
--> => CACHED [3/5] COPY package.json .
--> => CACHED [4/5] RUN npm install
--> => [5/5] COPY .
--> => exporting to image
--> => exporting layers
--> => writing image sha256:8723f8508fdd6246a056bb8fd132ae480f7a0a8db43dc5f1cd59b23e6f91f49a
--> => naming to docker.io/library/acimage

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

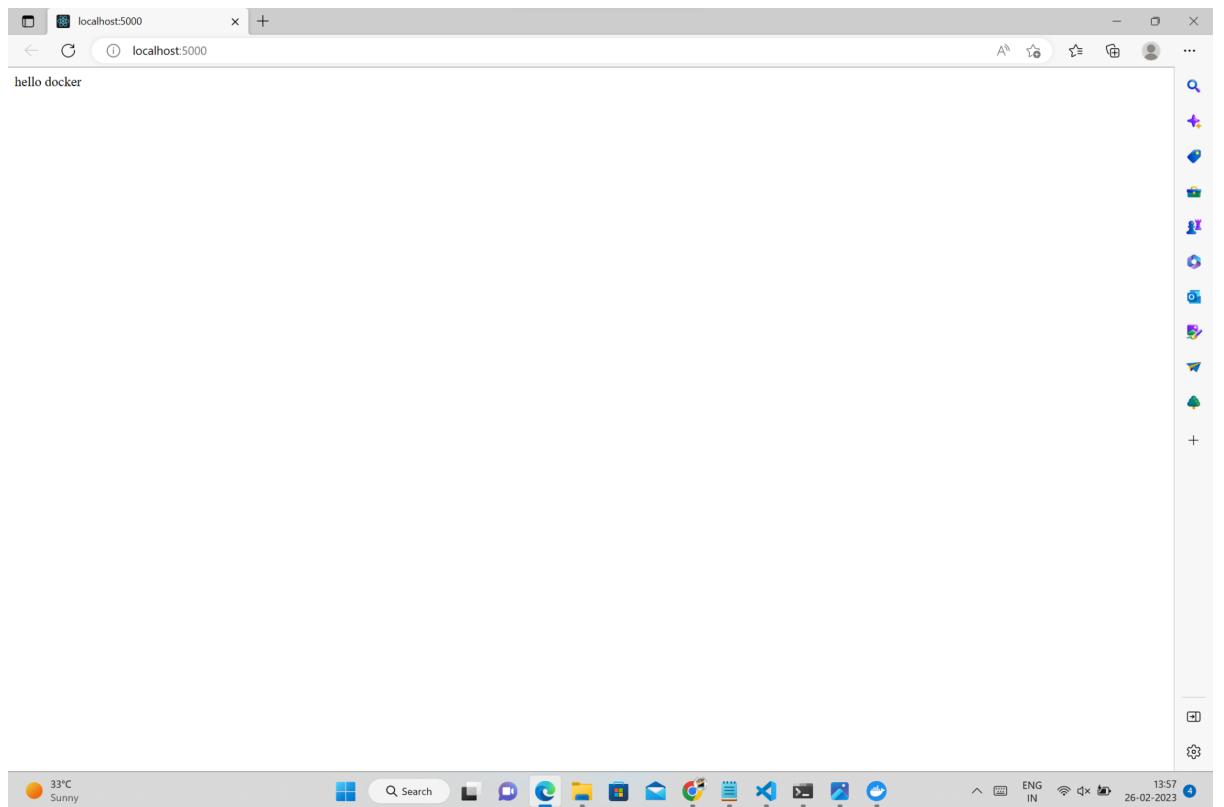
C:\Users\chait\OneDrive\Desktop\newnode\nodejs>docker tag:latest chaitanya8416/acimage
docker: 'tag:latest' is not a docker command.
See 'docker --help'

C:\Users\chait\OneDrive\Desktop\newnode\nodejs>docker tag acimage:latest chaitanya8416/nodedocker
C:\Users\chait\OneDrive\Desktop\newnode\nodejs>docker run -d -p 5000:5000 acimage
5b851699f4ba3cf86eccdd2992c4faa67e640b17ebab950cb80f1094b740396
```

Container is running in docker desktop



OUTPUT:



Q2.CI-CD PIPELINE APPLICATION USING JENKINS:

- Open nodejs folder in our local system and clear the path and open command prompt and create a repository in github.
- Now, enter the commands git init, git add . , git commit and git push origin master.
- Then all the files in nodejs folder will be pushed to our remote repository.
- Now, open docker desktop and open command prompt and enter docker login and build an image , and push that in to repository.
- We can find that in the repositories of docker hub.
- Now, open localhost:8000 here 8000 is our expose and create a new job (free style project)

Now, configure as follows:

The screenshot shows the Docker Hub interface for the repository `chaitanya8416/noderepo`. The repository has 0 private repositories. A banner at the top encourages using Wasm instead of Linux containers. The General tab is selected, showing options to add a description, push Docker commands, and view automated builds.

Description: This repository does not have a description.

Docker commands: `docker push chaitanya8416/noderepo:tagname`

Tags: The repository contains 1 tag(s). The latest tag was pushed 2 hours ago.

Automated Builds: Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

The screenshot shows the Jenkins interface for the project `node`. The `Project node` page displays the following information:

- Status**: this is a nodeapp
- Permalinks**: A list of recent builds:
 - Last build (#4 chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest), 1 hr 35 min ago
 - Last stable build (#4 chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest), 1 hr 35 min ago
 - Last successful build (#4 chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest), 1 hr 35 min ago
 - Last failed build (#1 chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest), 1 hr 46 min ago
 - Last unsuccessful build (#1 chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest), 1 hr 46 min ago
 - Last completed build (#4 chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest), 1 hr 35 min ago
- Build History**: A table showing build logs:

Build #	Log	Timestamp
#4	chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest	Feb 23, 2023 2:48 PM
#3	chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest	Feb 23, 2023 2:40 PM
#2	chaitanya8416/noderepo:latest chaitanya8416/noderepo:latest	Feb 23, 2023 2:38 PM

The screenshot shows a Jenkins console output window. The title bar says "New Jenkins CI_CD pipeline - Issue #2 chaitanya8416/noderepo:latest". The main area is titled "Console Output" with a green checkmark icon. The log output is as follows:

```

Started by user admin
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\node
The recommended git tool is: NONE
using credential c4cc1302-6ac2-40ef-8f3e-591be2148b24
> C:\Program Files\Git\cmd\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\node\.git # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\cmd\git.exe config remote.origin.url https://github.com/Neelima8416/neelima.git # timeout=10
Fetching upstream changes from https://github.com/Neelima8416/neelima.git
> 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/Neelima8416/neelima.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> C:\Program Files\Git\cmd\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 7d244bf0d3156997135b57dd5d33af22f6371698 (refs/remotes/origin/master)
> C:\Program Files\Git\cmd\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\cmd\git.exe checkout -f 7d244bf0d3156997135b57dd5d33af22f6371698 # timeout=10
Commit message: "ac"
> C:\Program Files\Git\cmd\git.exe rev-list --no-walk 7d244bf0d3156997135b57dd5d33af22f6371698 # timeout=10
[node] $ docker build -t chaitanya8416/noderepo:latest --pull=true C:\ProgramData\Jenkins\.jenkins\workspace\node
WARNING: Support for the legacy ~/.dockercfg configuration file and file-format is deprecated and will be removed in an upcoming release
WARNING: Support for the legacy ~/.dockercfg configuration file and file-format is deprecated and will be removed in an upcoming release
WARNING: Support for the legacy ~/.dockercfg configuration file and file-format is deprecated and will be removed in an upcoming release

```

Q3. STEPS TO CREATE CI-CD PIPELINE FOR PYTHON APPLICATIONS:

- Open jenkins like localhost:8000 and login with your credentials and add an item.
- Click on the freestyle project and click on OK.
- If we want to add a description we can add it in the description box.
- In source code management, enter the git repository URL which is containing the python flask application .
- If the repository is private, then add the credentials and select them otherwise select none.
- Now, in build triggers select poll SCM enter * * * * * it means that the task will be done every minute.
- In build steps, select Docker build and publish and add the docker hub repository credentials where the docker image and container for python flask is present.
- Now, save and apply and select the option build now.

- If the build is successful, then it indicates that a CI-CD pipeline is created successfully.

The screenshot shows the Jenkins configuration interface for a job named "python". The "General" section is selected. In the "Description" field, the text "this is python application" is entered. Under "Source Code Management", the "Git" option is selected, and a repository URL is listed as "https://github.com/Neelima8416/Ci-CD.git". The Jenkins status bar at the bottom indicates "26°C Mostly cloudy" and the date "26-02-2023".

This screenshot shows the same Jenkins configuration page for the "python" job, but with more detailed settings for the "Source Code Management" section. The "Repository URL" is set to "https://github.com/Neelima8416/Ci-CD.git", and the "Credentials" dropdown contains "Neelima8416/*****". The "Branches to build" field shows the branch specifier "*/*master". The Jenkins status bar at the bottom indicates "26°C Mostly cloudy" and the date "26-02-2023".

Screenshot of Jenkins configuration for a Python job.

Configure

Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?
Schedule ?

⚠️ Do you really mean "every minute" when you say "*****"? Perhaps you meant "H * * * *" to poll once per hour
Would last run have at Sunday, 26 February, 2023 at 8:59:31 pm India Standard Time; would next run at Sunday, 26 February, 2023 at 8:59:31 pm India Standard Time.
- Ignore post-commit hooks ?

Build Environment

- Delete workspace before build starts

Save **Apply**

26°C Mostly cloudy 21:01 26-02-2023

Screenshot of Jenkins configuration for a Python job, showing Docker integration.

Configure

Build Steps

Docker Build and Publish

- Repository Name ?
chaitanya8416 / pythonrepo
- Tag
latest
- Docker Host URI ?
[empty field]
- Server credentials
- none -
+ Add
- Docker registry URL ?
[empty field]
- Registry credentials
chaitanya8416/*****

Save **Apply**

26°C Mostly cloudy 21:01 26-02-2023