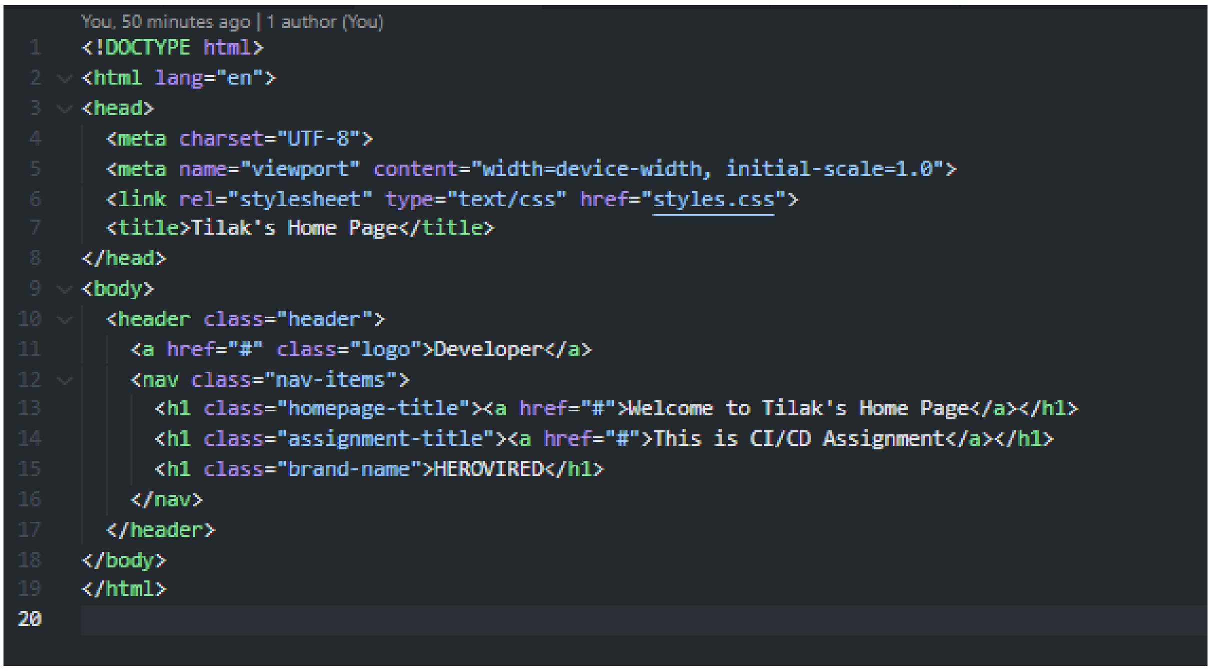
CI/CD pipeline

A step-by-step guide to creating a complete CI/CD pipeline using Bash, Python, and crontabs as per the tasks you've specified:

Task1: Setup a Simple HTML Project

Create a new HTML project folder on your local machine.
Create an **index.html** file within the project folder with some essential HTML content.

Example:



You, 50 minutes ago | 1 author (You)

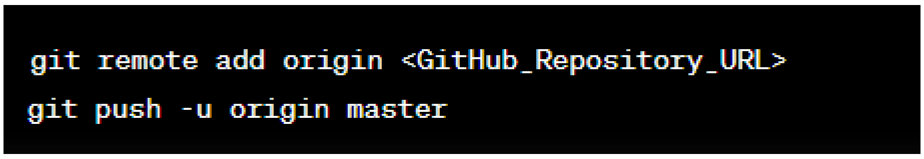
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="styles.css">
  <title>Tilak's Home Page</title>
</head>
<body>
  <header class="header">
    <a href="#" class="logo">Developer</a>
    <nav class="nav-items">
      <h1 class="homepage-title"><a href="#">Welcome to Tilak's Home Page</a></h1>
      <h1 class="assignment-title"><a href="#">This is CI/CD Assignment</a></h1>
      <h1 class="brand-name">HEROVIRED</h1>
    </nav>
  </header>
</body>
</html>
```

Initialize a local Git repository in the project folder (Use **Visual Code Studio**):

```
git init
```

1. git init
2. git config --global user.name <User-name>
3. git config --global user.email tilak.XXXXX@gmail.com

Create a new repository on GitHub and follow the instructions to link your local repository to the remote one:



```
git remote add origin <GitHub_Repository_URL>
git push -u origin master
```

Git URL: https://github.com/bpechetti/cicd_assignment.git
In My Case the **Git Branch** in the **Main.**

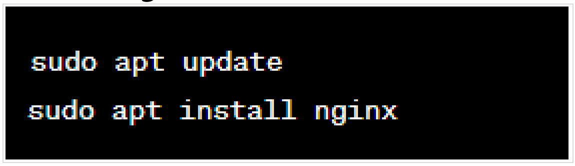
Prerequisites:

Make sure you are able to push the code from your local git to the GitHub repository

1. Generate SSH-Keygen on your local Computer **(Path: C:\Users\User_Name\.ssh\id_rsa.pub)**
2. Copy the key from **id_rsa.pub**
3. Add the SSH key to the associated with your account.
   a. Click on Git_Hub Account Settings
   b. Select SSH and GPGKeys in Access.
   c. Click on New SSH Key
   d. Add the key.

Task 2: Setup an AWS EC2/Local Linux Instance with Nginx

Launch an AWS EC2 instance or set up a local Linux instance.
Install Nginx on the instance:



```
sudo apt update
sudo apt install nginx
```

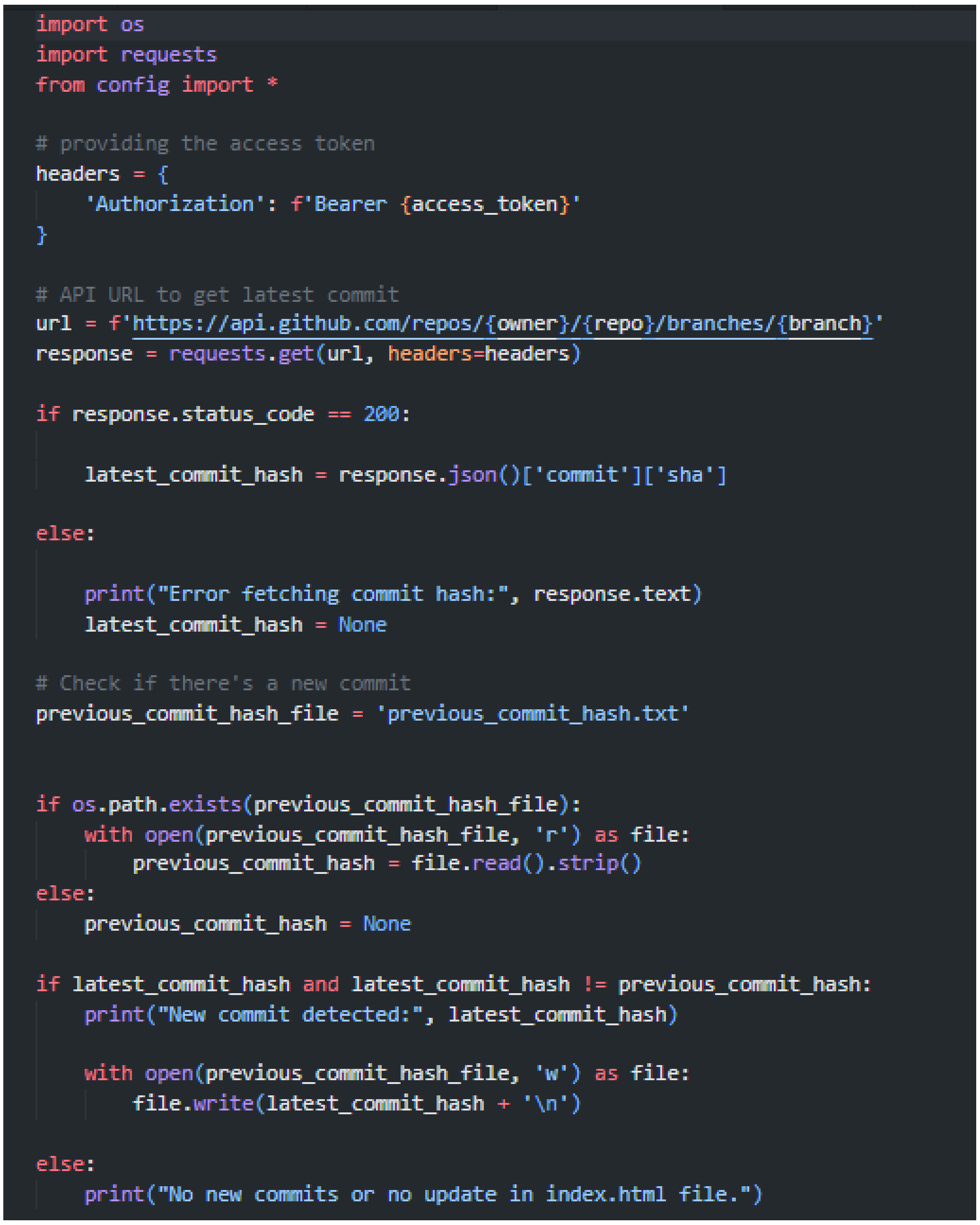
1. Sudo apt update
2. Sudo apt install nginx
3. Sudo systemctl restart nginx

Task 3: Write a Python script to check for new commits.

Create a Python script named **autodeploy.py** and **config.py**
Python code.

```
import os
import requests
from config import *

# providing the access token
headers = {
  'Authorization': f'Bearer {access_token}'
}
```

```
# API URL to get latest commit
url = f'https://api.github.com/repos/{owner}/{repo}/branches/{branch}'
response = requests.get(url, headers=headers)

if response.status_code == 200:

    latest_commit_hash = response.json()['commit']['sha']

else:

    print("Error fetching commit hash:", response.text)
    latest_commit_hash = None

# Check if there's a new commit
previous_commit_hash_file = 'previous_commit_hash.txt'

if os.path.exists(previous_commit_hash_file):
    with open(previous_commit_hash_file, 'r') as file:
        previous_commit_hash = file.read().strip()
else:
    previous_commit_hash = None

if latest_commit_hash and latest_commit_hash != previous_commit_hash:
    print("New commit detected:", latest_commit_hash)

    with open(previous_commit_hash_file, 'w') as file:
        file.write(latest_commit_hash + '\n')

else:
    print("No new commits or no update in index.html file.")
```



```python
import os
import requests
from config import *

# providing the access token
headers = {
    'Authorization': f'Bearer {access_token}'
}

# API URL to get latest commit
url = f'https://api.github.com/repos/{owner}/{repo}/branches/{branch}'
response = requests.get(url, headers=headers)

if response.status_code == 200:

    latest_commit_hash = response.json()['commit']['sha']

else:

    print("Error fetching commit hash:", response.text)
    latest_commit_hash = None

# Check if there's a new commit
previous_commit_hash_file = 'previous_commit_hash.txt'


if os.path.exists(previous_commit_hash_file):
    with open(previous_commit_hash_file, 'r') as file:
        previous_commit_hash = file.read().strip()
else:
    previous_commit_hash = None

if latest_commit_hash and latest_commit_hash != previous_commit_hash:
    print("New commit detected:", latest_commit_hash)

    with open(previous_commit_hash_file, 'w') as file:
        file.write(latest_commit_hash + '\n')

else:
    print("No new commits or no update in index.html file.")
```

Code for **Config.py**

```
access_token = 'github_pat_11ASWJZ4Q0U9ja8Vxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
owner = 'Git_hub User-id'
repo = 'CICD_Pipeline'
branch ='main'
local_repo = 'D:\herovired\CICD\CICD_Pipeline'
nginx_path ='/var/www/html/CICD_Pipeline'
files_to_copy ='index.html'
```

Task 4: Write a Bash Script to deploy the code

Create a Bash script named **cicd_code.sh**

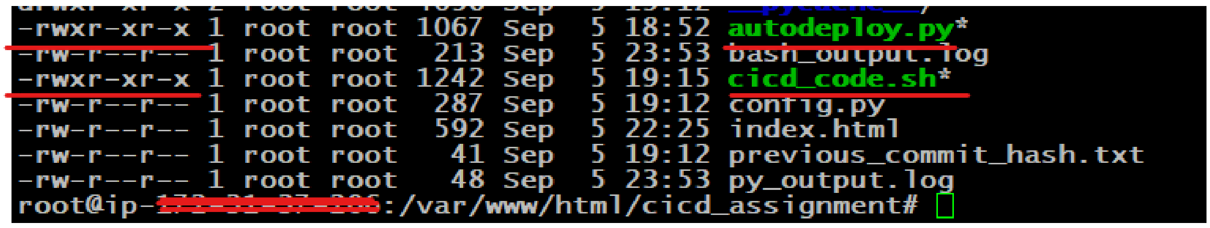
bashCopy code

```bash
#!/bin/bash

# Define the variables
GIT_REPO_URL="https://github.com/bpechetti/cicd_assignment.git"  # Replace with your Git repository URL
WEB_ROOT="/var/www/html/cicd_assignment"  # Replace with the path to your web application's root directory
NGINX_RESTART_CMD="sudo systemctl restart nginx"  # Replace with the appropriate command to restart Nginx

# Change to the web root directory
cd "$WEB_ROOT" || exit 1

# Clone the latest code from the Git repository
git pull origin main  # You can replace 'master' with the branch you want to clone

# Check if the Git operation was successful
if [ $? -eq 0 ]; then
   echo "Code successfully updated from Git repository."

   # Restart Nginx
   echo "Restarting Nginx..."
   $NGINX_RESTART_CMD

   # Check if Nginx restart was successful
   if [ $? -eq 0 ]; then
      echo "Nginx restarted successfully."
   else
      echo "Failed to restart Nginx. Please check the Nginx configuration and try again."
      exit 1
   fi
else
   echo "Failed to update code from Git repository. Please check the Git repository and try again."
   exit 1
fi
```

Make the script executable:

Bash Copy code

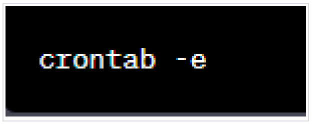
1. **Sudo chmod +x <File_name>** I have made changes to **Autodeploy.py** and **cicd_code.sh** Files.



```
-rwxr-xr-x 1 root root 1067 Sep  5 18:52 autodeploy.py*
-rw-r--r-- 1 root root  213 Sep  5 23:53 bash_output.log
-rwxr-xr-x 1 root root 1242 Sep  5 19:15 cicd_code.sh*
-rw-r--r-- 1 root root  287 Sep  5 19:12 config.py
-rw-r--r-- 1 root root  592 Sep  5 22:25 index.html
-rw-r--r-- 1 root root   41 Sep  5 19:12 previous_commit_hash.txt
-rw-r--r-- 1 root root   48 Sep  5 23:53 py_output.log
root@ip-[redacted]:/var/www/html/cicd_assignment#
```

Task5: Setup a Cronjob to Run the Python Script

Open your crontab for editing:



```
crontab -e
```

Command: crontab -e

Add the following line to run the Python script every minute:

*** * * * * /usr/bin/python3 /var/www/html/cicd_assignment/autodeploy.py > /var/www/html/cicd_assignment/py_output.log 2>&1**

*** * * * * usr/bin/bash /var/www/html/cicd_assignment/cicd_code.sh > /var/www/html/cicd_assignment/bash_output.log 2>&1**

Task 6: Test the Setup

Make a new commit to your GitHub repository.
Wait for the Python script to run via the cron job (after 30 minutes) or manually run it.
If the Python script detects a new commit, it will trigger the deployment Bash script.
The Bash script will pull the latest code and restart Nginx.
Check your Nginx server to ensure the changes have been deployed.