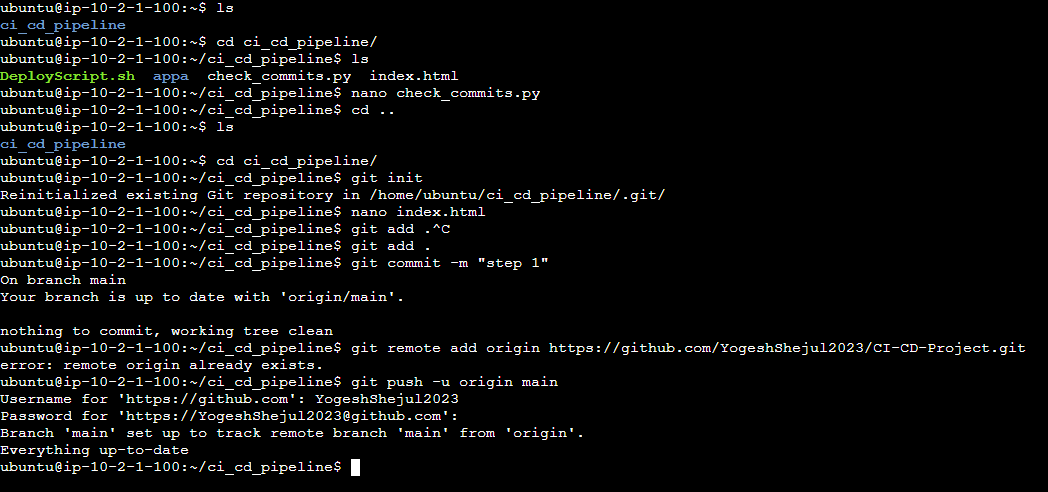
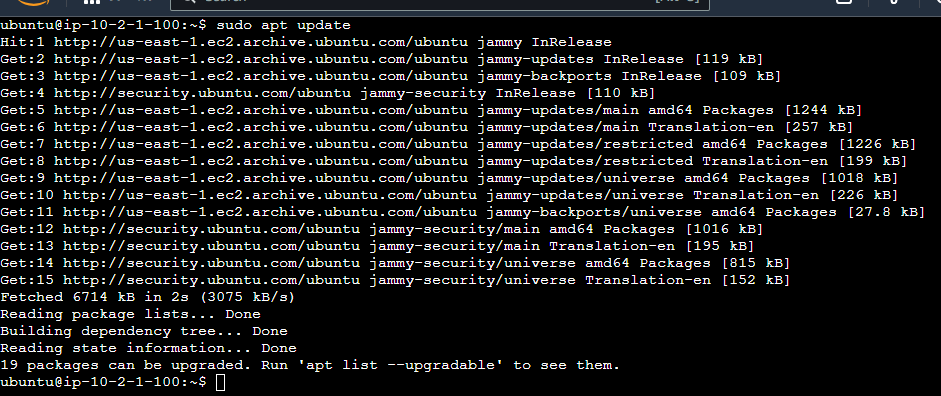
Create a complete CI-CD pipeline using bash, python and crontabs. The list of tasks is specified below:

**Task 1: Set Up a Simple HTML Project**

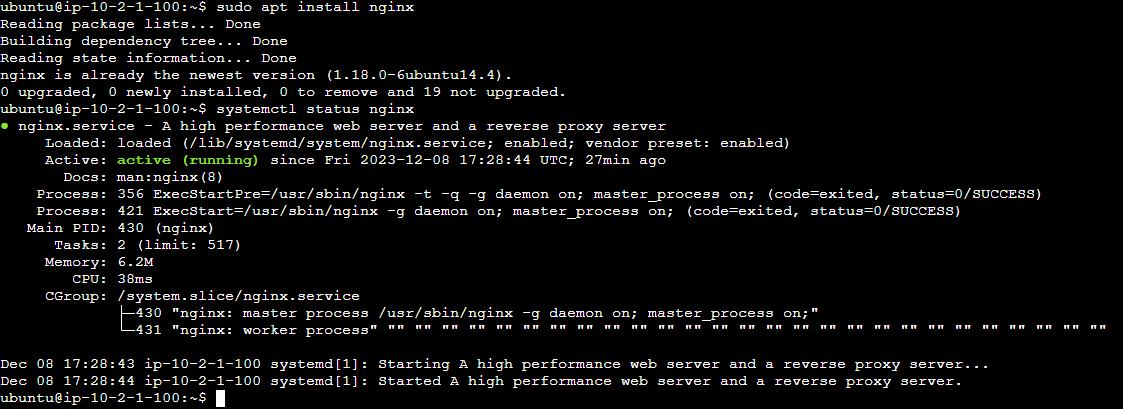
Create a simple HTML project and push it to a GitHub repository.



**Task 2: Set Up an AWS EC2/Local Linux Instance with Nginx**



Install Nginx



**Task 3: Write a Python Script to Check for New Commits**

Create a Python script to check for new commits using the GitHub API.

import os

import requests

import subprocess

import shutil

from git import Repo

from importlib.machinery import SourceFileLoader

credentials = SourceFileLoader("credentials", "credentials.py").load\_module()

owner = credentials.owner

repo = credentials.repo

branch = credentials.branch

local\_repo\_path = credentials.local\_repo\_path

access\_token = credentials.access\_token

nginx\_path = '/var/www/html/'

# API request headers

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 = '84d3df4a5fa1e609dfa5a7775522fb427fb4d82'

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)

    #method 1: using the bash script to clone the repo to ngnix folder

    subprocess.run(["bash","/home/hero/cicdtool/deploy.sh"])

    #Method 2 : check if the repo is already there if not then clone it.

    # Clone or pull the repository

    if os.path.exists(local\_repo\_path):

        repo = Repo(local\_repo\_path)

        repo.remotes.origin.pull()

    else:

        repo = Repo.clone\_from(f'https://github.com/{owner}/{repo}.git', local\_repo\_path)

    # Check if index.html has changed

    if repo.git.diff(previous\_commit\_hash, latest\_commit\_hash, '--', file\_to\_copy):

        src\_path = os.path.join(local\_repo\_path, file\_to\_copy)

        dest\_path = os.path.join(nginx\_path, file\_to\_copy)

        if os.path.exists(src\_path):

            shutil.copy(src\_path, dest\_path)

            print("Copied index.html to Nginx folder.")

    else:

        print("No changes in index.html.")

    # Update the previous commit hash

    with open(previous\_commit\_hash\_file, 'w') as file:

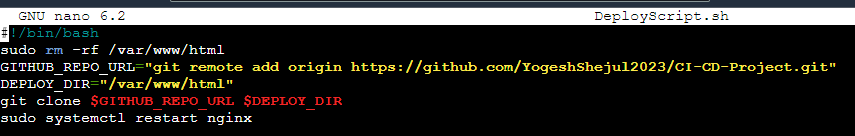
        file.write(latest\_commit\_hash)

else:

    print("No new commits.")

**Task 4: Write a Bash Script to Deploy the Code**

Create a bash script to clone the latest code and restart Nginx.

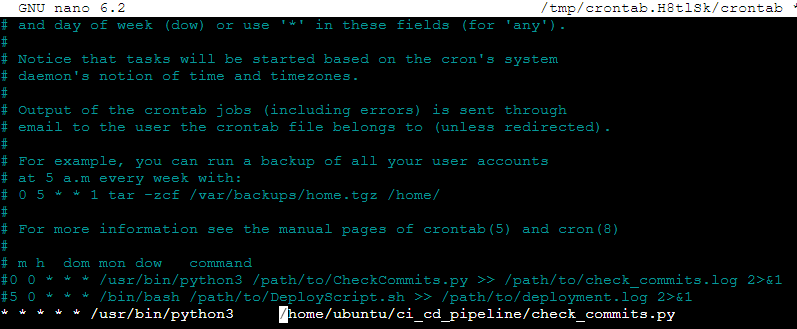




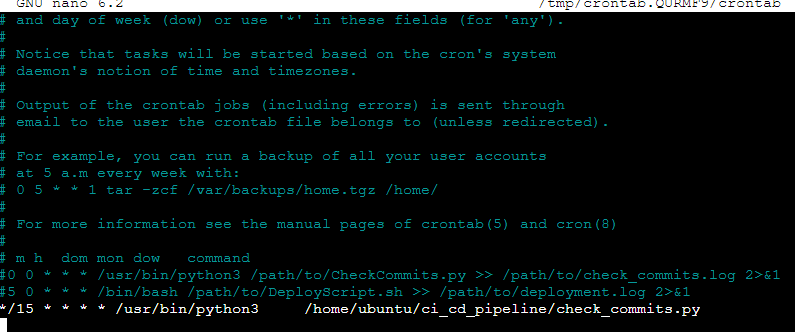
**Task 5: Set Up a Cron Job to Run the Python Script**

Create a cron job to run the Python script at regular intervals.

Created a cron job which will run the python script after every min



Then changes the timing to 15 min, after every 15 min the python script will get executed automatically.



**Task 6: Test the Setup**

Make a new commit to the GitHub repository and check that the changes are automatically deployed.

Before doing any change in backend nginx page

Made some changes like added a new project4 edited the email, LinkedIn and GitHub URLs.

After making the changes in the file added the file committed it and pushed it to GITHUB repository.