Python coding interview questions related to DevOps

Write a Python script to build a Docker image from a Dockerfile and run a container from the image.

```
[root@ip-172-31-40-46 pythontask]# python dockerimage.py
Container 2caee54776f0 is running
[root@ip-172-31-40-46 pythontask]# docker ps
CONTAINER ID   IMAGE          COMMAND             CREATED          STATUS          PORTS      NAMES
2caee54776f0   cf600c39df44   "catalina.sh run"   3 seconds ago    Up 2 seconds    8080/tcp   wonderful_booth
af71f7df9b27   cf600c39df44   "catalina.sh run"   15 seconds ago   Up 15 seconds   8080/tcp   dreamy_ritchie
[root@ip-172-31-40-46 pythontask]# ls
dockerimage.py
[root@ip-172-31-40-46 pythontask]# vi dockerimage.py
[root@ip-172-31-40-46 pythontask]# cat dockerimage.py
import docker

# Create a Docker client
client = docker.from_env()

# Build the Docker image
image, build_logs = client.images.build(path="/home/ec2-user/Docker/java-hello-world-webapp", tag="my_image:latest")

# Run a container
container = client.containers.run(image, detach=True)
print(f"Container {container.short_id} is running")
```

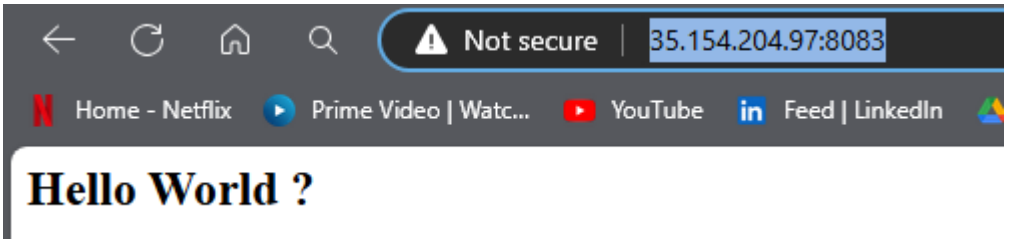
Write a Python script to trigger a Jenkins job and print the build status using the Jenkins API

```
[root@ip-172-31-40-46 pythontask]# python jenkinstrigger.py
FAILURE
[root@ip-172-31-40-46 pythontask]# python jenkinstrigger.py
SUCCESS
[root@ip-172-31-40-46 pythontask]# ls
dockerimage.py  jenkinstrigger.py
[root@ip-172-31-40-46 pythontask]# cat jenkinstrigger.py
import requests
from requests.auth import HTTPBasicAuth

jenkins_url = 'http://35.154.204.97:8080/job/sample/build'
username = 'kushagra'
password = 'sheero'

# Trigger Jenkins job
response = requests.post(jenkins_url, auth=HTTPBasicAuth(username, password))
if response.status_code == 201:
    print("Job triggered successfully")

# Print build status
build_status_url = 'http://35.154.204.97:8080/job/sample/lastBuild/api/json'
status_response = requests.get(build_status_url, auth=HTTPBasicAuth(username, password))
print(status_response.json()['result'])

[root@ip-172-31-40-46 pythontask]#
```

Write a Python script to list all pods in a specific namespace using the Kubernetes API

```
[ec2-user@ip-172-31-40-46 pythontask]$ ls
application.log  dockerimage.py  health.py  jenkinslog.py  jenkinstrigger.py  log.py  podsink8sns.py
[ec2-user@ip-172-31-40-46 pythontask]$ python podsink8sns.py
my-pod
[ec2-user@ip-172-31-40-46 pythontask]$ kubectl get pods --namespace kushagra
NAME     READY   STATUS    RESTARTS   AGE
my-pod   1/1     Running   0          62s
[ec2-user@ip-172-31-40-46 pythontask]$ cat podsink8sns.py
from kubernetes import client, config

# Load kube config
config.load_kube_config()

v1 = client.CoreV1Api()
namespace = 'kushagra'

# List all pods
pods = v1.list_namespaced_pod(namespace)
for pod in pods.items:
    print(pod.metadata.name)

[ec2-user@ip-172-31-40-46 pythontask]$
```

Write a Python script to check the health of multiple web services and log their statuses.




```
[ec2-user@ip-172-31-40-46 pythontask]$ python health.py
http://35.154.204.97:8083/ is up
[ec2-user@ip-172-31-40-46 pythontask]$ cat health.py
import requests

urls = ['http://35.154.204.97:8083/']

for url in urls:
    try:
        response = requests.get(url)
        status = 'up' if response.status_code == 200 else 'down'
    except requests.exceptions.RequestException:
        status = 'down'
    print(f'{url} is {status}')

[ec2-user@ip-172-31-40-46 pythontask]$
```

Write a Python script to parse a log file and count the number of occurrences of each type of error

```
[ec2-user@ip-172-31-40-46 pythontask]$ ls
application.log  dockerimage.py  health.py  jenkinslog.py  jenkinstrigger.py  log.py
[ec2-user@ip-172-31-40-46 pythontask]$ cat jenkinslog.py
import requests

# Jenkins server details
hostname = 'http://35.154.204.97:8080'
username = 'kushagra'
password = 'sheero'
job_name = 'sample'

# Jenkins API endpoint for obtaining last build console output
api_url = f"{hostname}/job/{job_name}/lastBuild/consoleText"

try:
    # Authenticate with Jenkins using basic authentication
    response = requests.get(api_url, auth=(username, password))

    # Check if the request was successful (HTTP status code 200)
    if response.status_code == 200:
        logs = response.text  # Assuming logs are returned as plain text
        print(logs)
    else:
        print(f"Failed to fetch logs. Status code: {response.status_code}")

except requests.exceptions.RequestException as e:
    print(f"Error: {e}")

[ec2-user@ip-172-31-40-46 pythontask]$ python jenkinslog.py
Started by user Kushagra Agarwal
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/sample
[sample] $ /bin/sh -xe /tmp/jenkins17079032185300852437.sh
+ echo 'hello world'
hello world
Finished: SUCCESS

[ec2-user@ip-172-31-40-46 pythontask]$
```

Write a Python script to initialize a Terraform configuration and apply it to create infrastructure

```
ec2-user@ip-172-31-40-46:~/pythontask
import subprocess

def apply_terraform(directory):
    terraform_init = subprocess.run(['terraform', 'init', directory], capture_output=True)
    print(terraform_init.stdout.decode('utf-8'))

    terraform_apply = subprocess.run(['terraform', 'apply', '-auto-approve', directory], capture_output=True)
    print(terraform_apply.stdout.decode('utf-8'))

directory = '/path/to/terraform/config'
apply_terraform(directory)
```

Write a Python script to parse a log file and count the number of occurrences of each type of error

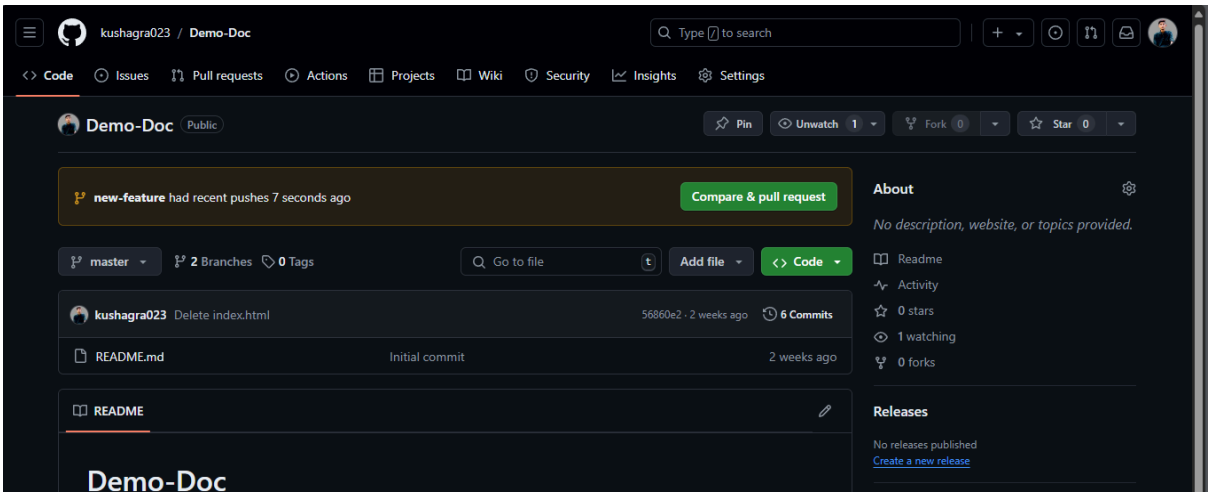
```
[ec2-user@ip-172-31-40-46 pythontask]$ ls
application.log  dockerimage.py  health.py  jenkinstrigger.py  log.py
[ec2-user@ip-172-31-40-46 pythontask]$ vi log.py
[ec2-user@ip-172-31-40-46 pythontask]$ cat log.py
from collections import Counter

def count_errors_and_warnings(log_file):
    # Initialize counters
    error_counter = Counter()
    warning_counter = Counter()

    # Open log file
    with open(log_file, 'r') as file:
        # Iterate through each line in the file
        for line in file:
            # Check for 'ERROR' and 'WARNING' in the line
            if 'ERROR' in line:
                error_counter['ERROR'] += 1
            elif 'WARNING' in line:
                warning_counter['WARNING'] += 1

    return error_counter, warning_counter

# Specify the log file path
log_file = 'application.log'

# Count errors and warnings
error_counter, warning_counter = count_errors_and_warnings(log_file)

# Print results
print(f"Errors: {error_counter}")
print(f"Warnings: {warning_counter}")

[ec2-user@ip-172-31-40-46 pythontask]$ python log.py
Errors: Counter()
Warnings: Counter()
[ec2-user@ip-172-31-40-46 pythontask]$
```

Write a Python script to backup a MySQL database and upload the backup to an S3 bucket.

```
ec2-user@ip-172-31-40-46:~/pythontask
import subprocess
import datetime

def backup_mysql_database(host, username, password, database_name, backup_dir):
    timestamp = datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
    backup_file = f"{database_name}_{timestamp}.sql"
    backup_path = os.path.join(backup_dir, backup_file)
    cmd = f"mysqldump -h {host} -u {username} -p{password} {database_name} > {backup
_path}"
    subprocess.run(cmd, shell=True)
    print(f"MySQL database '{database_name}' backed up to '{backup_path}'")

host = 'localhost'
username = 'root'
password = 'password'
database_name = 'mydatabase'
backup_dir = '/path/to/backup/directory'
backup_mysql_database(host, username, password, database_name, backup_dir)

~
~
~
~
~
```

Write a Python script to clone a Git repository, create a new branch, and push a file to the new branch.

```
[ec2-user@ip-172-31-40-46 pythontask]$ python gitauto.py
Username for 'https://github.com': kushagra023
Password for 'https://kushagra023@github.com':
Username for 'https://github.com': kushagra023
Password for 'https://kushagra023@github.com':
[ec2-user@ip-172-31-40-46 pythontask]$ ls
application.log  dockerimage.py  health.py      jenkinstrigger.py  monitormetrics.py
dbbackup.py      gitauto.py      jenkinslog.py  log.py             podsink8sns.py
[ec2-user@ip-172-31-40-46 pythontask]$ cat gitauto.py
import os
import git

def clone_create_branch_push(repo_url, branch_name, file_path, commit_message):
    repo_dir = '/home/ec2-user/Demo-Doc'

    # Check if the directory already exists
    if os.path.exists(repo_dir):
        print(f"Directory '{repo_dir}' already exists. Removing it.")
        os.system(f"rm -rf {repo_dir}")  # Remove the existing directory

    # Clone the repository
    repo = git.Repo.clone_from(repo_url, repo_dir)
    repo.git.checkout('-b', branch_name)

    # Make changes to the file
    with open(os.path.join(repo_dir, file_path), 'w') as f:
        f.write('New content')

    repo.index.add([file_path])
    repo.index.commit(commit_message)
    origin = repo.remote(name='origin')
    origin.push(branch_name)

repo_url = 'https://github.com/kushagra023/Demo-Doc.git'
branch_name = 'new-feature'
file_path = 'newfeature.txt'
commit_message = 'Added new feature'
clone_create_branch_push(repo_url, branch_name, file_path, commit_message)
```



Write A python Script to give system metrics .

```
ec2-user@ip-172-31-40-46:~/pythontask
[ec2-user@ip-172-31-40-46 pythontask]$ python monitormetrics.py
CPU Usage: 5.0%
Memory Usage: 58.3%
CPU Usage: 4.8%
Memory Usage: 58.4%
CPU Usage: 4.6%
Memory Usage: 58.2%
[ec2-user@ip-172-31-40-46 pythontask]$ ls
application.log  dockerimage.py  jenkinslog.py      log.py             podsink8sns.py
dbbackup.py      health.py       jenkinstrigger.py  monitormetrics.py
[ec2-user@ip-172-31-40-46 pythontask]$ vi monitormetrics.py
[ec2-user@ip-172-31-40-46 pythontask]$ cat monitormetrics.py
import psutil
import time

def monitor_system_metrics(interval=10, duration=60):
    end_time = time.time() + duration
    while time.time() < end_time:
        cpu_percent = psutil.cpu_percent(interval=interval)
        memory_stats = psutil.virtual_memory()
        print(f"CPU Usage: {cpu_percent}%")
        print(f"Memory Usage: {memory_stats.percent}%")
        time.sleep(interval)

monitor_system_metrics()

[ec2-user@ip-172-31-40-46 pythontask]$
```

Write a Python script to test a REST API endpoint and validate the response

```
[ec2-user@ip-172-31-40-46 pythontask]$ python apitest.py
API Endpoint http://35.154.204.97:8083/ returned non-JSON content: <html>
<body>
<h2>Hello World ?</h2>
</body>
</html>

[ec2-user@ip-172-31-40-46 pythontask]$ cat apitest.py
import requests

def test_api_endpoint(url):
    try:
        response = requests.get(url)

        # Check the content type of the response
        content_type = response.headers.get('Content-Type', '').lower()

        if 'application/json' in content_type:
            data = response.json()
            print(f"API Endpoint {url} is reachable. Response: {data}")
        else:
            print(f"API Endpoint {url} returned non-JSON content: {response.text}")

    except requests.exceptions.RequestException as e:
        print(f"Failed to reach API Endpoint {url}. Error: {e}")

url = 'http://35.154.204.97:8083/'
test_api_endpoint(url)

[ec2-user@ip-172-31-40-46 pythontask]$
```