KUBERNETES ANSWERS

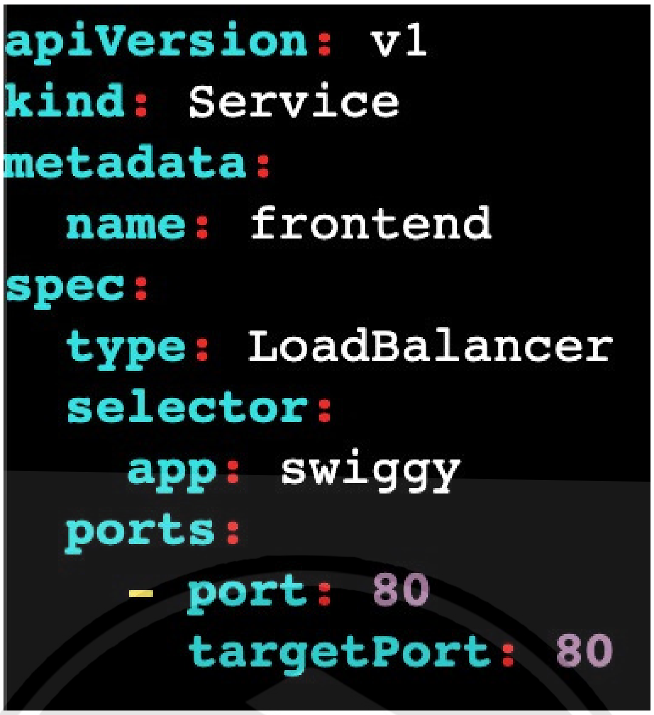
1. WRITE A MANIFEST FILE TO CREATE A POD WITH 2 CONTAINERS

Ans:

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
  labels:
    app: swiggy
spec:
  containers:
  - name: cont-1
    image: nginx
    ports:
    - containerPort: 80
  - name: cont-2
    image: httpd
    ports:
    - containerPort: 80
```

2. WRITE A MANIFEST FILE TO CREATE A LOAD BALANCER SERVICE TO EXPOSE THE POD.

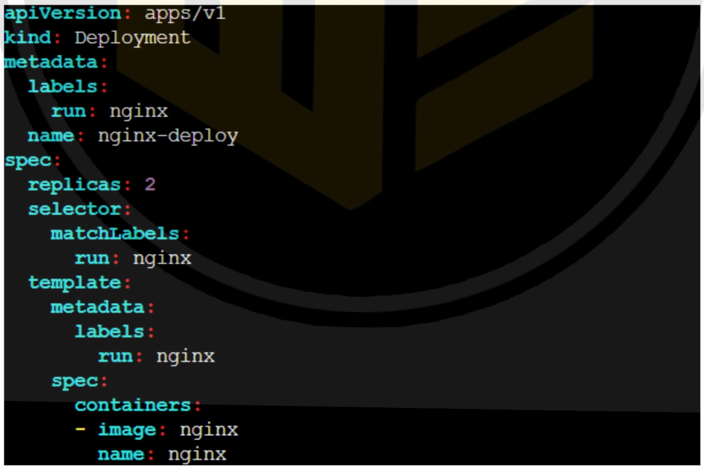
Ans:



```yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend
spec:
  type: LoadBalancer
  selector:
    app: swiggy
  ports:
    - port: 80
      targetPort: 80
```

3. WRITE A MANIFEST FILE FOR DEPLOYMENT.

Ans:



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    run: nginx
  name: nginx-deploy
spec:
  replicas: 2
  selector:
    matchLabels:
      run: nginx
  template:
    metadata:
      labels:
        run: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
```

4. DIFFERENCE BETWEEN DEPLOYMENT AND STATEFUL APPLICATION.

DEPLOYMENT	STATEFUL SET
• It will create POD's with random ID's • Scale down the POD's in random ID's • POD's are stateless POD's • We use this for application deployment	• It will create POD's with sticky ID's • Scale down the POD's in reverse order • POD's are stateful POD's • We use this for database deployment

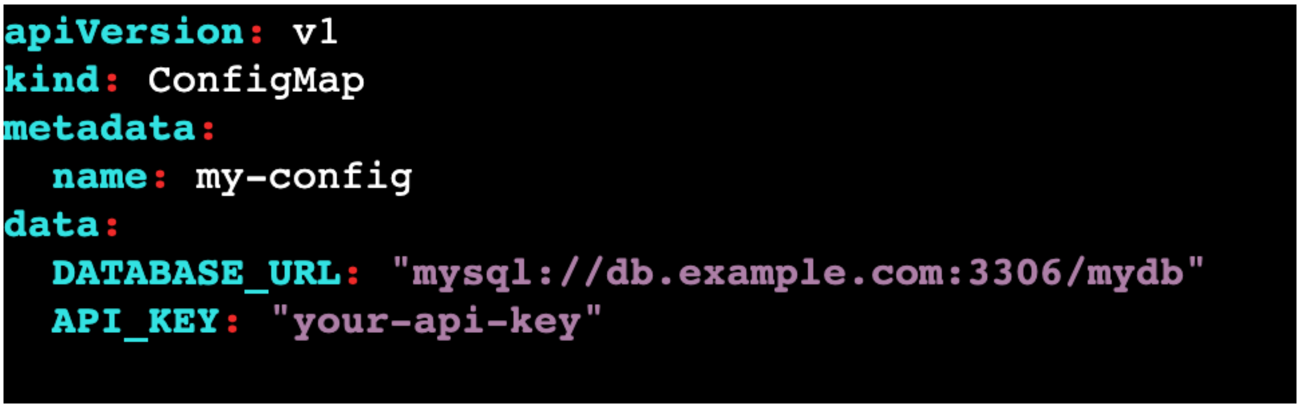
5. BRIEF ABOUNT CONFIG MAPS AND SECRETS AND WRITE MANIFEST FILES.

Ans:

CONFIG MAPS:

- ConfigMap is used to store the configuration data in key-value pairs within Kubernetes.
- But the data should be non confidential data.
- This is one of the ways to decouple the configuration from the application to get rid of hardcoded values.
- Also, if you observe some important values keep changing according to the environments such as development, testing, production, etc ConfigMap helps to fix this issue to decouple the configurations
- So we can set the configuration of data of application separately
- But it does not provider security and encryption. If we want to provide encryption use secrets in Kubernetes.
- Limit of config map data in only 1 MB (we cannot store more than that)
- But if we want to store a large amount of data in config maps we have to mount a volume or use a separate database or file service.

Manifest file to create a config map

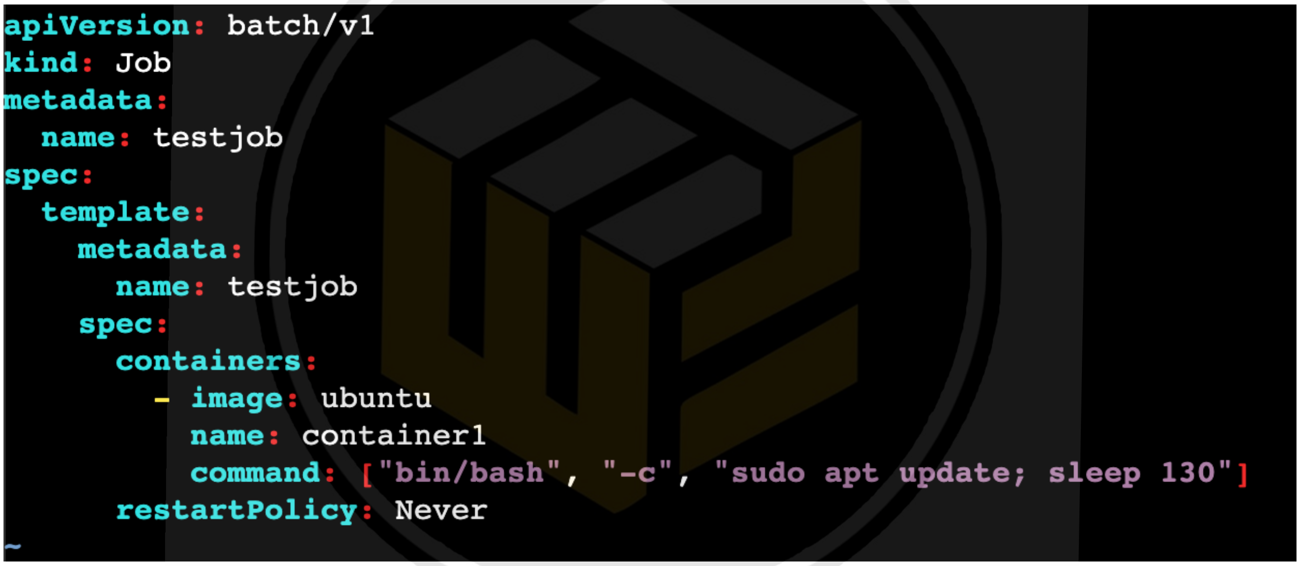


```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  DATABASE_URL: "mysql://db.example.com:3306/mydb"
  API_KEY: "your-api-key"
```

SECRETS:

- There are lot of confidential information that needs to be stored on the server such as database usernames, passwords, or API Keys.
- To keep all the important data secure, Kubernetes has a Secrets feature that encrypts the data.
- Secrets can store data up to 1MB which would be enough.
- Secrets can be created via imperative or declarative ways.
- Secrets are stored in the /tmps directory and can be accessible to pods only.
- After creating the Secrets, applications need to use the credentials or database credentials which will be done by injecting with the pods.

6. WRITE A MANIFEST FILE TO SCHEDULE A JOB.



```
apiVersion: batch/v1
kind: Job
metadata:
  name: testjob
spec:
  template:
    metadata:
      name: testjob
    spec:
      containers:
        - image: ubuntu
          name: container1
          command: ["bin/bash", "-c", "sudo apt update; sleep 130"]
      restartPolicy: Never
~
```

7. COMMAND TO CREATE A SERVICE

Ans:

kubectl expose pod <pod-name> --name=<service-name> \

 --port=<port> --target-port=<targetPort> --type=<service-type>

8. COMMAND TO DESCRIBE ALL PODS.

Ans: kubectl describe po

9. COMMAND TO CREATE A NAME SPACE

Ans: kubectl create ns flm

10. COMMAND TO SWITHC THE NAMESPACE

Ans: kubectl config set-context --current --namespace=your-namespace

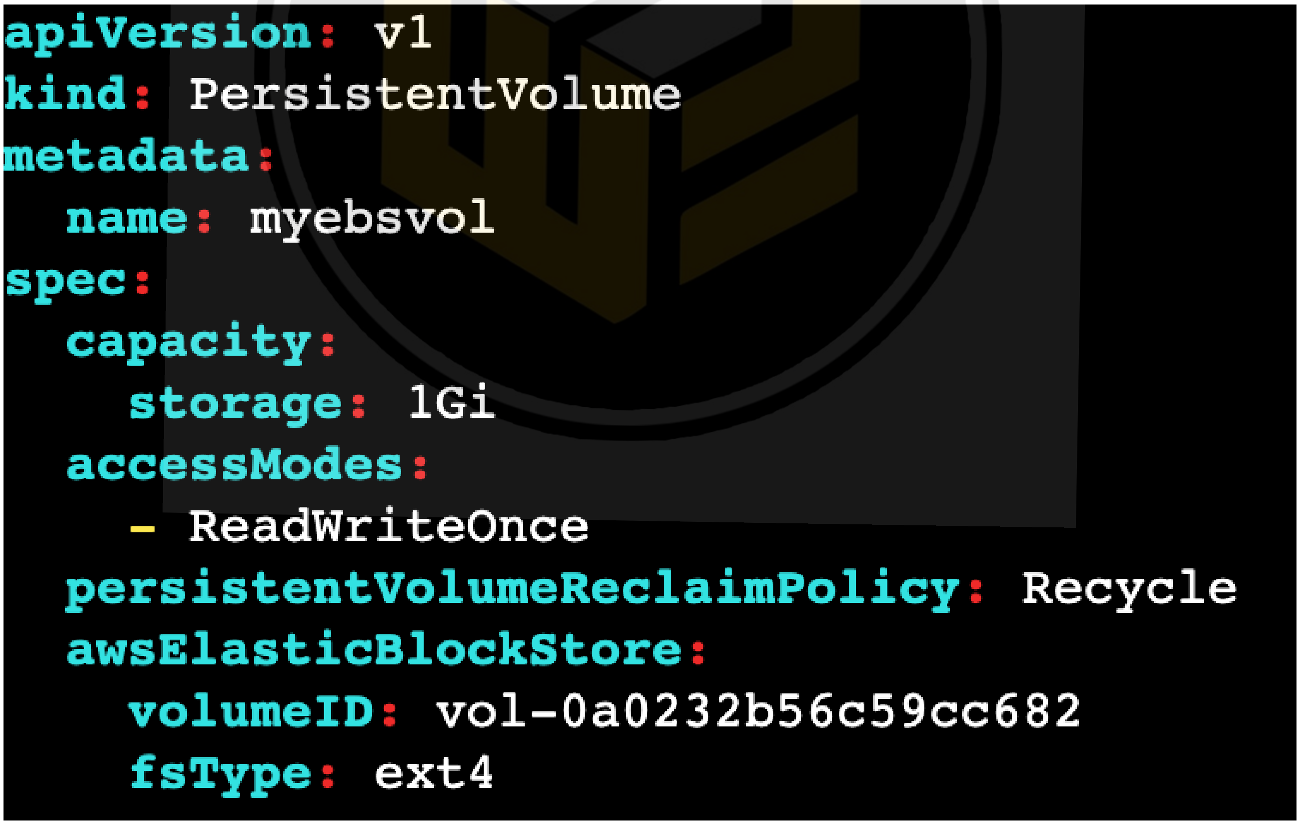
11. COMMAND TO SCALE UP THE DEPLOYMENT

Ans: kubectl scale deployment deployment-name --replicas=desired-replica-count

12. WRITE A MANIFEST FILE FOR PV AND PVC

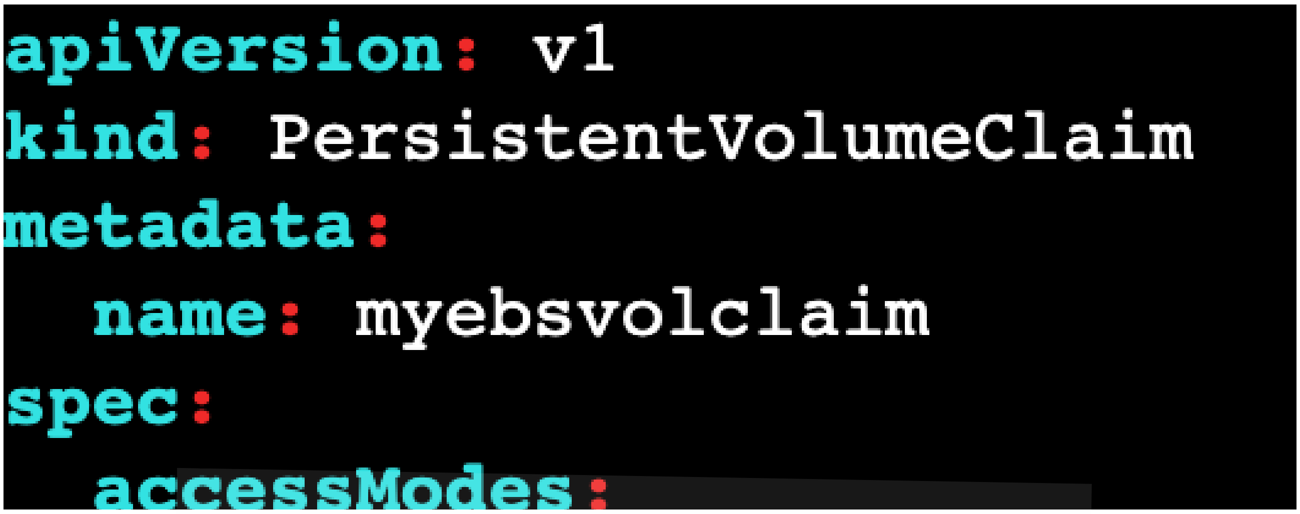
Ans:

pv:



```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: myebsvol
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  awsElasticBlockStore:
    volumeID: vol-0a0232b56c59cc682
    fsType: ext4
```

pvc:



```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myebsvolclaim
spec:
  accessModes:
```