Core concepts- 13%

Pod

1	Kubectl run –dry-run –o yaml nginx --image=nginx --restart=Never -l wf=ov,hs=karthi --env=mami=lax --env=cnt=jp --port=80 -- /bin/sh -c "echo hi world" > pod.yaml
2	kubectl run --generator=run-pod/v1 nginx --image=nginx --port=8080 --command echo hi
3	kubectl get pod podname –o yaml –export > pod.yaml
4	kubectl set image pod/nginx nginx=nginx:1.7.1
5	kubectl exec podname -it bash **or** kubectl exec podname –it -- /bin/sh # get inside the pod

ReplicationController

1	kubectl run replicontrolr --generator=run/v1 --image=redis --replicas=2 --dry-run -o yaml

ReplicaSet

1	kubectl run --generator=deployment/v1beta1 nginx --image=nginx --dry-run --replicas=4 -o yaml // edit the Deployment to replicaSet , remove strategy and empty properties
2	kubectl scale –replicas=3 rc/rc1 rc/rc2 rc/rc3 | kubectl scale deploy mydeploy –replicas=5

Deployments

1	kubectl run --dry-run nginx --image=nginx -l wf=ov,hs=karthi --env=mami=lax --env=cnt=jp --port=80 -o yaml --replicas=5 -- /bin/sh -c "echo hi world" > dep.yaml
2	kubectl set image deploy nginx nginx=nginx:1.7.1
3	kubectl run --generator=deployment/v1beta1 nginx --image=nginx --dry-run --replicas=4 -o yaml
4	kubectl autoscale deploy nginx --min=5 --max=10 --cpu-percent=80 --dry-run -o yaml

Service:

1	kubectl create service clusterip ngservice --tcp=80:80 --dry-run -o yaml
2	kubectl create service nodeport nginx --tcp=80:8000 --node-port=30080 --dry-run -o yaml
3	kubectl expose deployment nginx --type=NodePort --port=80 --target-port=8000 --name=nginx-serv --dry-run -o yaml kubectl expose deployment nginx –port=80 –target-port=8000
4	kubectl run --image=nginx ng --port=8080 --expose --dry-run -o yaml kubectl run nginx --image=nginx --restart=Never --port=80 --expose

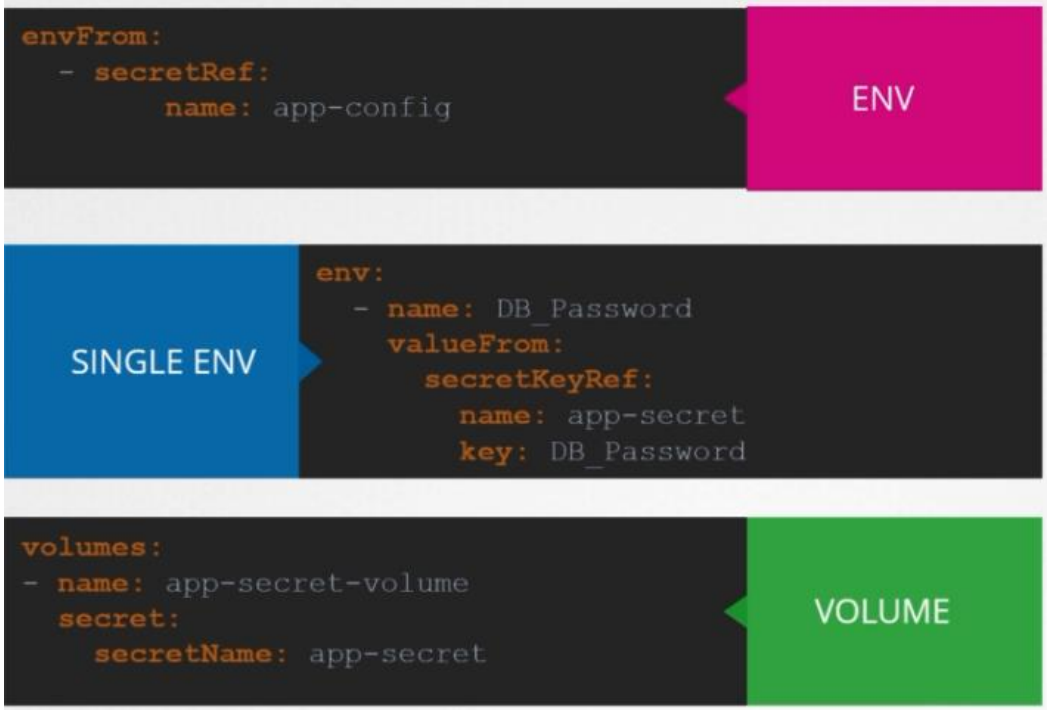
NameSpaces:

1.	Kubectl create namespace mynamespace
2	kubectl get all --all-namespaces
3	kubectl run nginx –image=nginx –n mynamespace

MultiPod container- 10%

	Patterns : Side car , Adapter, ambassador
	Generate single container and Practise copy paste many containers Insering Env variables/ mounting Volumes
1	kubectl exec –it pod-name -c container-name-2 -- /bin/sh

Configurations- 18%

1.	Know about command (entryPoint) and args (cmd) , env
2.	Kubectl create cm app-config –from-litereal=wif=ov –from-literal=hus=karth
3.	
4.	Kubectl create secret generic app-secret –from-literal=DB_HOST=mysql
5	

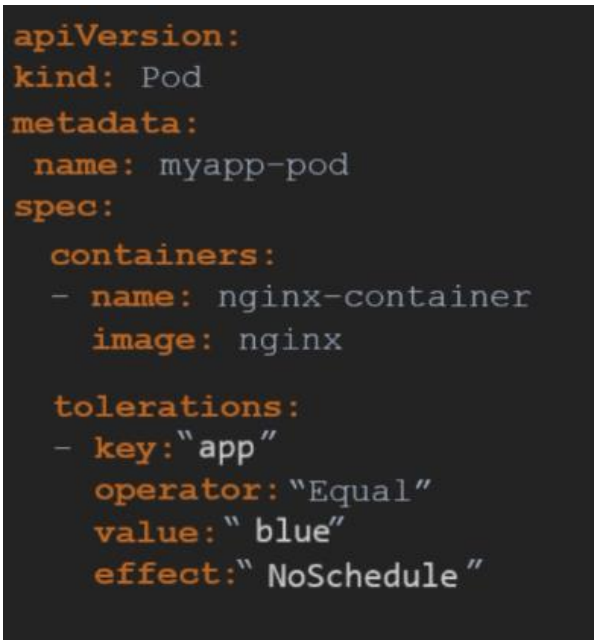
6	**Security contexts**
	Pod level: **Container level (here only capabilities supported)**
	Security Context
	`apiVersion: v1`
	`kind: Pod`
	`metadata:`
	`  name: web-pod`
	`spec:`
	`  securityContext:`
	`  runAsUser: 1000`
	`  containers:`
	`  - name: ubuntu`
	`  image: ubuntu`
	`  command: ["sleep", "3600"]`
	`apiVersion: v1`
	`kind: Pod`
	`metadata:`
	`  name: web-pod`
	`spec:`
	`  containers:`
	`  - name: ubuntu`
	`  image: ubuntu`
	`  command: ["sleep", "3600"]`
	`  securityContext:`
	`  runAsUser: 1000`
	`  capabilities:`
	`  add: ["MAC_ADMIN"]`

7.	Kubectl create sa my-service-account

8	pod-definition.yml
	`apiVersion: v1`
	`kind: Pod`
	`metadata:`
	`  name: my-kubernetes-dashboard`
	`spec:`
	`  containers:`
	`  - name: my-kubernetes-dashboard`
	`  image: my-kubernetes-dashboard`
	`  serviceAccount: dashboard-sa`

9	you can replace the POD type with Deployment, then add your **serviceAccountName: default** under
	`template:`
	`  spec:`
	`  serviceAccountName: default`

10	kubectl run nginx --image=nginx --restart=Never --requests=cpu=100m,memory=256Mi --limits=cpu=200m,memory=512Mi --dry-run -o yaml
11	`kubectl taint nodes node-name key=value:taint-effect` NoSchedule | PreferNoSchedule | NoExecute

12	
13	Kubectl label node node-name size=Large
14	
15	Node affinity -> more options In, NotIn, Exists, DoesNotExist, Gt, Lt ; Get template from k8s.io/docs

POD DESIGN – 20%

1	Kubectl get pods –selector app=App1 Kubectl get pod –show-labels // to display all labels Kubectl get pod –L app // capital L for only specifying Key . Kubectl get pod –l app=ov
2	kubectl label pod nginx2 app=v2 kubectl label pod nginx2 app=v2 –overwrite kubectl label po nginx1 nginx2 nginx3 app- // to remove app label from the pods
3	kubectl annotate po nginx1 nginx2 nginx3 description='my description' kubectl annotate po nginx1 nginx2 nginx3 description- Kubectl rollout status deployment my-app-deployment

	Kubectl rollout history deployment my-app-deployment kubectl rollout pause deploy nginx // to pause the rollout kubectl rollout resume deploy nginx
4	Strategy -> Recreate and Rolling update
5	kubectl rollout undo deployment/mydeploy kubectl rollout undo deploy nginx --to-revision=2
6	kubectl run --generator=job/v1 --image=ubuntu myjob --restart=OnFailure -- /bin/sh -c 'echo hello;sleep 30;echo world'

	``yaml
	apiVersion: batch/v1
	kind: Job
	metadata:
	  name: random-error-job
	spec:
	  completions: 3
	  parallelism: 3
	  template:
	  spec:
	  containers:
	  - name: random-error
	  image: kodekloud/random-error
	  restartPolicy: Never
	``
	backoffLimit: 25 # This is so the job does not quit before it succeeds

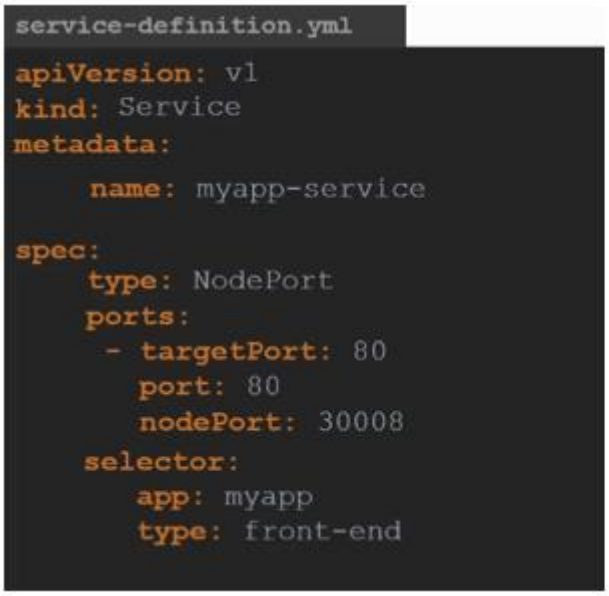
	kubectl run --generator=cronjob/v1beta1 --image=ubuntu cron-job --restart=Never --schedule="30 21 * * *"

Observability – 18%

1.	```yaml
	readinessProbe:
	  httpGet:
	  path: /api/ready
	  port: 8080
	  initialDelaySeconds: 10
	  periodSeconds: 5
	  failureThreshold: 8
	``
	``yaml
	readinessProbe:
	  tcpSocket:
	  port: 3306
	``
	``yaml
	readinessProbe:
	  exec:
	  command:
	  - cat
	  - /app/is_ready
	``
	Kubectl rollout history deployment my-app-deployment

2	 
3	Kubectl logs –f pod-name container-name Kubectl logs podname
4	Kubectl top node

Network and Services – 13%

1	 
2	**kubectl create service nodeport webapp-service --node-port=30080 --tcp=8080:8080 --dry-run -o yaml > t.yaml** kubectl run nginx --image=nginx --restart=Never --port=80 --expose

3

Ingress-wear-watch.yaml

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-wear-watch
spec:
  rules:
  - http:
      paths:
      - path: /wear
        backend:
          serviceName: wear-service
          servicePort: 80
      - path: /watch
        backend:
          serviceName: watch-service
          servicePort: 80
```

Ingress-wear-watch.yaml

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-wear-watch
spec:

  rules:
  - host: wear.my-online-store.com
    http:
      paths:
      - backend:
          serviceName: wear-service
          servicePort: 80
  - host: watch.my-online-store.com
    http:
      paths:
      - backend:
          serviceName: watch-service
          servicePort: 80
```

4

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: db-policy
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          name: api-pod
    ports:
    - protocol: TCP
      port: 3306
```

State Persistence – 8%

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: random-number-generator
spec:
  containers:
  - image: alpine
    name: alpine
    command: ["/bin/sh","-c"]
    args: ["shuf -i 0-100 -n 1 >> /opt/number.out;"]
    volumeMounts:
    - mountPath: /opt
      name: data-volume

  volumes:
  - name: data-volume
    hostPath:
      path: /data
      type: Directory
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
   name: pv-vol1
spec:
  accessModes:
      - ReadWriteOnce
  capacity:
      storage: 1Gi
  hostPath:
     path: /tmp/data
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
   name: pv-vol1
spec:
  accessModes:
      - ReadWriteOnce
  capacity:
      storage: 1Gi
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
     - ReadWriteOnce

  resources:
     requests:
       storage: 500Mi
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-vol1
spec:
  accessModes:
      - ReadWriteOnce
  capacity:
      storage: 1Gi
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

Use PVC in the POD

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp
spec:
  containers:
  - name: nginxx
    image: nginx

    volumeMounts:
    - mountPath: /log
      name: log-volume

  volumes:
  - name: log-volume
    persistentVolumeClaim:
      claimName: myclaim
```