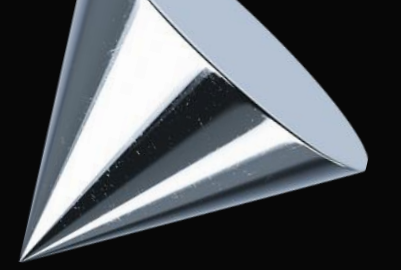


Cheat Sheet

Navdeep Kaur



Dockers



Docker Containers

```
docker run httpd -name httpd-container -d -p 8080:80 httpd:alpine.3.14
```

```
docker ps
```

```
docker ps -a
```

```
docker exec -it <container_id> sh
```

```
docker rm -f <container_id>
```

```
docker stop <container_id>
```

```
docker start <container_id>
```

```
docker cp <container_id>:/usr/local/apache2/htdocs/index.html .
```

```
docker cp new-index.html <container_id>:/usr/local/apache2/htdocs
```



Docker Images

```
docker pull httpd:alpine.3.14
```

```
docker images
```

```
docker build -t myimage:1.0.0 .
```

```
docker push myimage:1.0.0
```

```
docker rmi httpd:alpine3.14
```



Docker Statistics/Inspection

docker logs <container_id>

docker top <container_id>

docker inspect <container_id>

docker port <container_id>

docker network ls



Kubernetes



Pods commands

```
kubectl run httpd-pod --image httpd
```

```
kubectl get pod
```

```
kubectl describe pod httpd-pod
```

```
kubectl logs httpd-pod
```

```
kubectl exec -it httpd-pod sh
```

```
kubectl run httpd-new --image=httpd:alpine --port=80 --expose
```

```
kubectl port-forward httpd-new 8085:80
```

Pods using yaml

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: myweb-app
  labels:
    type: frontend
spec:
  containers:
    - name: httpd-container
      image: httpd
    - name: mongodb-container
      image: mongo
```

```
kubectl create -f pod-definition.yaml
kubectl exec -it myweb-app -c httpd-container sh
kubectl logs myweb-app -c httpd-container sh
kubectl exec -it myweb-app -c mongodb-container sh
kubectl logs myweb-pod -c mongodb-container
kubectl run myweb-app1 -- image httpd --dry-run=client
-o yaml > webapp.yaml
kubectl get pod myweb-app1 -o yaml > webapp1.yaml
kubectl get pod --selector type=frontend
```


Replica Sets



Replica Sets using Yaml

replica-definition.yaml

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myweb-replicas
  labels:
    type: front-replica
spec:
  replicas: 3
  selector:
    matchLabels:
      type: frontend
  template:
    metadata:
      name: myweb-app
      labels:
        type: frontend
    spec:
      containers:
        - name: httpd-container
          image: httpd
```

```
kubectl create -f replica-definition.yaml
kubectl get replicaset

kubectl describe replicaset mywebapp-replicas

kubectl scale --replicas=6 -f replica-set.yaml

kubectl get replicaset -o yaml > newreplica.yaml
```

Deployments



Deployments using Yaml

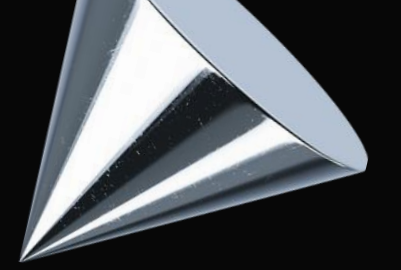
deployment-definition.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myweb-deployment
  labels:
    type: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      type: frontend
  template:
    metadata:
      name: myweb-app
      labels:
        type: frontend
    spec:
      containers:
        - name: httpd-container
          image: httpd
```

```
kubectl create -f deployment-definition.yaml
kubectl get deployment
kubectl describe deployment myweb-deployment
kubectl edit deployment myweb-deployment
kubectl rollout history deployment/mywebapp
kubectl rollout undo deployment/mywebapp
kubectl rollout undo deployment/backend --to-revision=2
kubectl rollout status -w deployment/mywebapp
kubectl rollout restart deployment/mywebapp

kubectl create deployment webapp1 --image=httpd --replicas=3
kubectl create deployment webapp1 --image=httpd --replicas=3
--dry-run=client -o yaml > webapp.yaml
```

Namespace



Namespace commands

namespace-definition.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: dev
```

```
kubectl create -f namespace-definition.yaml
```

```
kubectl create namespace prod
```

```
kubectl get namespaces
```

```
kubectl run myweb-app1 -- image httpd -n dev
```

```
kubectl get pod -n dev
```

```
kubectl get pod -all-namespaces
```

```
kubectl get all -all-namespaces
```


Environment Variables



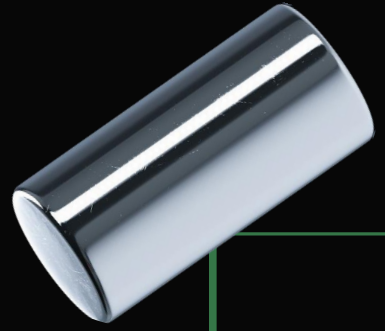
Pods using yaml

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: myweb-app
  labels:
    type: frontend
spec:
  containers:
  - name: httpd-container
    image: httpd
    env:
      - name: APP_COLOR
        value: PINK
      - name: APP_ENV
        value: PROD
```

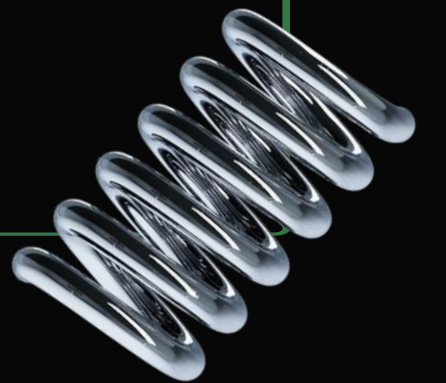
ConfigMaps





```
kubectl create configmap myconfigmap  
--from-literal=APP_COLOR=pink  
--from-literal=APP_LOCATION=us
```

```
kubectl create configmap myconfigmap1 --from-file=db.properties
```



Using Config Maps

config-map.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: myweb-config
data:
  color : PINK
  location: US
```

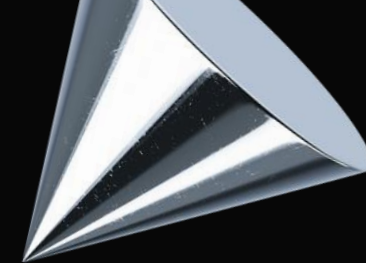
pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
spec:
  containers:
  - name: httpd-container
    image: httpd
    envFrom:
    - configMapRef:
        name: myweb-config
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
spec:
  containers:
  - name: httpd-container
    image: httpd
    env:
    - name: APP_COLOR
      valueFrom:
        configMapKeyRef:
          name: myweb-config
          key: color
```

Secrets



Using Secrets

secrets.yaml

```
apiVersion: v1
kind: Secrets
metadata:
  name: myweb-secrets
data:
  DB_HOST: mysql
  DB_PASSWORD: pass123
```

```
kubectl create -f secret.yaml
echo -n 'mysql' | base64 bXlzcWw=
echo -n 'pass123' | base64 cGFzc2EyMw==
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
spec:
  containers:
  - name: httpd-container
    image: httpd
    envFrom:
    - configMapRef:
        name: myweb-config
    - secretRef:
        name: myweb-secrets
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
spec:
  containers:
  - name: httpd-container
    image: httpd
    env:
    - name: MONGO_HOST
      valueFrom:
        secretKeyRef:
          name: myweb-secrets
          key: DB_HOST
```


Mount Secrets as Volumes

secrets.yaml

```
apiVersion: v1
kind: Secrets
metadata:
  name: myweb-secrets
data:
  DB_HOST: mysql
  DB_PASSWORD: pass123
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
spec:
  containers:
    - name: httpd-container
      image: httpd
      volumeMount:
        - name: app-secret-volume
          mountPath: /etc/secret-volume
      volumes:
        - name: app-secret-volume
          secret:
            secretName: myweb-secrets
```

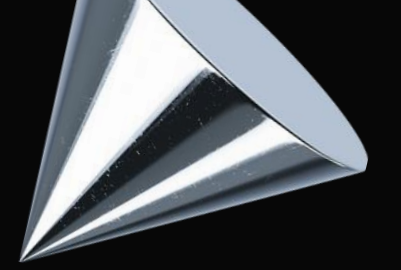


```
kubectl create secret generic my-secret  
--from-literal=DB_HOST=mysql  
--from-literal=DB_PASS=pass123
```

```
kubectl create secret generic mysecret --from-file= db.properties
```



Private Repos

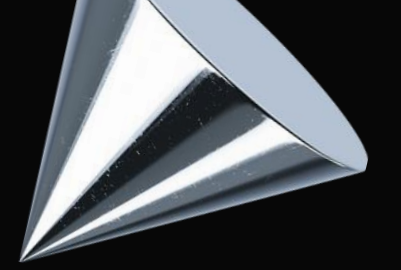


pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nodeapp-pod
spec:
  containers:
  - name: nodeapp-container
    image: navjoy220161/nodemongo:1.0.10
    imagePullSecrets:
    - name: regpriv1
```

```
Kubectl create secret docker-registry regpriv
--docker-server=https://index.docker.io/v1/
--docker-username=<username>
--docker-password=<password>
--docker-email=<email>
```

Volumes

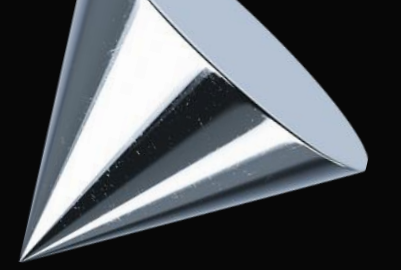


Mount Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: myweb-app
  labels:
    type: frontend
spec:
  containers:
    - name: httpd-container
      image: httpd
      volumeMount:
        - mountPath: /opt
          name: mongodb-volume

  volumes:
    - name: mongodb-volume
      hostPath:
        path: /mongo
        type: Directory
```

Persistent Volumes



Persistent Volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv1-volume
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  hostPath:
    path: /tmp/data
```

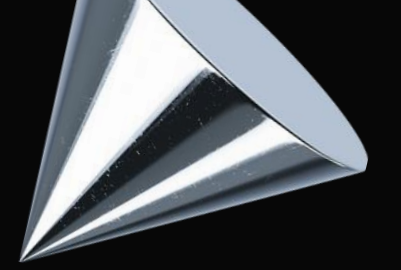
accessModes

Read Only Mode

Read Write Once

Read Write Many

Persistent Volumes Claim



Persistent Volume Claim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500 Mi
```

Mount Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: myweb-app
  labels:
    type: frontend
spec:
  containers:
    - name: httpd-container
      image: httpd
      volumeMount:
        - mountPath: /data/db
          name: mongodb-volume

  volumes:
    - name: mongodb-volume
      persistentVolumeClaim:
        claimName: mongo-pvc
```

Storage Class



Persistent Volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv1-volume
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  gcePersistentDisk:
    fsType: ext4
    pdName: pd-disk
```

```
gcloud beta compute disks create \
--size 2GB
--region us-east-1b
pd-disk
```

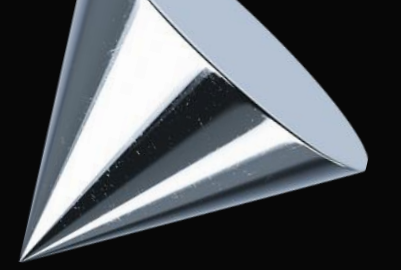
Static Provisioning

```
apiVersion: storage.k8.io/v1
kind: StorageClass
metadata:
  name: google-storage
provisioner: kubernetes.io/gce-pd
```

Dyanmic Provisioning

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-storage
```

Probes



pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
  labels:
    type: frontend
    location: us
spec:
  containers:
  - name: httpd-container
    image: httpd
```

```
  readinessProbe:
    exec:
      command:
      - cat
      - hello
    initialDelaySeconds: 5
    periodSecond: 3
    failureThreshold: 3
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
  labels:
    type: frontend
    location: us
spec:
  containers:
  - name: httpd-container
    image: httpd
```

```
  readinessProbe:
    httpGet:
      path: /
      port: 8080
    initialDelaySeconds: 5
    periodSecond: 3
    failureThreshold: 3
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
  labels:
    type: frontend
    location: us
spec:
  containers:
  - name: httpd-container
    image: httpd
```

```
  readinessProbe:
    tcpSocket:
      port: 8080
    initialDelaySeconds: 5
    periodSecond: 3
    failureThreshold: 3
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
  labels:
    type: frontend
    location: us
spec:
  containers:
  - name: httpd-container
    image: httpd
```

```
livenessProbe:
  exec:
    command:
      - cat
      - hello
  initialDelaySeconds: 5
  periodSecond: 3
  failureThreshold: 3
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
  labels:
    type: frontend
    location: us
spec:
  containers:
  - name: httpd-container
    image: httpd
```

```
livenessProbe:
  httpGet:
    path: /
    port: 8080
  initialDelaySeconds: 5
  periodSecond: 3
  failureThreshold: 3
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
  labels:
    type: frontend
    location: us
spec:
  containers:
  - name: httpd-container
    image: httpd
```

```
livenessProbe:
  tcpSocket:
    port: 8080
  initialDelaySeconds: 5
  periodSecond: 3
  failureThreshold: 3
```

Taints/Tolerations





Taint Node

Taint

```
kubectl taint node docker-desktop  
app=blue:NoSchedule
```

NoSchedule pod with no tolerations
will not be scheduled

preferNoSchedule Softer version. it will
try not be schedule
But no guarantees

NoExecute Existing pods with no
tolerations will be evicted
immediately

UnTaint

```
kubectl taint node docker-desktop  
app=blue:NoSchedule-
```

Add Toleration

apiVersion: v1

kind: Pod

metadata:

name: httpd-pod

spec:

containers:

- name: httpd-container

image: httpd

tolerations:

- key: "app"
operator: "Equals"
value: "blue"
effect: "NoSchedule"





Taint/Untaint Node

```
kubectl taint node docker-desktop  
app=red :NoSchedule
```

```
Kubectl taint node docker-desktop  
app=red: NoSchedule-
```

```
Kubectl taint -- all node app=red:NoSchedule-
```

Add Tolerance

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: httpd-pod
```

```
spec:
```

```
  containers:
```

```
  - name: httpd-container
```

```
    image: httpd
```

```
  tolerations:
```

```
  - key: "app"
```

```
    operator: "Equals"
```

```
  - value: "red"
```

```
    effect: "NoSchedule"
```

