Agenda

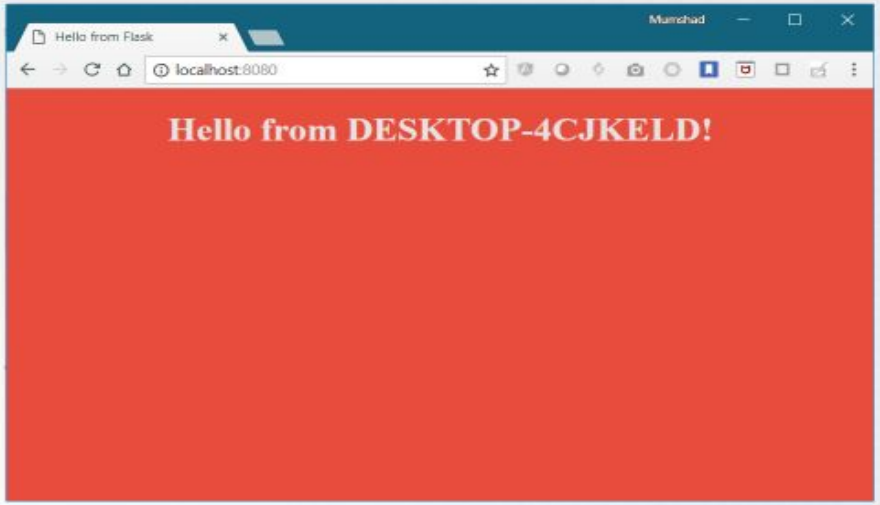
- Env variables
- Secrets
- Volumes
- Security

ENVIRONMENT VARIABLES

Environment Variables

app.py

```
import os
from flask import Flask

app = Flask(__name__)

…
…

color = "red"

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

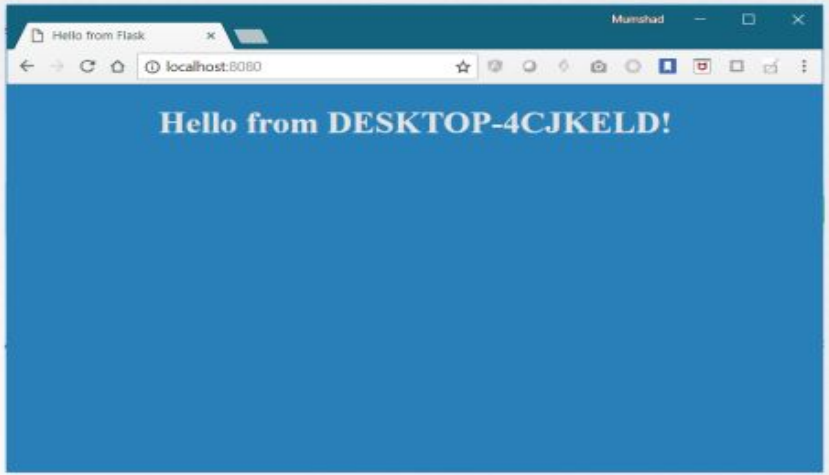



```
python app.py
```

Environment Variables

app.py

```
import os
from flask import Flask

app = Flask(__name__)

…
…

color = os.environ.get('APP_COLOR')

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```




```
export APP_COLOR=blue; python app.py
```

ENV Variables in Docker

```
docker run  -e APP_COLOR=blue    simple-webapp-color
```


```
docker run  -e APP_COLOR=green simple-webapp-color
```


```
docker run  -e APP_COLOR=pink simple-webapp-color
```

ENV Variables in Kubernetes

```
docker run  -e APP_COLOR=pink simple-webapp-color
```

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080

    env:
      - name: APP_COLOR
        value: pink
```

ENV Value Types

```
env:
  - name: APP_COLOR
    value: pink
```

```
env:
  - name: APP_COLOR
    valueFrom:
        configMapKeyRef:
```

```
env:
  - name: APP_COLOR
    valueFrom:
        secretKeyRef:
```

1 Plain Key Value

2 ConfigMap

3 Secrets

ConfigMaps

ConfigMap

```
APP_COLOR: blue
APP_MODE: prod
```

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080

    envFrom:
    - configMapRef:
          name: app-config
```

1 Create ConfigMap

2 Inject into Pod

Create ConfigMaps

ConfigMap

```
APP_COLOR: blue
APP_MODE: prod
```

Imperative

```
kubectl create configmap
    <config-name> --from-literal=<key>=<value>
```

```
kubectl create configmap \
    app-config --from-literal=APP_COLOR=blue \
               --from-literal=APP_MOD=prod
```

```
kubectl create configmap
    <config-name> --from-file=<path-to-file>
```

```
kubectl create configmap \
    app-config --from-file=app_config.properties
```

Create ConfigMap

Create ConfigMaps

ConfigMap

```
APP_COLOR: blue
APP_MODE: prod
```

Declarative

```
kubectl create -f
```

config-map.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_COLOR: blue
  APP_MODE: prod
```

Create ConfigMap

```
kubectl create -f config-map.yaml
```

View ConfigMaps

```
kubectl get configmaps
NAME          DATA      AGE
app-config    2         3s
```

```
kubectl describe configmaps
Name:         app-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
APP_COLOR:
----
blue
APP_MODE:
----
prod
Events:  <none>
```

ConfigMap in Pods

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    envFrom:
      - configMapRef:
          name: app-config
```

config-map.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_COLOR: blue
  APP_MODE: prod
```


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

ConfigMap in Pods

```yaml
envFrom:
  - configMapRef:
      name: app-config
```

ENV

SINGLE ENV

```yaml
env:
  - name: APP_COLOR
    valueFrom:
      configMapKeyRef:
        name: app-config
        key: APP_COLOR
```

```yaml
volumes:
- name: app-config-volume
  configMap:
    name: app-config
```

VOLUME

Kubernetes Secrets

Web-MySQL Application

app.py

```
import os
from flask import Flask

app = Flask(__name__)


@app.route("/")
def main():

    mysql.connector.connect(host='mysql', database='mysql',
                            user='root', password='paswrd')

    return render_template('hello.html', color=fetchcolor())


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

Web-MySQL Application

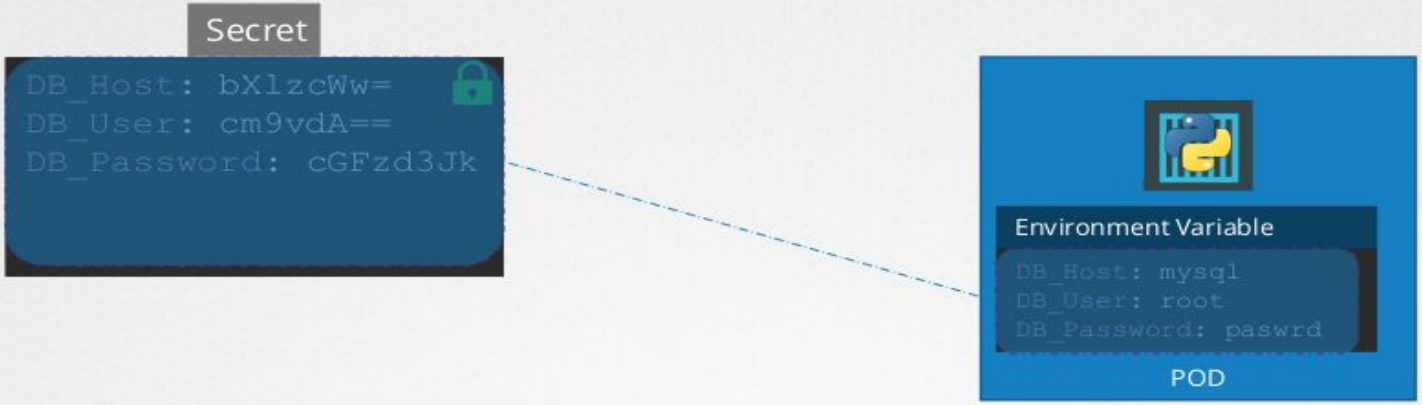
app.py

```
import os
from flask import Flask

app = Flask(__name__)


@app.route("/")
def main():

    mysql.connector.connect(host='mysql', database='mysql',
                            user='root', password='paswrd')

    return render_template('hello.html', color=fetchcolor())


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

config-map.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  DB_Host: mysql
  DB_User: root
  DB_Password: paswrd
```

Secret



1
Create Secret

2
Inject into Pod

Create Secrets

Secret

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

Imperative

```
kubectl create secret generic
    <secret-name> --from-literal=<key>=<value>
```

```
kubectl create secret generic \
    app-secret --from-literal=DB_Host=mysql      \
               --from-literal=DB_User=root
               --from-literal=DB_Password=paswrd
```

```
kubectl create secret generic
    <secret-name> --from-file=<path-to-file>
```

```
kubectl create secret generic \
    app-secret --from-file=app_secret.properties
```

Create Secret

Create Secrets

Secret

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

Declarative

```
kubectl create -f
```

secret-data.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzd3Jk
```

```
kubectl create -f secret-data.yaml
```

Create Secret

Encode Secrets

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

```
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
echo -n 'mysql' | base64
bXlzcWw=
```

```
echo -n 'root' | base64
cm9vdA==
```

```
echo -n 'paswrd' | base64
cGFzd3Jk
```

View Secrets

```
kubectl get secrets
```

```
NAME                  TYPE                                  DATA      AGE
app-secret            Opaque                                3         10m
default-token-mvtkv   kubernetes.io/service-account-token   3         2h
```

```
kubectl describe secrets
```

```
Name:         app-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
DB_Host:      10 bytes
DB_Password:  6 bytes
DB_User:      4 bytes
```

```
kubectl get secret app-secret -o yaml
```

```
apiVersion: v1
data:
  DB_Host: bXlzcWw=
  DB_Password: cGFzd3Jk
  DB_User: cm9vdA==
kind: Secret
metadata:
  creationTimestamp: 2018-10-18T10:01:12Z
  labels:
    name: app-secret
  name: app-secret
  namespace: default
uid: be96e989-d2bc-11e8-a545-080027931072
type: Opaque
```

Decode Secrets

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

```
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
echo -n 'bXlzcWw=' | base64 --decode
mysql
```

```
echo -n 'cm9vdA==' | base64 --decode
root
```

```
echo -n 'cGFzd3Jk' | base64 --decode
paswrd
```

Secrets in Pods

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    envFrom:
      - secretRef:
          name: app-secret
```

secret-data.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzd3Jk
```

Inject into Pod

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

Secrets in Pods

```yaml
envFrom:
  - secretRef:
      name: app-config
```

ENV

SINGLE ENV

```yaml
env:
  - name: DB_Password
    valueFrom:
      secretKeyRef:
        name: app-secret
        key: DB_Password
```

```yaml
volumes:
- name: app-secret-volume
  secret:
    secretName: app-secret
```

VOLUME

Secrets in Pods as Volumes

```
volumes:
- name: app-secret-volume
  secret:
    secretName: app-secret
```

VOLUME

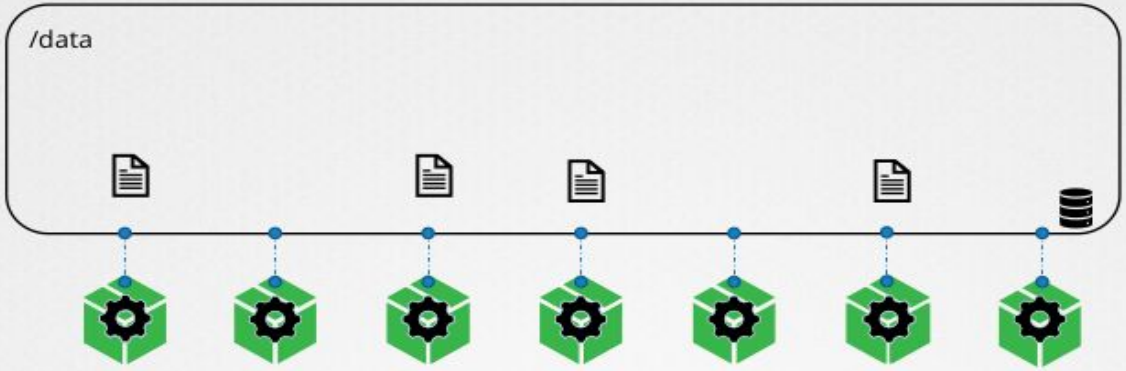
```
ls /opt/app-secret-volumes
DB_Host      DB_Password  DB_User
```

```
cat /opt/app-secret-volumes/DB_Password
paswrd
```

Inside the Container

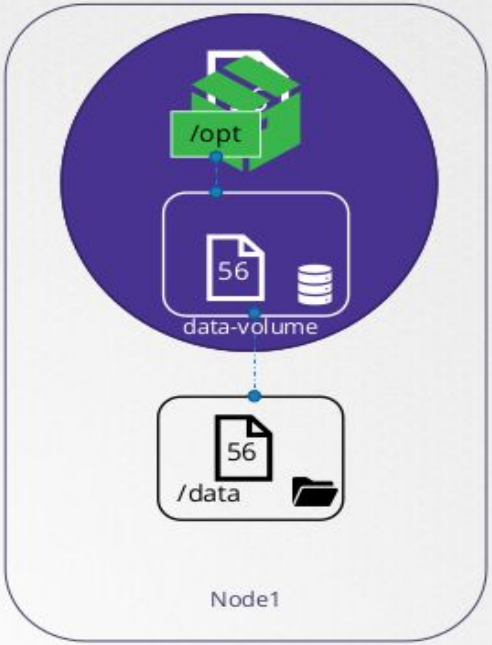
Volumes

Volume

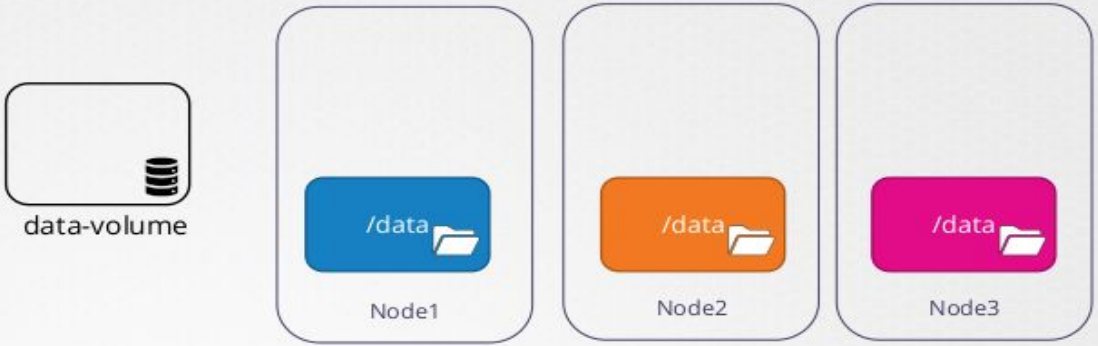


Volumes & Mounts

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



Volume Types

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

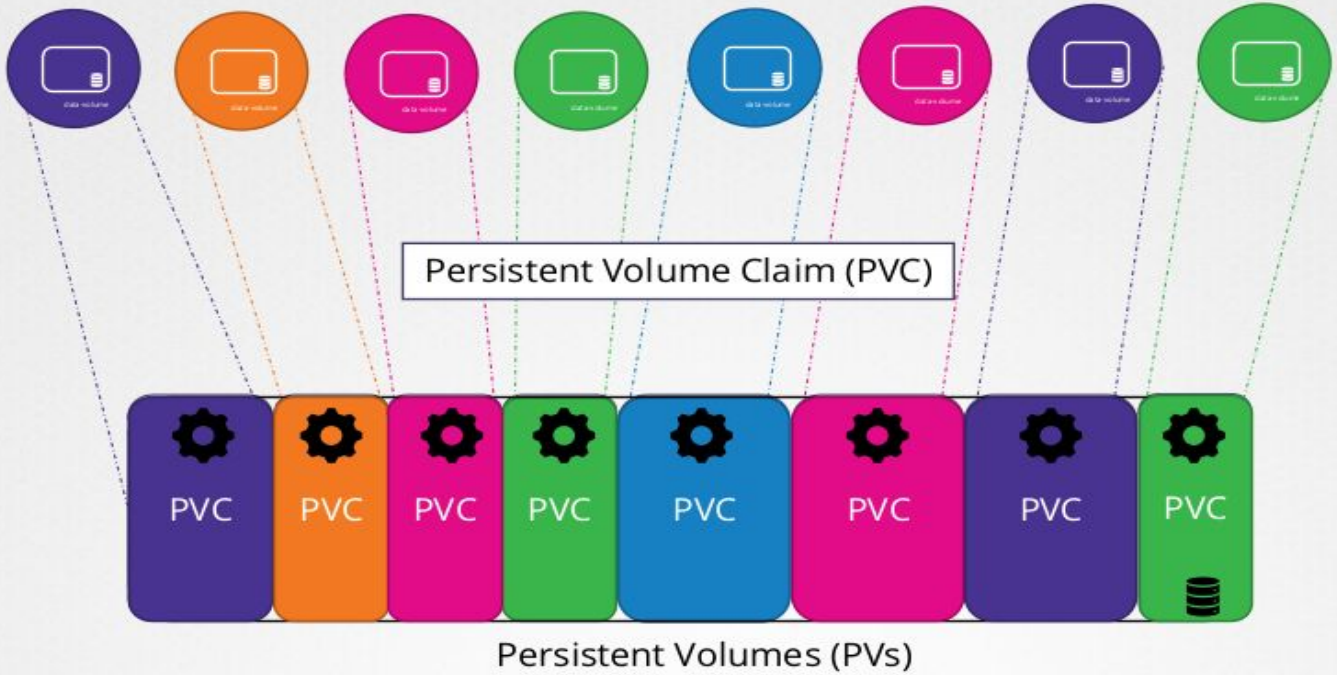
Volume Types

```
volumes:
- name: data-volume
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

data-volume

Persistent Volumes

Persistent Volume



Persistent Volume

pv-definition.yaml

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

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

```
kubectl get persistentvolume
```

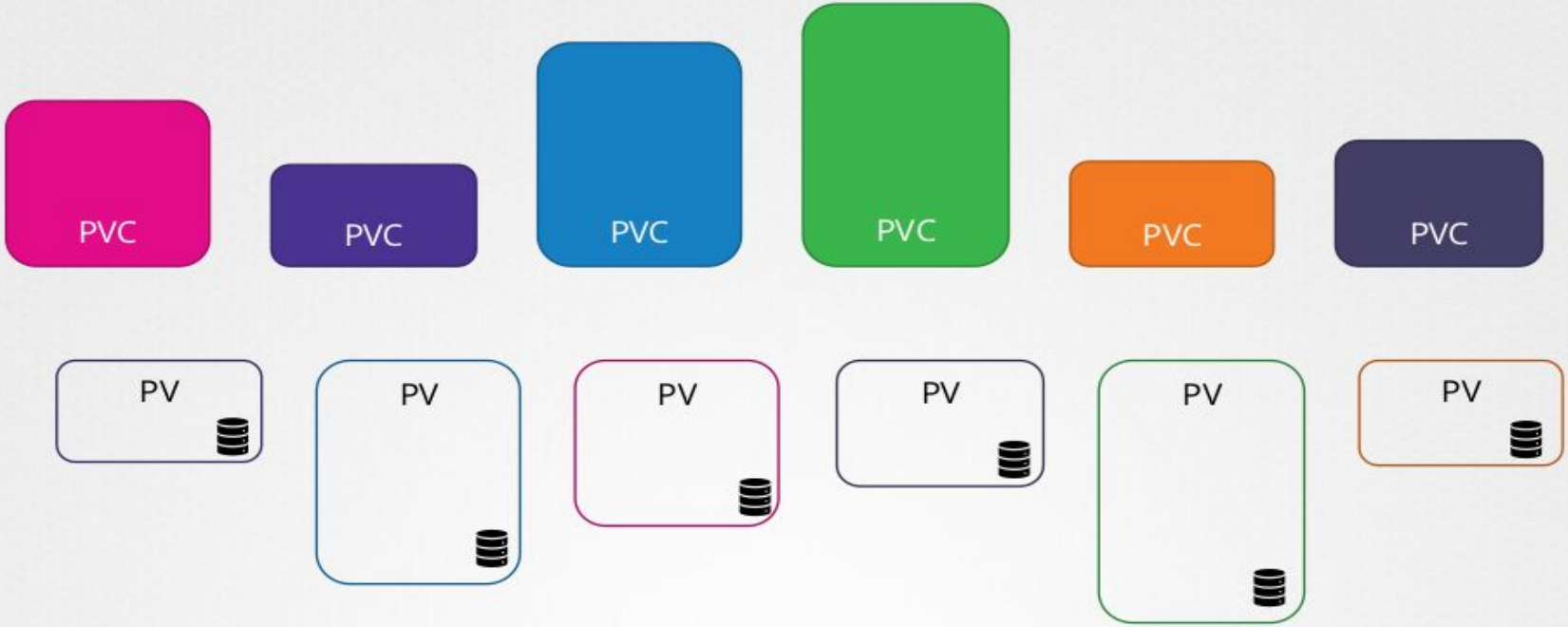
NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv-vol1	1Gi	RWO	Retain	Available				3m

ReadOnlyMany
ReadWriteOnce
ReadWriteMany

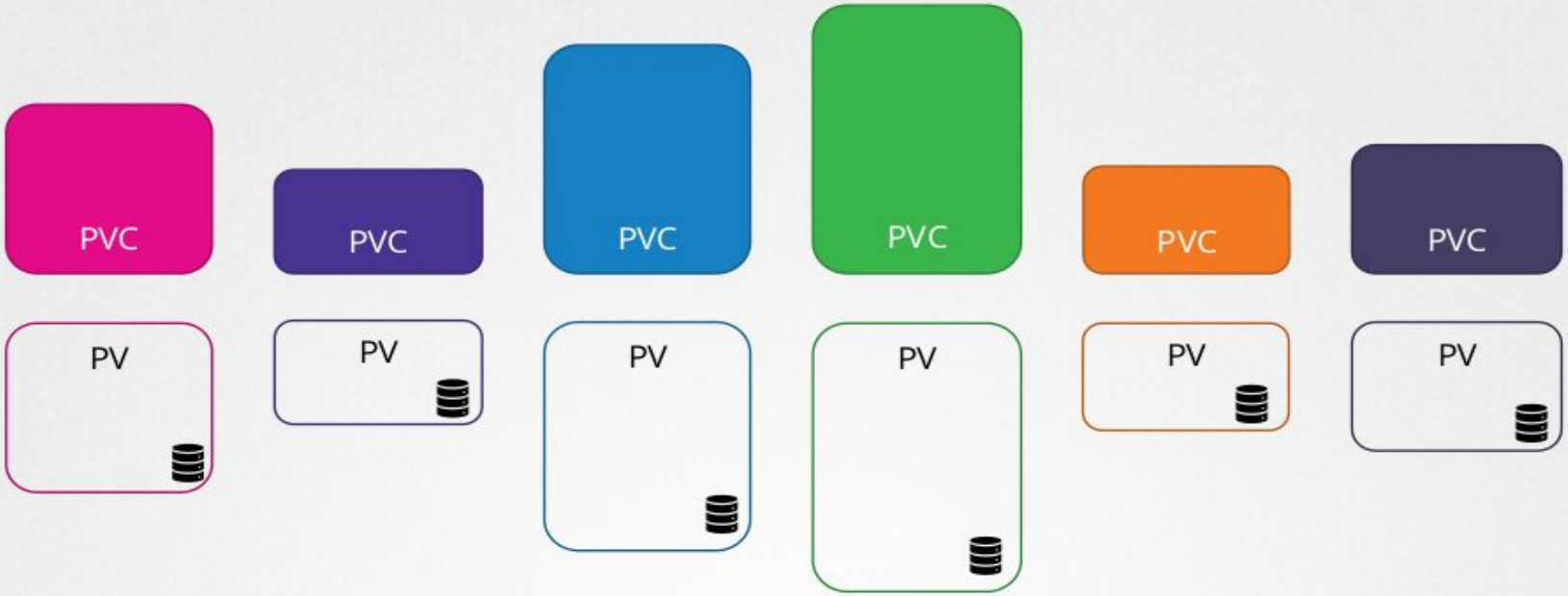
Persistent Volume (PV)

Persistent Volume Claims

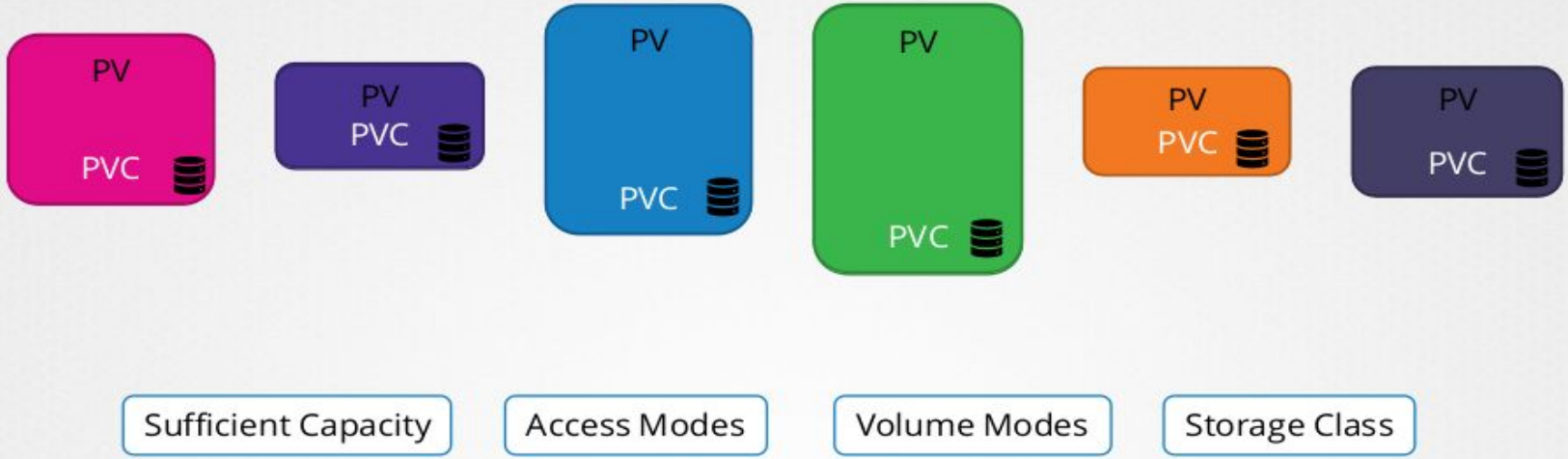
Persistent Volume Claim



Binding



Binding


PV
PVC
PV
PVC
PV
PVC
PV
PVC
PV
PVC
PV
PVC
Sufficient Capacity
Access Modes
Volume Modes
Storage Class

Binding

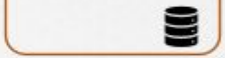
PVC

```
selector:
  matchLabels:
    name: my-pv
```

PV

```
labels:
  name: my-pv
```

PV



Sufficient Capacity | Access Modes | Volume Modes | Storage Class | Selector

Binding

Pending

Sufficient Capacity | Access Modes | Volume Modes | Storage Class | Selector

Persistent Volume Claim

pvc-definition.yaml

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

  resources:
    requests:
      storage: 500Mi
```

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

```
kubectl get persistentvolumeclaim
```

```
NAME        STATUS    VOLUME    CAPACITY    ACCESS MODES
myclaim     Pending
```

Persistent Volume Claim

pvc-definition.yaml

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

  resources:
     requests:
       storage: 500Mi
```

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

pv-definition.yaml

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

View PVCs

```
kubectl get persistentvolumeclaim
```

```
NAME        STATUS    VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
myclaim     Bound     pv-vol1   1Gi        RWO                           43m
```

Delete PVCs

```
kubectl delete persistentvolumeclaim myclaim
```

```
persistentvolumeclaim "myclaim" deleted
```

STORAGE CLASSES

PV and PVCs

pv-definition.yaml

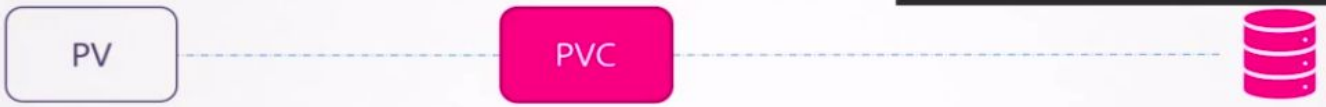
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-vol1
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 500Mi
  gcePersistentDisk:
    pdName: pd-disk
    fsType: ext4
```

pvc-definition.yaml

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

pod-definition.yaml

```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
    persistentVolumeClaim:
      claimName: myclaim
```



Static Provisioning

```
gcloud beta compute disks create \
    --size 1GB
    --region us-east1
    pd-disk
```

pv-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-vol1
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 500Mi
  gcePersistentDisk:
    pdName: pd-disk
    fsType: ext4
```

PV

Dynamic Provisioning

pv-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
   name: pv-vol1
spec:
  accessModes:
      - ReadWriteOnce
  capacity:
      storage: 500Mi
  gcePersistentDisk:
     pdName: pd-disk
     fsType: ext4
```

PV

sc-definition.yaml

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: google-storage
provisioner: kubernetes.io/gce-pd
```

SC

Dynamic Provisioning

pvc-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-storage
  resources:
    requests:
      storage: 500Mi
```

pod-definition.yaml

```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/
    volumeMounts:
    - mountPath: /opt
      name: data-volume
  volumes:
  - name: data-volume
    persistentVolumeClaim:
      claimName: myclaim
```

sc-definition.yaml

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: google-storage
provisioner: kubernetes.io/gce-pd
```

SC

PV

PVC

SECURITY PRIMITIVES

Secure Kubernetes

kube-apiserver

Who can access?

What can they do?

Authentication

Who can access?

- Files – Username and Passwords
- Files – Username and Tokens
- Certificates
- External Authentication providers - LDAP
- Service Accounts

Auth Mechanisms - Basic

Static Password File

user-details.csv

```
password123,user1,u0001,group1
password123,user2,u0002,group1
password123,user3,u0003,group2
password123,user4,u0004,group2
password123,user5,u0005,group2
```

Static Token File

user-token-details.csv

```
KpjCVbI7rCFAHYPkByTIzRb7gu1cUc4B,user10,u0010,group1
rJjncHmvtXHc6MlWQddhtvNyyhgTdxSC,user11,u0011,group1
mjpOFIEiFOkL9toikaRNtt59ePtczZSq,user12,u0012,group2
PG41IXhs7QjqwWkmBkvgGT9glOyUqZij,user13,u0013,group2
```

```
--token-auth-file=user-details.csv
```

```
curl -v -k https://master-node-ip:6443/api/v1/pods --header "Authorization: Bearer KpjCVbI7rCFAHYPkBzRb7gu1cUc4B"
```

```
curl https://my-kube-playground:6443/api/v1/pods \
  --key admin.key
  --cert admin.crt
  --cacert ca.crt
```

```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
  },
  "items": []
}
```

```
kubectl get pods
      --server my-kube-playground:6443
      --client-key admin.key
      --client-certificate admin.crt
      --certificate-authority ca.crt
```

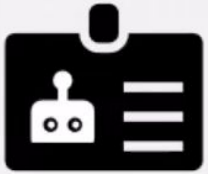
```
No resources found.
```

SERVICE ACCOUNTS

User

Admin

Developer



Service

Prometheus

Jenkins

```
kubectl create serviceaccount dashboard-sa
```

```
serviceaccount “dashboard-sa” created
```

```
kubectl get serviceaccount
```

```
NAME                  SECRETS   AGE
default               1         218d
dashboard-sa          1         4d
```

```
kubectl describe serviceaccount dashboard-sa
```

```
Name:                dashboard-sa
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   dashboard-sa-token-kbbdm
Tokens:              dashboard-sa-token-kbbdm
Events:              <none>
```

```
kubectl describe serviceaccount dashboard-sa

Name:                dashboard-sa
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   dashboard-sa-token-kbbdm
Tokens:              dashboard-sa-token-kbbdm
Events:              <none>
```

```
kubectl describe secret dashboard-sa-token-kbbdm

Name:         dashboard-sa-token-kbbdm
Namespace:    default
Labels:       <none>

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1025 bytes
namespace:  7 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL
3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3V
udC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3
```

Secret

```
token:
eyJhbGciOiJSUzI1NiIsImtp
ZCI6IiJ9.eyJpc3MiOiJrdWJ
lcm5ldGVzL3NlcnZpY2VhY2N
vdW50Iiwia3ViZXJuZXRlcy5
pby9zZXJ2aWNlYWNjb3Vud….
```

```
kubectl get serviceaccount
```

```
NAME                 SECRETS   AGE
default              1         218d
dashboard-sa         1         4d
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-kubernetes-dashboard
spec:
  containers:
    - name: my-kubernetes-dashboard
      image: my-kubernetes-dashboard
```

```
kubectl describe pod my-kubernetes-dashboard
```

```
Name:         my-kubernetes-dashboard
Namespace:    default
Annotations:  <none>
Status:       Running
IP:           10.244.0.15
Containers:
  nginx:
    Image:          my-kubernetes-dashboard
Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-j4hkv (ro)
Conditions:
  Type           Status
Volumes:
  default-token-j4hkv:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-j4hkv
    Optional:    false
```

```
kubectl describe pod my-kubernetes-dashboard
```

```
Name:         my-kubernetes-dashboard
Namespace:    default
Annotations:  <none>
Status:       Running
IP:           10.244.0.15
Containers:
  nginx:
    Image:          my-kubernetes-dashboard
Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-j4hkv (ro)
Conditions:
  Type           Status
Volumes:
  default-token-j4hkv:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-j4hkv
    Optional:    false
```

```
kubectl exec -it my-kubernetes-dashboard ls /var/run/secrets/kubernetes.io/serviceaccount
```

```
ca.crt  namespace  token
```

```
kubectl exec -it my-kubernetes-dashboard cat /var/run/secrets/kubernetes.io/serviceaccount/token
```


```
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3V
udC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRlZmF1bHQtdG9rZW4tajRoa3Y
iLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWF
jY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjcxZGM4YWExLTU2MGMtMTFlOC04YmI0LTA4MDAyNzkzMTA3MiIsInN1YiI6InN5c3RlbTpzZXJ2aWN
```

```
kubectl get serviceaccount
```

```
NAME                 SECRETS   AGE
default              1         218d
dashboard-sa         1         4d
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-kubernetes-dashboard
spec:
  containers:
    - name: my-kubernetes-dashboard
      image: my-kubernetes-dashboard
  serviceAccountName: dashboard-sa
```

```
kubectl describe pod my-kubernetes-dashboard
```

```
Name:         my-kubernetes-dashboard
Namespace:    default
Annotations:  <none>
Status:       Running
IP:           10.244.0.15
Containers:
  nginx:
    Image:          my-kubernetes-dashboard
Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from dashboard-sa-token-kbbdm (ro)
Conditions:
  Type           Status
Volumes:
  dashboard-sa-token-kbbdm:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  dashboard-sa-token-kbbdm
    Optional:    false
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-kubernetes-dashboard
spec:
  containers:
    - name: my-kubernetes-dashboard
      image: my-kubernetes-dashboard
  automountServiceAccountToken: false
```

Security
KUBECONFIG

config $HOME/.kube/config

KubeConfig File

```
--server my-kube-playground:6443
--client-key admin.key
--client-certificate admin.crt
--certificate-authority ca.crt
```

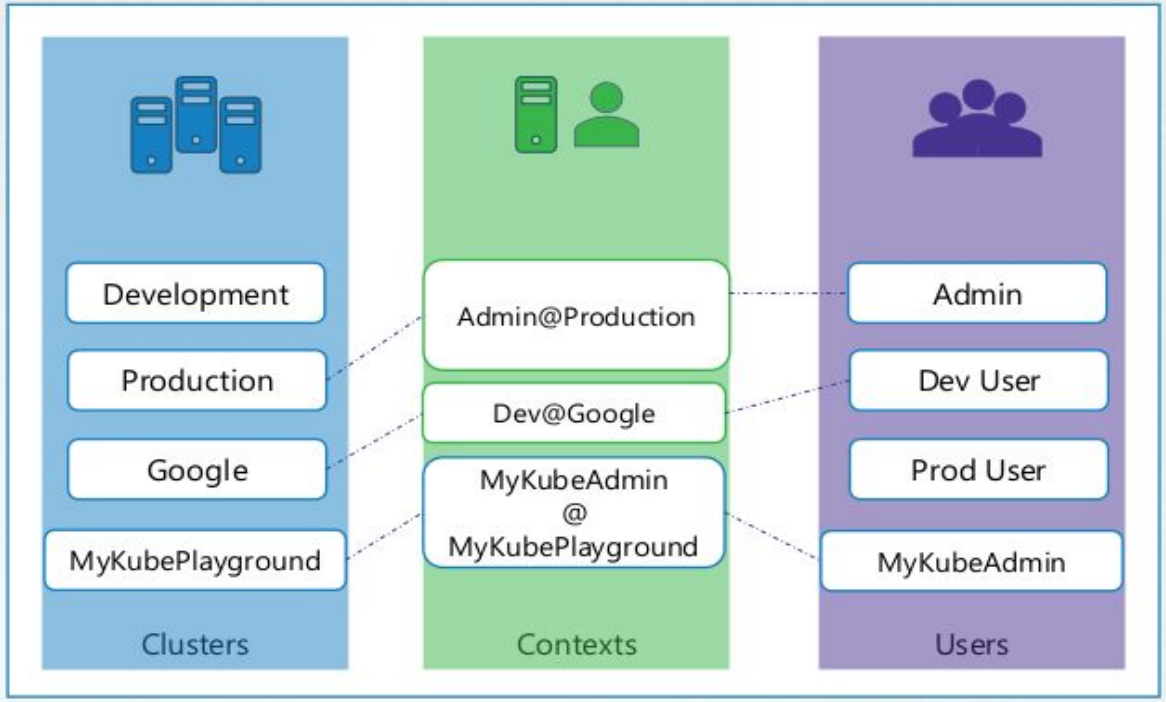
```
kubectl get pods
  --kubeconfig config
```

```
No resources found.
```

KubeConfig File

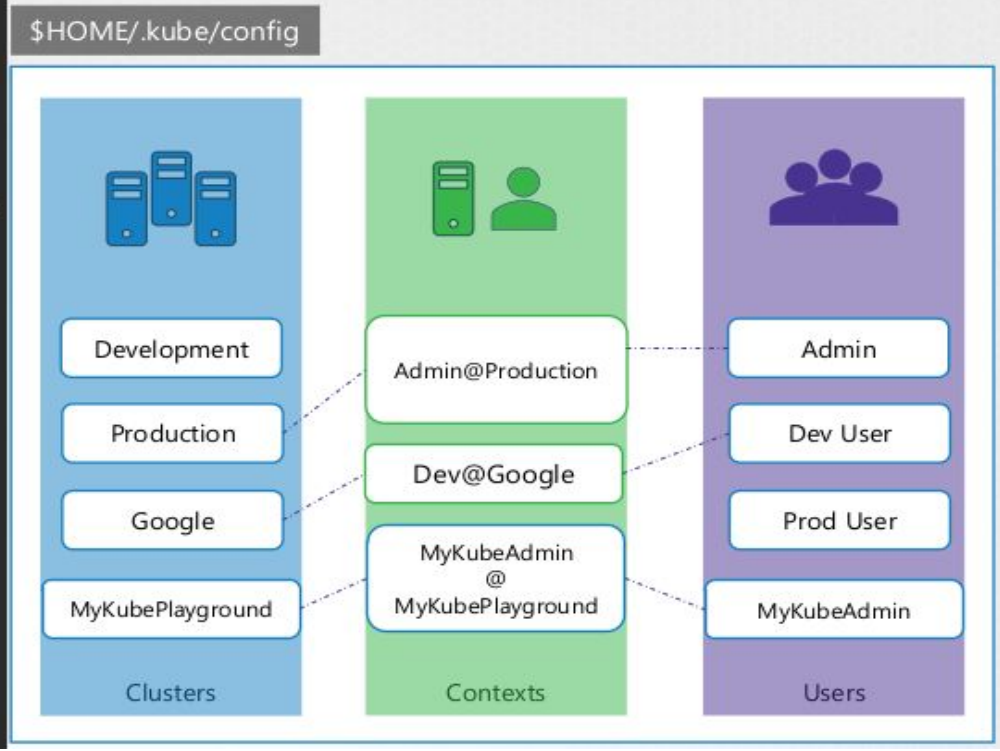
$HOME/.kube/config

```
--server my-kube-playground:6443
--client-key admin.key
--client-certificate admin.crt
--certificate-authority ca.crt
```



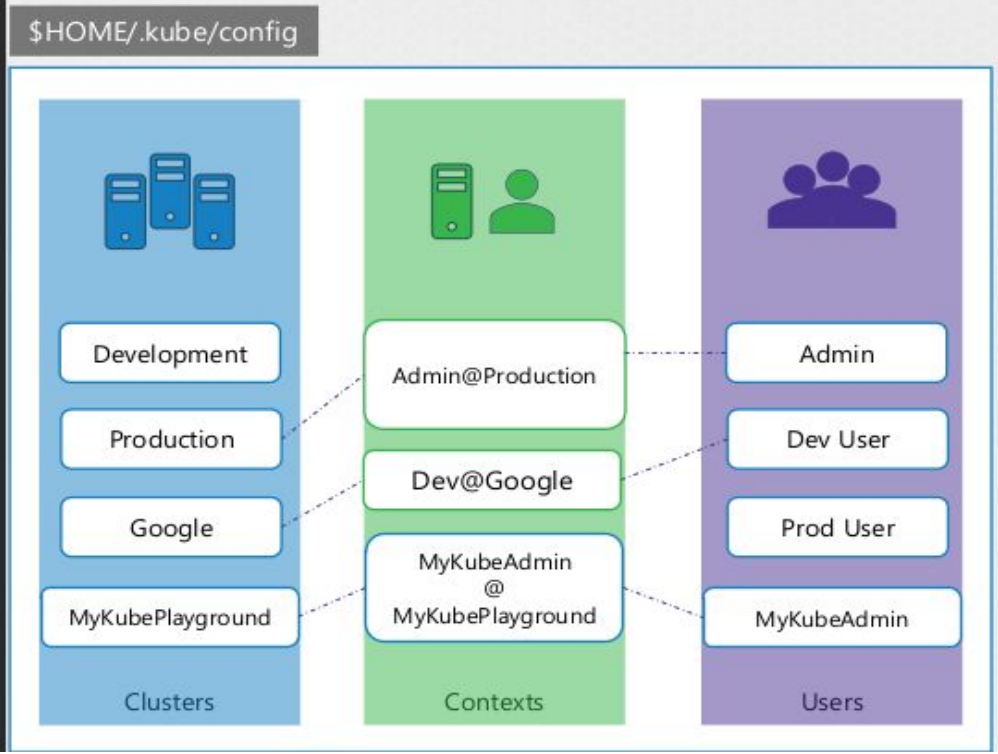
KubeConfig File

```yaml
apiVersion: v1
kind: Config

clusters:

- name: my-kube-playground
  cluster:
    certificate-authority: ca.crt
    server: https://my-kube-playground:6443


contexts:

- name: my-kube-admin@my-kube-playground
  context:
    cluster:
    user:



users:

- name: my-kube-admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```



KubeConfig File

```yaml
apiVersion: v1
kind: Config
current-context: dev-user@google

clusters:
- name: my-kube-playground   (values hidden...)
- name: development
- name: production
- name: google

contexts:
- name: my-kube-admin@my-kube-playground
- name: dev-user@google
- name: prod-user@production

users:
- name: my-kube-admin
- name: admin
- name: dev-user
```



Kubectl config

```
kubectl config view
```

```
apiVersion: v1

kind: Config
current-context: kubernetes-admin@kubernetes

clusters:
- cluster:
    certificate-authority-data: REDACTED
    server: https://172.17.0.5:6443
  name: kubernetes

contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes


users:
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
```

```
kubectl config view –kubeconfig=my-custom-config
```

```
apiVersion: v1

kind: Config
current-context: my-kube-admin@my-kube-playground


clusters:
- name: my-kube-playground
- name: development
- name: production


contexts:
-  name: my-kube-admin@my-kube-playground
-  Name: prod-user@production


users:
- name: my-kube-admin
- name: prod-user
```

KODEKLOUD

Kubectl config

```
kubectl config view
```

```
apiVersion: v1

kind: Config
current-context: my-kube-admin@my-kube-playground


clusters:
- name: my-kube-playground
- name: development
- name: production


contexts:
-  name: my-kube-admin@my-kube-playground
-  Name: prod-user@production


users:
- name: my-kube-admin
- name: prod-user
```

```
kubectl config use-context prod-user@production
```

```
apiVersion: v1

kind: Config
current-context: prod-user@production


clusters:
- name: my-kube-playground
- name: development
- name: production


contexts:
-  name: my-kube-admin@my-kube-playground
-  Name: prod-user@production


users:
- name: my-kube-admin
- name: prod-user
```

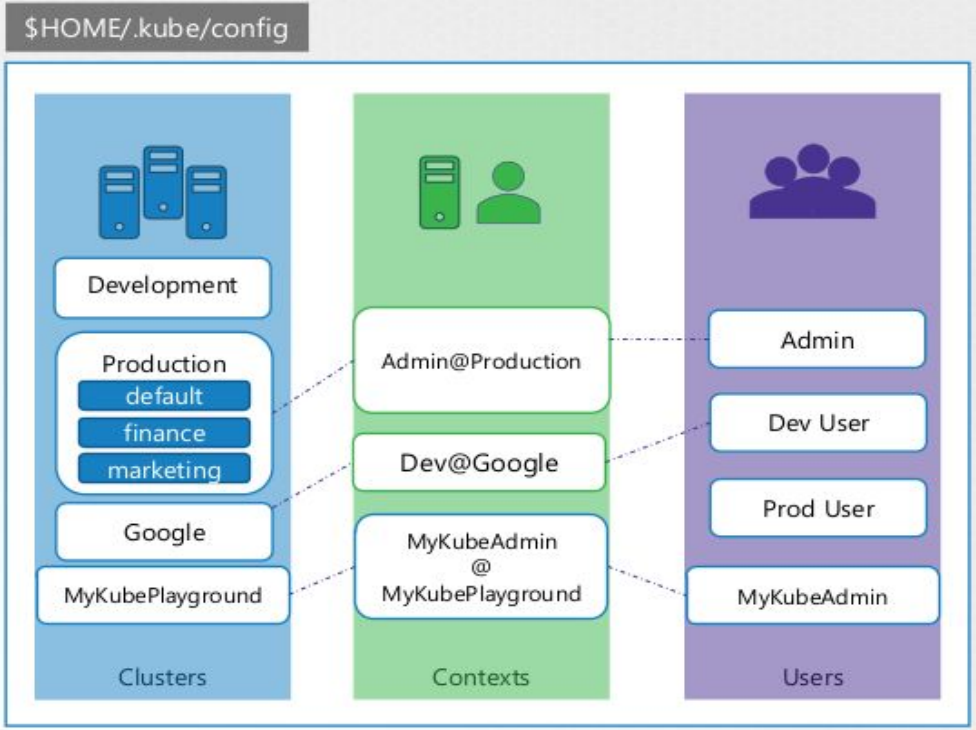
Kubectl config

```
kubectl config -h
```

```
Available Commands:
  current-context Displays the current-context
  delete-cluster  Delete the specified cluster from the kubeconfig
  delete-context  Delete the specified context from the kubeconfig
  get-clusters    Display clusters defined in the kubeconfig
  get-contexts    Describe one or many contexts
  rename-context  Renames a context from the kubeconfig file.
  set             Sets an individual value in a kubeconfig file
  set-cluster     Sets a cluster entry in kubeconfig
  set-context     Sets a context entry in kubeconfig
  set-credentials Sets a user entry in kubeconfig
  unset           Unsets an individual value in a kubeconfig file
  use-context     Sets the current-context in a kubeconfig file
  view            Display merged kubeconfig settings or a specified kubeconfig file
```

Namespaces

```yaml
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: ca.crt
    server: https://172.17.0.51:6443

contexts:
- name: admin@production
  context:
    cluster: production
    user: admin
    namespace: finance

users:
- name: admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```



KODEKLOUD

References:

- https://www.udemy.com/course/certified-kubernetes-administrator-with-practice-tests
- https://www.udemy.com/course/certified-kubernetes-application-developer
- https://kubernetes.io/docs