TEL4VN

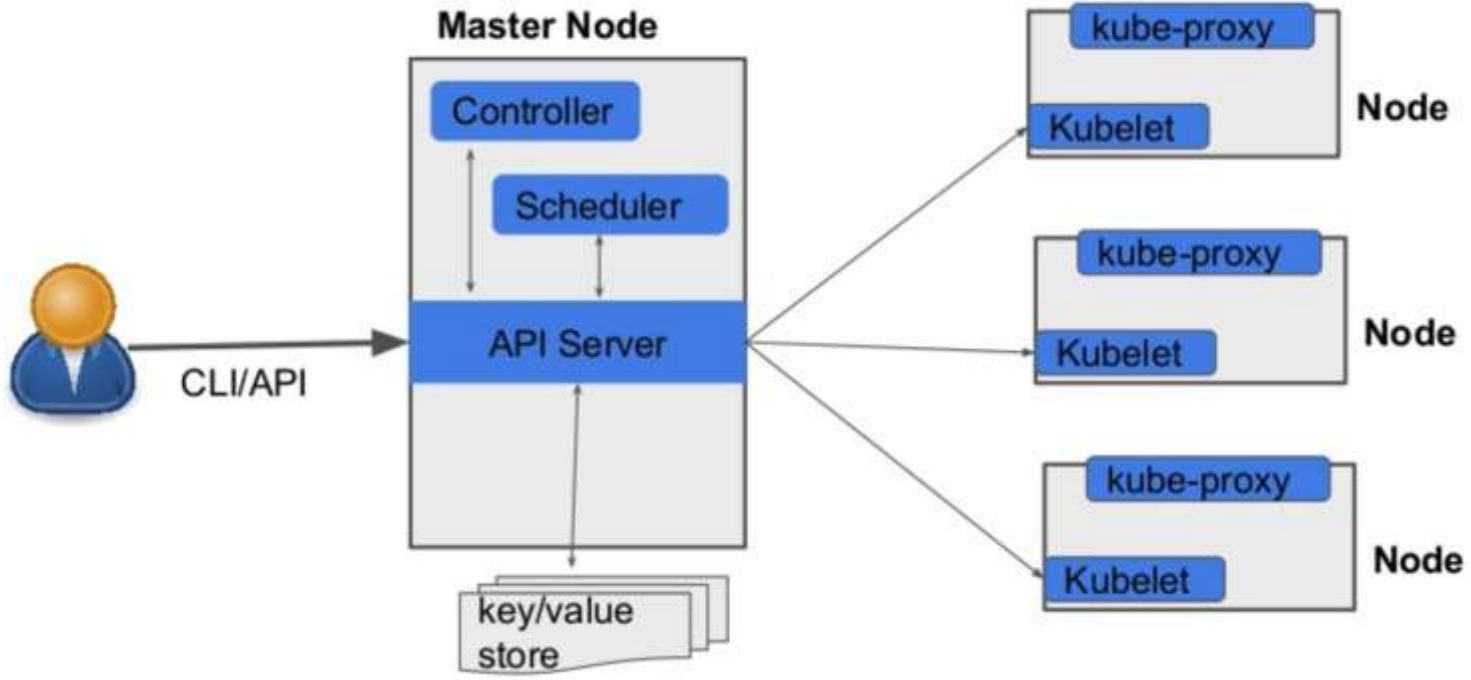
Securing your cluster

THAOLUONG
09/2020

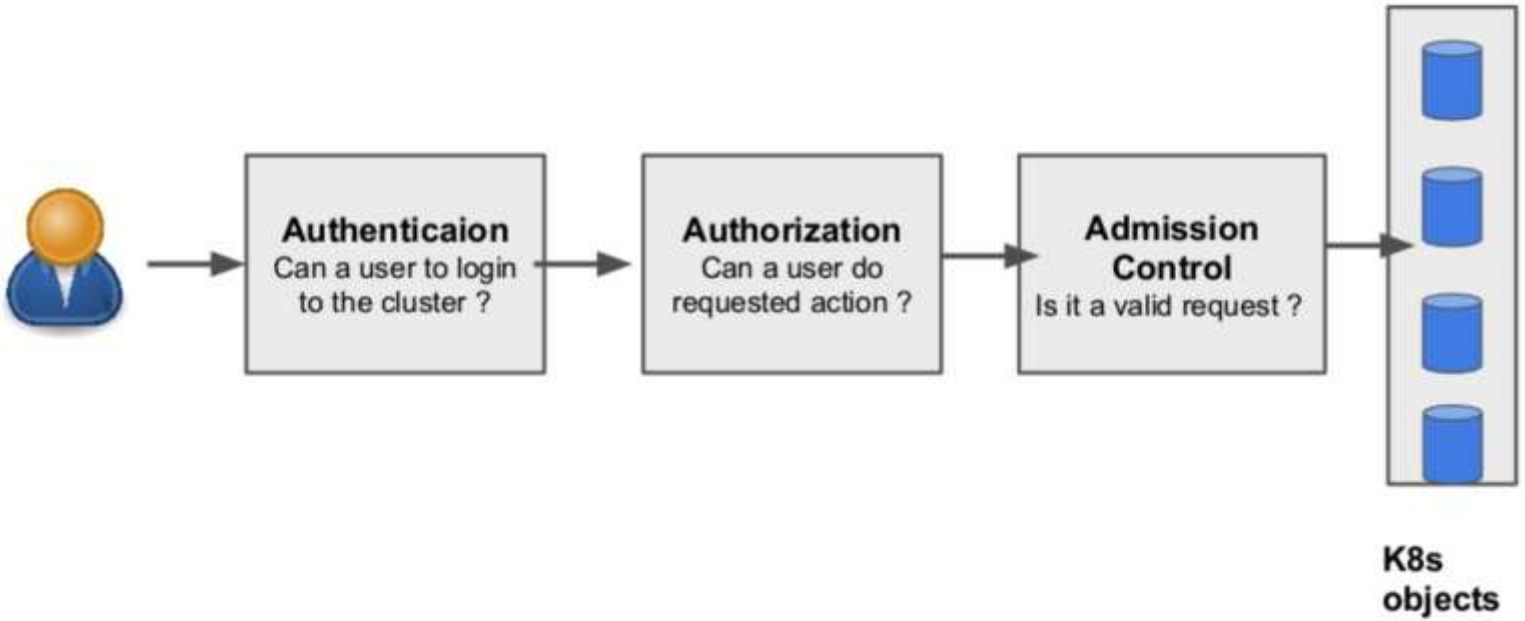
Content

- ❑ API request
- ❑ RBAC
- ❑ Kube config
- ❑ TLS certificate
- ❑ Network Policy

Kubernetes Architecture



Kubernetes API request



Authentication

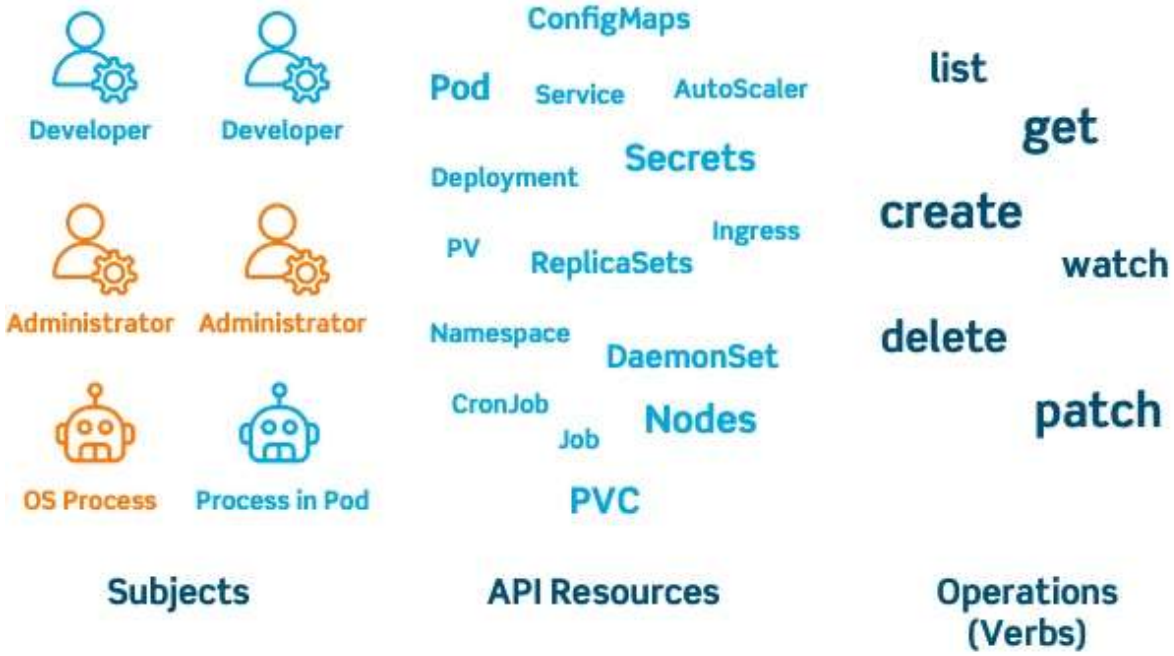
Who can access ?

- Files – Username and Password
- Certificates
- External Authentication providers – LDAP
- Service Account

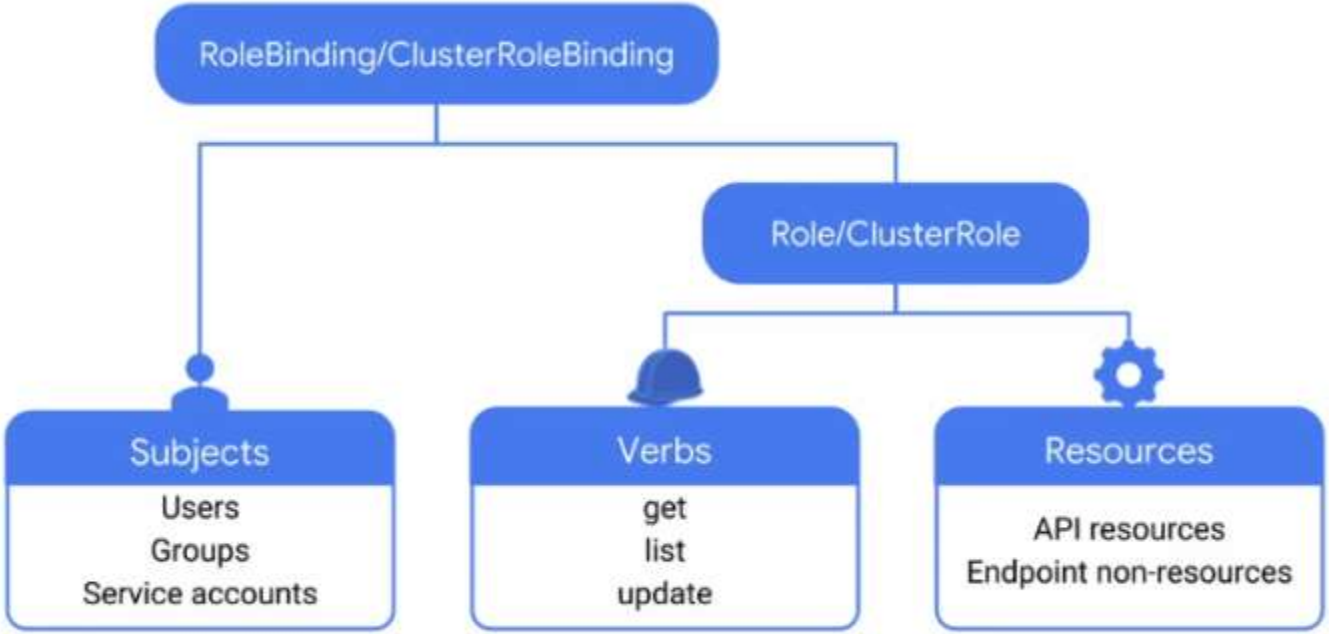
Authorization

- RBAC Authorization
- ABAC Authorization
- Node Authorization
- Webhook mode

Role-Based Access Control



RBAC Structure



No User Management In Kubernetes

Expectation:

~~kubectl create user john~~

~~kubectl create group admin~~

~~kubectl add john to admin~~

How to manage user? User Plugin

- Certificate Based Authentication (X509)
- Token Based Authentication
- Basic Authentication
- OAUTH2: OIDC

Service Account

Accessing the API Server From a Pod

For example, some applications might need to know:

- The status of the cluster's nodes.
- The namespaces available.
- The Pods running in the cluster, or in a specific namespace.
- And other things like that.

RBAC - Roles

Role

"Applicable to a given namespace only."

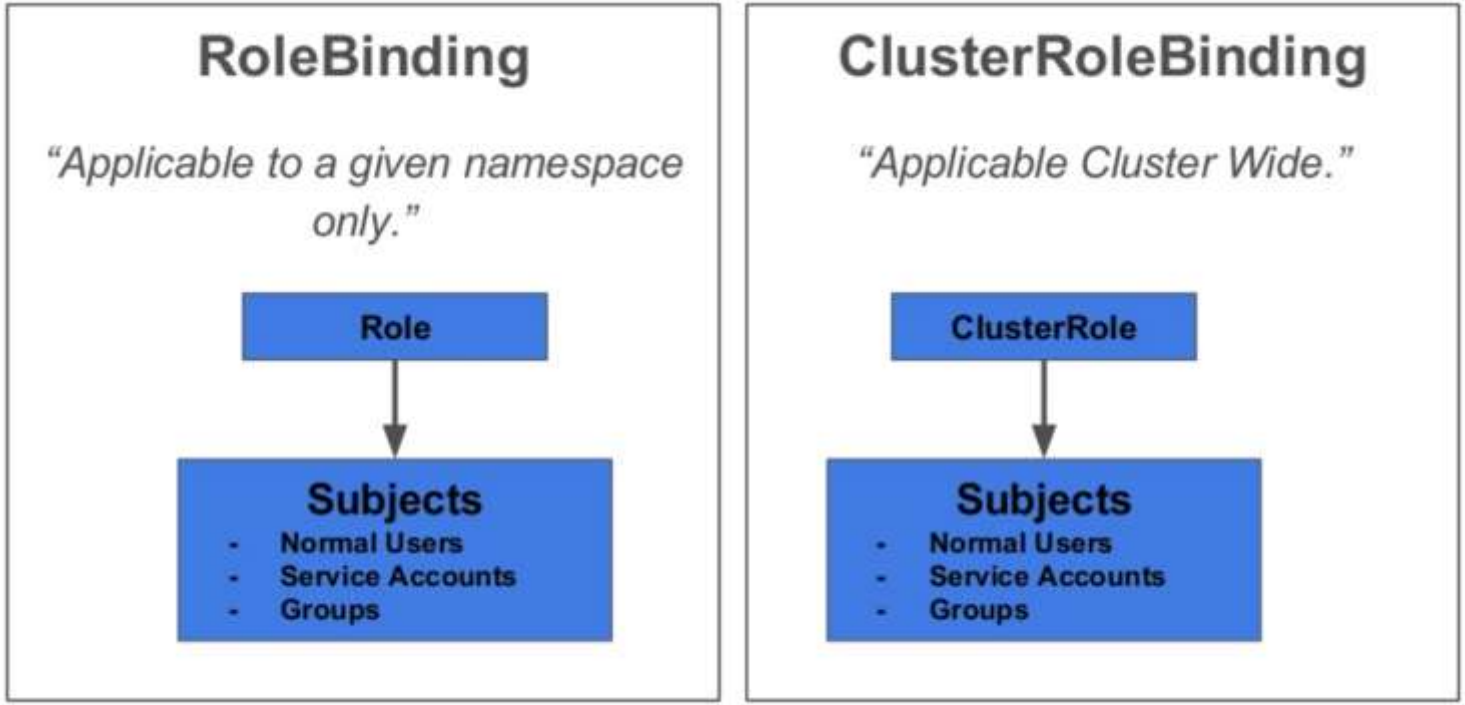
```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: cloudyuga
  name: deployment-manager
rules:
- apiGroups: ["", "apps"]
  resources: ["deployments", "replicasets", "pods"]
  verbs: ["get", "list", "watch", "create", "update"]
```

ClusterRole

"Applicable Cluster Wide."

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployment-manager-cluster
rules:
- apiGroups: ["", "apps"]
  resources: ["deployments", "replicasets", "pods"]
  verbs: ["get", "list", "watch", "create", "update"]
```

RBAC - Roles Binding



RBAC - Roles Binding

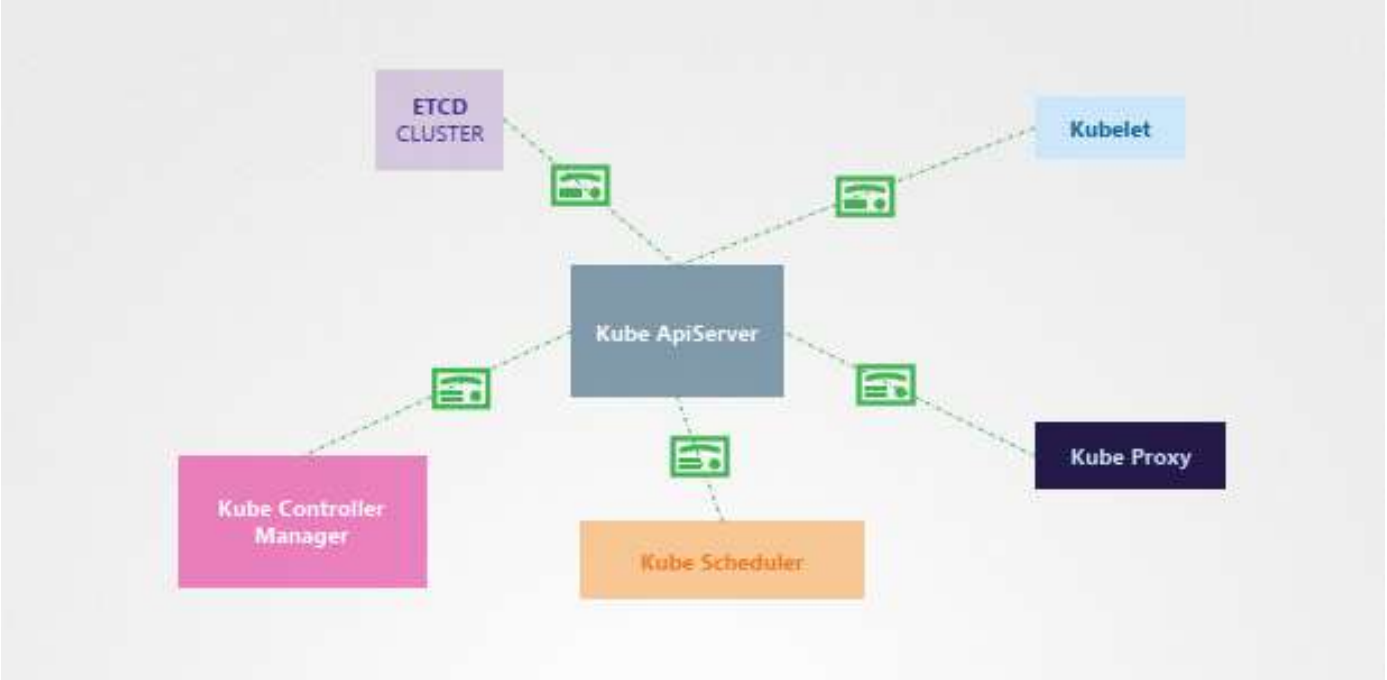
RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployment-manager-binding
  namespace: cloudyuga
subjects:
- kind: User
  name: nkhare
  apiGroup: "rbac.authorization.k8s.io"
roleRef:
  kind: Role
  name: deployment-manager
  apiGroup: "rbac.authorization.k8s.io"
```

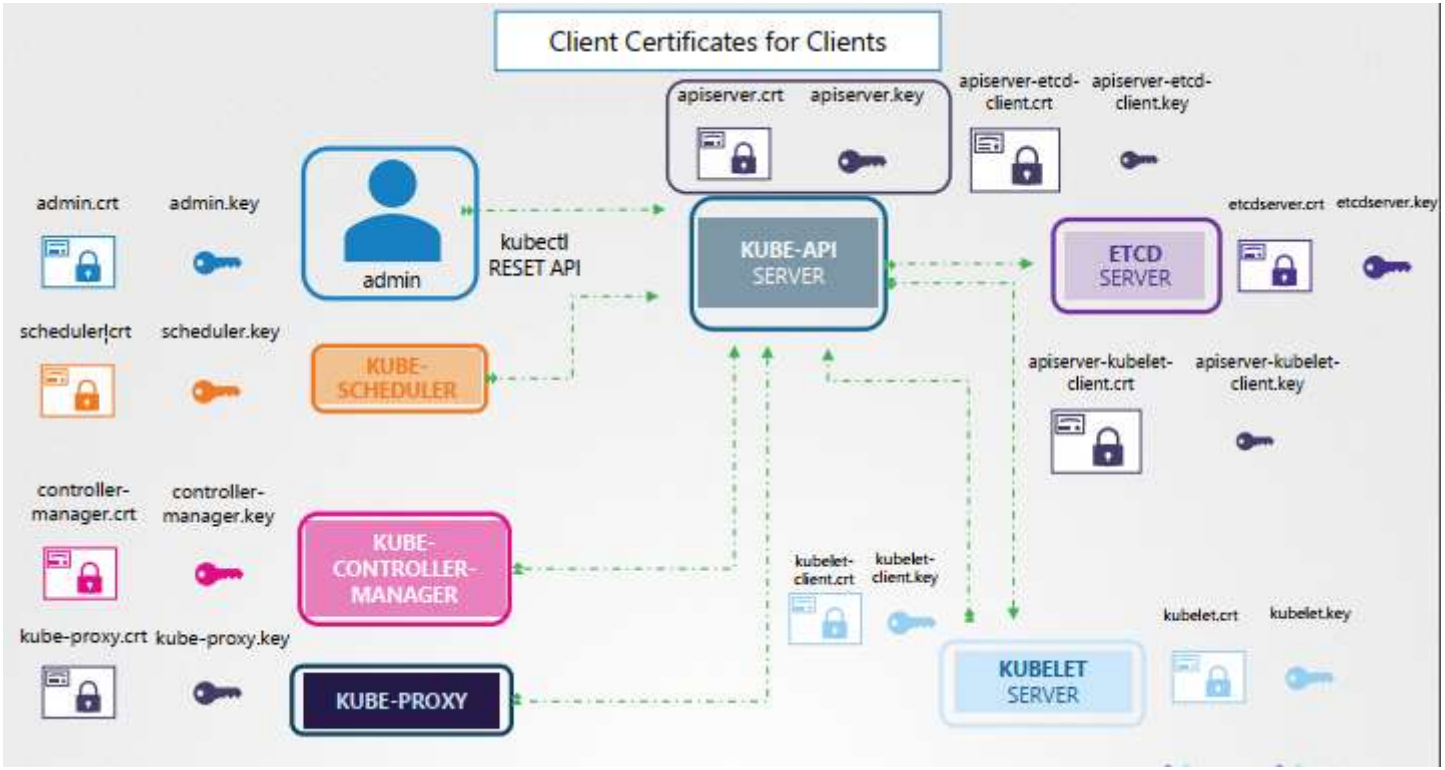
ClusterRoleBinding

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cluster-manager-binding
subjects:
- kind: User
  name: nkhare
  apiGroup: "rbac.authorization.k8s.io"
roleRef:
  kind: ClusterRole
  name: deployment-manager-cluster
  apiGroup: "rbac.authorization.k8s.io"
```

TLS certificate



TLS


Client Certificates for Clients
apiserver.crt
apiserver.key
apiserver-etcd-client.crt
apiserver-etcd-client.key
admin.crt
admin.key
admin
kubectl RESET API
KUBE-API SERVER
ETCD SERVER
etcdserver.crt
etcdserver.key
scheduler.crt
scheduler.key
KUBE-SCHEDULER
apiserver-kubelet-client.crt
apiserver-kubelet-client.key
controller-manager.crt
controller-manager.key
KUBE-CONTROLLER-MANAGER
kubelet-client.crt
kubelet-client.key
kubelet.crt
kubelet.key
kube-proxy.crt
kube-proxy.key
KUBE-PROXY
KUBELET SERVER

Kubeconfig

```
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
```

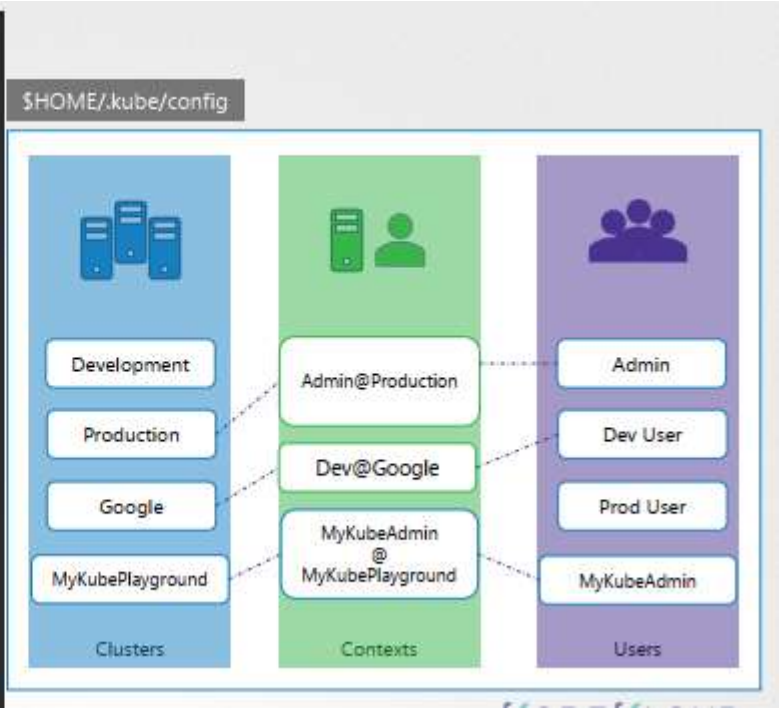
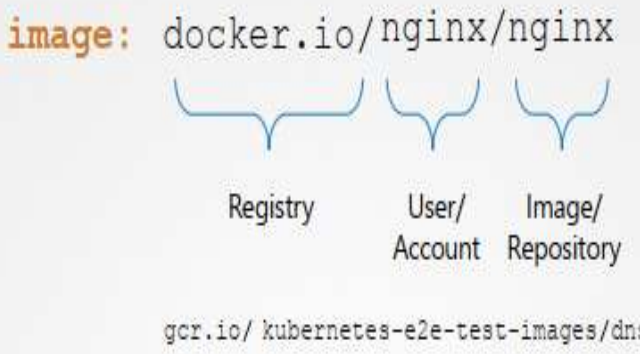


Image Security

nginx-pod.yaml

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



Private repository

```
docker login private-registry.io
```

```
docker run private-registry.io/apps/internal-app
```

```
kubectl create secret docker-registry regcred \
    --docker-server= private-registry.io     \
    --docker-username= registry-user  \
    --docker-password= registry-password  \
    --docker-email= registry-user@org.com
```

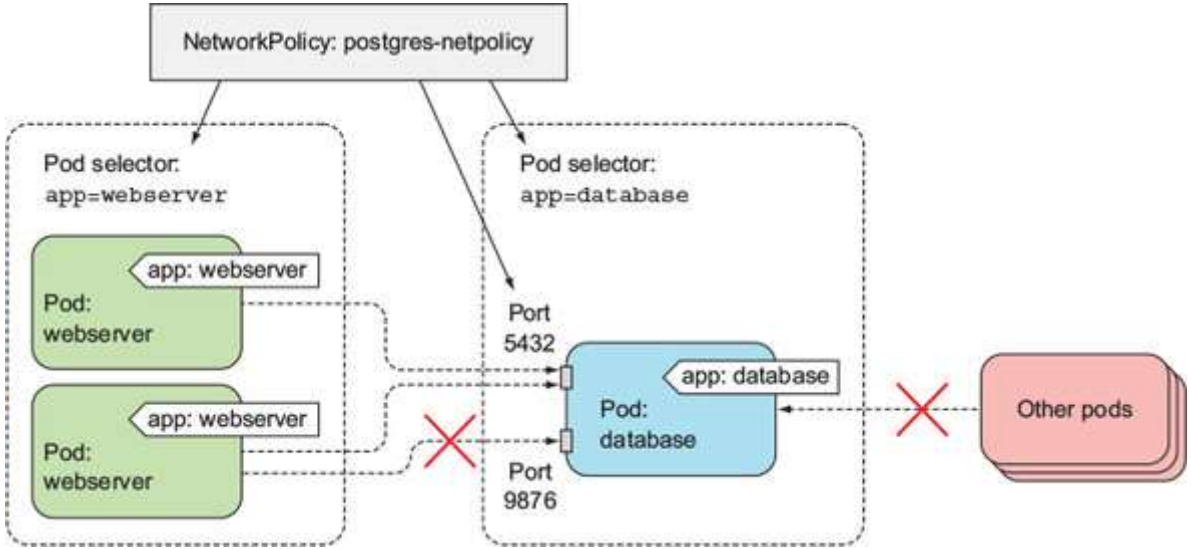
nginx-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: private-registry.io/apps/internal-app
  imagePullSecrets:
  - name: regcred
```

Network Policies

- By default, Kubernetes does not restrict traffic between pods running inside the cluster.
- Securing Kubernetes Cluster Networking.
- A specification of how groups of pods are allowed to communicate with each other and other network endpoints.
- Declaratively configure which pods are allowed to connect to each other.
- NetworkPolicy resources use labels to select pods and define rules which specify what traffic is allowed to the selected pods.

Network Policy Use Case



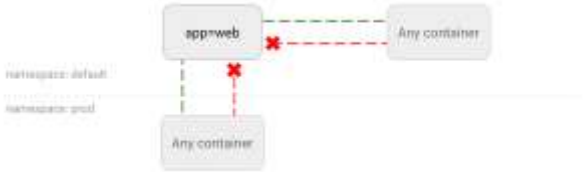

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: postgres-netpolicy
spec:
  podSelector:
    matchLabels:
      app: database
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: webserver
    ports:
    - port: 5432
```

Network Policy Manifest

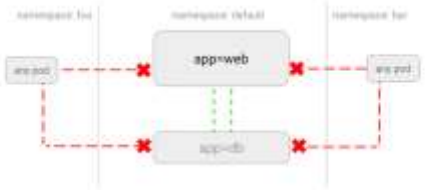
```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: api-allow
spec:
  podSelector:
    matchLabels:
      app: bookstore
      role: api
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: bookstore
  - from:
      - podSelector:
          matchLabels:
            app: inventory
```

Network Policy Use Case

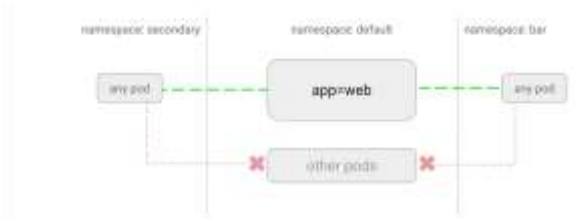
Deny All



Limit Traffic



Deny From Other Namespace



Selective traffic


Q
A