Kubernetes Notes

Presented by Rohit Prakash

INTRODUCTION

Kubernetes is an open source orchestration tool developed by Google for managing microservices or containerized applications across a distributed cluster of nodes. Kubernetes provides highly resilient infrastructure with zero downtime deployment capabilities, automatic rollback, scaling, and self-healing of containers (which consists of auto-placement, auto-restart, auto-replication, and scaling of containers on the basis of CPU usage). Kubernetes created from Borg & Omega projects by google as they use it to orchestrate they data center since 2003. Google open-sourced kubernetes at 2014.

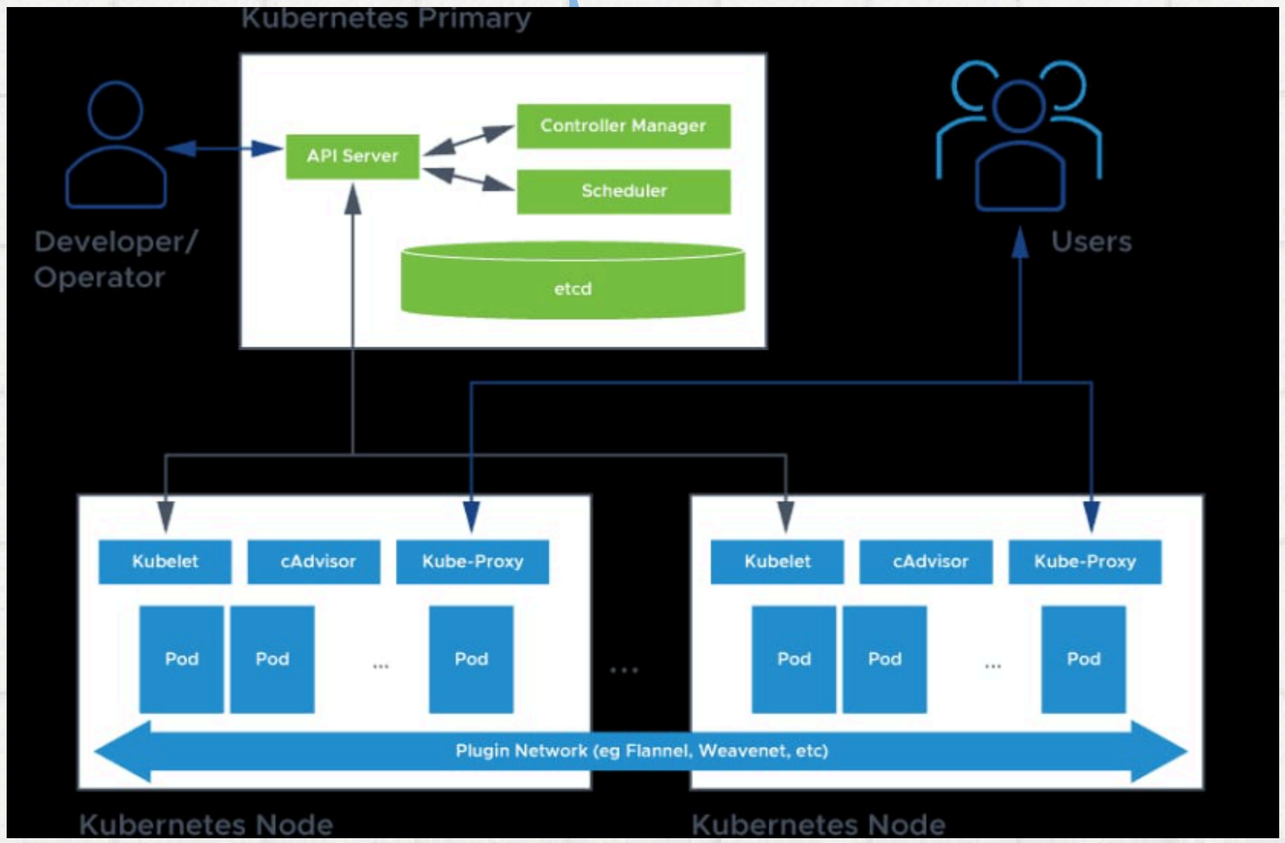
What is Orchestration Do?

- Configuring and scheduling of containers.
- Provisioning and deployments of containers.
- High Availability of containers.
- Configuration of the applications that run in containers.
- Scaling of containers to equally balance the application workloads across infrastructure.
- Allocation of HW resources between containers.
- Load balancing, traffic routing and service discovery of containers.
- Health monitoring of containers.
- Securing the interactions between containers.

Famous container orchestator

Docker Swarm , Mesos (Mesos Sphere),Normand ,Cloud Foundry , Cattel & Cloud (Azure, Amazon, Google, Alibaba, IBM)

Kubernetes Architecture



K8s components

- **K8s Master Node**: the master server that will create the cluster and it has all the components and service that manage, plan, schedule and monitor all the worker nodes.
- **Worker Node:** the server that has host the applications as Pods and containers.

K8s Master Node components

- **API server** – is the primary management components of kubernetes and is responsible for orchestrating all operations (scaling, updates, and so on) in the cluster. It also acts as the gateway to the cluster, so the API server must be accessible by clients from outside the cluster integration with CLI and GUI.
- **Controller-manager** - The Controller Manager is the engine that runs the core control loops, create Pods, watches the state of the cluster, and makes changes to drive status toward the desired state.
- **Replication-Controller** - A ReplicationController ensures that a specified number of pod replicas are running at any one time. It makes sure that a pod is always up and available.
- **Node Controller** - The node controller is a Kubernetes master component which manages various aspects of nodes.
- **Scheduler** - is identify the right node to place a container on based resource limitations or guarantees, taints, tolerations and affinity/anti-affinity roles.
- **etcd cluster** - etcd is a critical part of the Kubernetes. etcd database that stores the state of the cluster, including node and workload information in a key/value format.

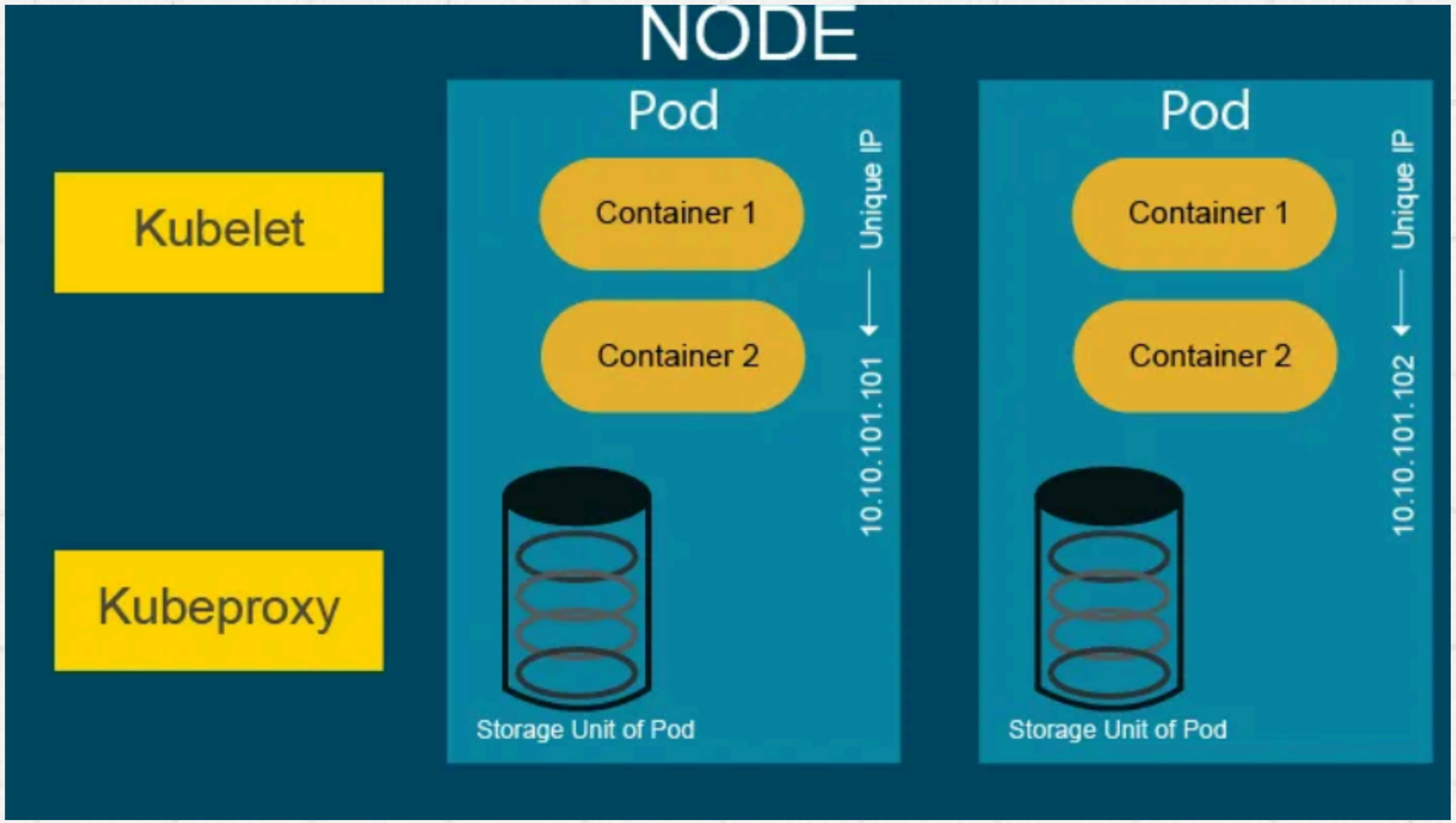
K8s Worker Node components

- **kubelet** - the main service on a node, connect between Master and Node and ensuring that pods and their containers are healthy and running in the desired state. This component also reports to the master on the health of the host where it is running.

- **kube-proxy** - a proxy service that runs on each worker node to deal with individual host subnetting and expose services to the external world. It performs request forwarding to the correct pods/containers across the various isolated networks in a cluster.

- **Kubectl** - kubectl command is a line tool that interacts with kube-apiserver and send commands to the master node. Each command is converted into an API call.

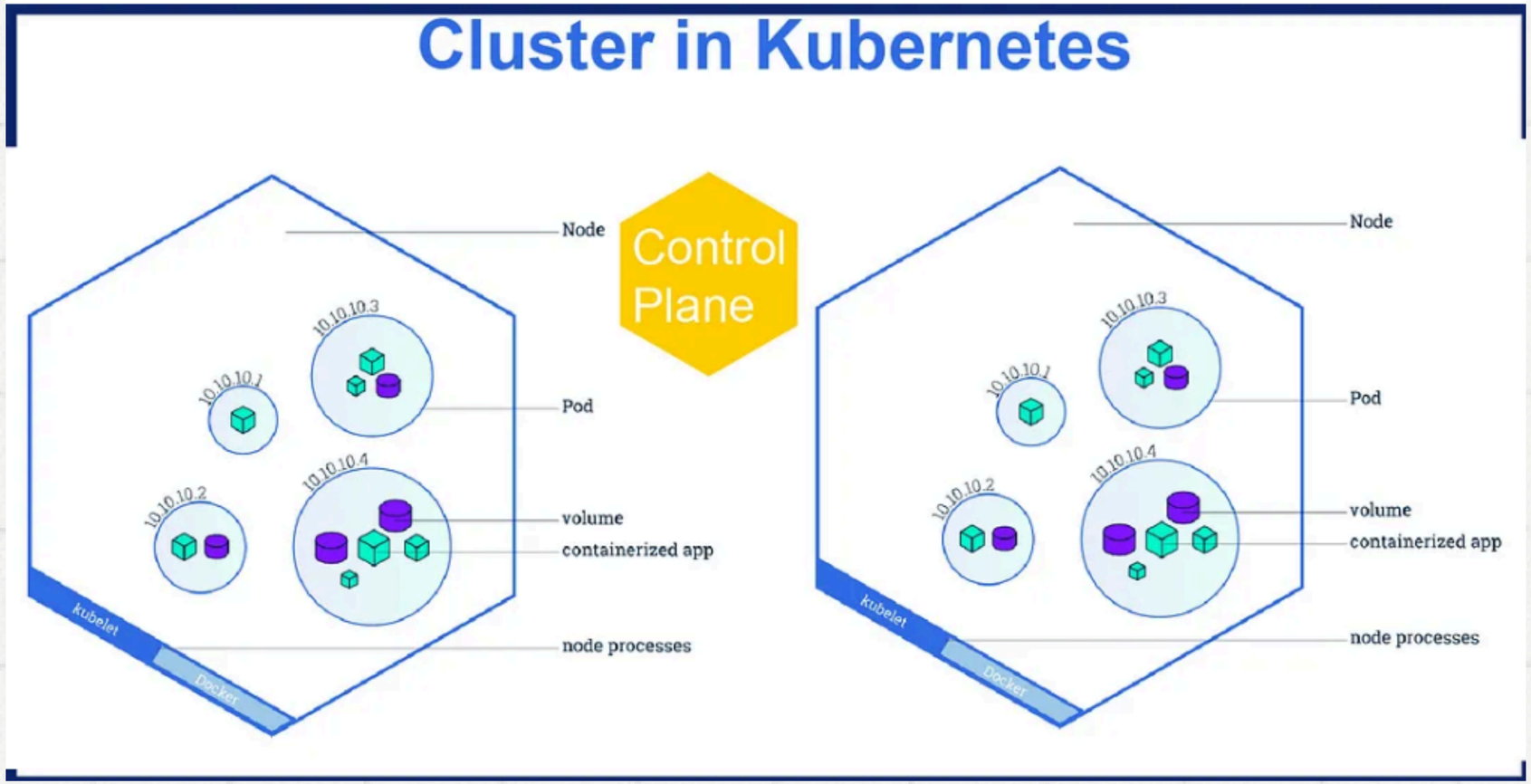
Working of worker node



PODS-A pod is the smallest execution unit in Kubernetes. In Kubernetes, we cannot interact with the containers directly but containers are enclosed in the functional unit called Pods.

NODE-Node is also a box like a Pod, and it encloses Pods inside it. Node is a single host which is capable of running on a physical or virtual machine.

Clusters in Kubernetes:



*A **Kubernetes cluster** is a set of nodes that run containerized applications. Containerizing applications packages an app with its dependences and some necessary services. They are more lightweight and flexible than virtual machines. In this way, Kubernetes clusters allow for applications to be more easily developed, moved and managed.*

Services in K8s

In Kubernetes, a service is an entity that represents a set of pods running an application or functional component. The service holds access policies, and is responsible for enforcing these policies for incoming requests.

Types of services

Cluster IP - It is the default kubernetes service used for internal communication within the cluster.
NodePort - It will open a ports on each nodes and traffic will be forwarded to the service through random port. And I can access the service (Pod) with the node IP + Defined port.
L**oadBalancer** - It is a type that forwards all external traffic to a service through this type. And I can access the service (Pod) with the node name only.
External Name - it is a type used to access a service internally that is hosted outside cluster through DNS CName or A record.

NOTE: For load balncer we need to create Ingress policy as it is not there in K8s. An Ingress may be configured to give Services externally-reachable URLs, load balance traffic, terminate SSL / TLS, and offer name based virtual hosting. . It can be used by Service.Type=NodePort or Service.Type=LoadBalancer.

Service yaml e.g

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

K8s Storage

- Persistent volume claim (PVC) A Persistent Volume Claim (PVC) is a claim request for some storage space by users.

- Persistent volume (PV) A Persistent Volume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes.

In simple words, Persistent Volume is a solution to store data of our containers permanently even after the pods got deleted.

More Info about storage

Persistent Volume supports three types of Reclaim Policy

Retain Delete Recycle

Persistent Volume supports three types of access modes

 ReadWriteOnce ReadOnlyMany ReadWriteMany

L**et's see how to create Persistent Volume.**

- Create a yaml file for persistent volume to get the storage space for our kubernetes cluster.
- Create a yaml file to claim the space using peristent volume claim as per our requirement.
- Define the persistent volume claim in your pod deployment file

Storage yaml eg

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

K8s Security-RBAC

Subjects

The set of users and processes that want to access the Kubernetes API

Resources

The set of Kubernetes API Objects available in the cluster. Examples include Pods, Deployments, Services, Nodes, and PersistentVolumes, among others.

Verbs

The set of operations that can be executed to the resources above. Different verbs are available (examples: get, watch, create, delete, etc.), but ultimately all of them are Create, Read, Update or Delete (CRUD) operations.

```yaml
kind: Role
apiVersion:
rbac.authorization.k8s.io/v1beta1
metadata:
   name: pod-read-create
   namespace: test
rules:
 - apiGroups: [""]
   resources: ["pods"]
   verbs: ["get", "list", "create"]
```

```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: salme-pods
  namespace: test
subjects:
- kind: User
  name: jsalmeron
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: ns-admin
  apiGroup: rbac.authorization.k8s.io
```

Role creation yaml eg

K8s Monitoring &logging

Can be done with Promethus

It is a monitoring and alerting system. It was built at SoundCloud and was open sourced in 2012. It handles the multi-dimensional data very well.

Sematext Docker Agent

It is a modern Docker-aware metrics, events, and log collection agent. It runs as a tiny container on every Docker host and collects logs, metrics, and events for all cluster node and containers. It discovers all containers (one pod might contain multiple containers) including containers for Kubernetes core services, if the core services are deployed in Docker containers. After its deployment, all logs and metrics are immediately available out of the box.

Kubernetes Log

Kubernetes containers' logs are not much different from Docker container logs. However, Kubernetes users need to view logs for the deployed pods. Hence, it is very useful to have Kubernetes-specific information available for log search, such as -

Kubernetes namespace
Kubernetes pod name
Kubernetes container name
Docker image name
Kubernetes UID

Deployment types mostly used

Exploring k8s

Rolling Deployment

A Rolling Deployment strategy allows for a gradual update of the application by incrementally replacing old instances with new ones. This approach ensures minimal downtime as the new instances are gradually introduced while the old ones are gracefully terminated. Rolling deployments are ideal for applications that require high availability and continuous operations.

CANARY DEPLOYMENT

Canary Deployment is a strategy where a small subset of users or traffic is directed to the new version of the application, while the majority of traffic continues to use the previous version. This approach enables the monitoring of the new version's performance and detects any potential issues before fully rolling out the update. Canary deployments are particularly useful for large-scale applications that cater to a diverse user base.

Blue/Green Deployment

A Blue/Green Deployment strategy involves maintaining two identical environments, one referred to as "blue" (the production environment) and the other as "green" (the new version of the application). The new version is deployed in the green environment, thoroughly tested, and then traffic is switched from the blue to the green environment. This approach allows for instant rollbacks if any issues arise, ensuring minimal impact on users.

.

A/B Deployment

A/B Testing involves deploying multiple versions of an application and directing different subsets of users or traffic to each version.

K8s Autoscaling

Kubernetes autoscaling is a feature that allows a cluster to automatically increase or decrease the number of nodes, or adjust pod resources, in response to demand

Three types of this are:

1. Kubernetes ReplicationController
2. Kubernetes ReplicaSets
3. Deployments

01. This is an older replication mechanism, which was replaced by ReplicaSets, but still works and is widely used. It creates a certain number of identical pods, and if a pod fails, replaces it. It also lets you update several pods or delete pods with one command.

02. ReplicaSets are declared in a similar way to ReplicationControllers, but provide more selector options. According to the Kubernetes documentation, ReplicaSets are usually not used directly—they are typically created as part of a deployment.

03. Deployments are one level up from ReplicaSets. They allow you to use a declarative method to deploy ReplicaSets and pods. You use YAML configuration to define what your group of pods should look like, and the deployment manipulates Kubernetes objects to create pods exactly according to the YAML specification.
Unlike a ReplicaSet, in a deployment, you can modify the YAML Pod template, and the deployment will take care of rolling out the changes. You can roll back to a previous version of your configuration if necessary.

#Kubernetes Commands

PODS

$ kubectl get pods
$ kubectl get pods --all-namespaces
$ kubectl get pod monkey -o wide
$ kubectl get pod monkey -o yaml
$ kubectl describe pod monkey

Create Deployments

Create single deployment: $ kubectl run nginx --image=nginx –record

Scaling PODs

$ kubectl scale deployment/POD_NAME --replicas=N

POD Upgrade and history

List history of deployments

$ kubectl rollout history deployment/DEPLOYMENT_NAME

Jump to specific revision

$ kubectl rollout undo deployment/DEPLOYMENT_NAME --to-revision=N

#Kubernetes Commands

SERVICES

List services

$ kubectl get services

Expose PODs as services (creates endpoints)

$ kubectl expose deployment nginx --port=80 --type=NodePort

VOLUMES

Lists PV and PVC

$ kubectl get pv

$ kubectl get pvc

Secrets

$ kubectl get secrets

$ kubectl create secret generic --help

$ kubectl create secret generic mysql --from-literal=password=root

$ kubectl get secrets mysql -o yaml

#Kubernetes Commands

ConfigMaps

$ kubectl create configmap foobar --from-file=config.js
$ kubectl get configmap foobar -o yaml DNS List DNS-PODs:
$ kubectl get pods --all-namespaces |grep dns

C**heck DNS for pod nginx (assuming a busybox POD/container is running)**

$ kubectl exec -ti busybox -- nslookup nginx
Note: kube-proxy running in the worker nodes manage services and set iptables rules to direct traffic
.

Ingress(Commands to manage Ingress for ClusterIP service type:)

$ kubectl get ingress
$ kubectl expose deployment ghost --port=2368
Spec for ingress: backend

Horizontal Pod Autoscaler

When heapster runs:
$ kubectl get hpa
$ kubectl autoscale --help

#Kubernetes Commands

DaemonSets
$ kubectl get daemonsets
$ kubectl get ds Scheduler

NodeSelector based policy:
$ kubectl label node minikube foo=bar Node Binding through API Server:
$ kubectl proxy $ curl -H "Content-Type: application/json" -X POST --data @binding.json http://localhost:8001/api/v1/namespaces/default/pods/foobar-sched/binding

Tains and Tolerations
$ kubectl taint node master foo=bar:NoSchedule

Troubleshooting
$ kubectl describe
$ kubectl logs
$ kubectl exec
$ kubectl get nodes --show-labels
$ kubectl get events

Docs Cluster: https://kubernetes.io/docs/tasks/debug-application-cluster/debug-cluster/ https://github.com/kubernetes/kubernetes/wiki/Debugging-FAQ

#Kubernetes Commands

Role Based Access Control

$ kubectl create role fluent-reader --verb=get --verb=list --verb=watch --resource=pods
$ kubectl create rolebinding foo --role=fluent-reader --user=minikube
$ kubectl get rolebinding foo -o yaml

Security Contexts

Docs: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Pod Security Policies

Docs: https://github.com/kubernetes/kubernetes/blob/master/examples/podsecuritypolicy/rbac/READ ME.md

Network Policies

Network isolation at Pod level by using annotations
$ kubectl annotate ns "net.beta.kubernetes.io/network-policy={\"ingress\": {\"isolation\": \"DefaultDeny\"}}"

More about Network Policies as a resource: https://kubernetes.io/docs/tasks/administer-cluster/declare-network-policy/

K8s YAML

apiVersion

Which version of the Kubernetes API you're using to create this object

kind

What kind of object you want to create

metadata

Data that helps uniquely identify the object, including a name string, UID, and optional namespace

spec

What state you desire for the object

Thank you very much!

https://www.linkedin.com/in/rohit-prakash-204391252/