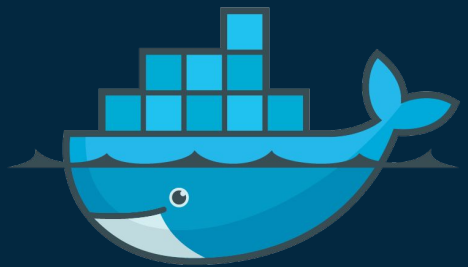
K21Academy
Learn Cloud from Experts

Docker & Certified Kubernetes Administrator (CKA)

Concepts & Architecture | Storage | Networking | HA & Clusters | Scheduling
Administration | Docker Compose | Security | Troubleshooting

16 Module | 60+ Lessons | 39 Hands-On Labs | Exam Preparation | On-Job Support

Atul Kumar
Oracle ACE & Cloud Expert



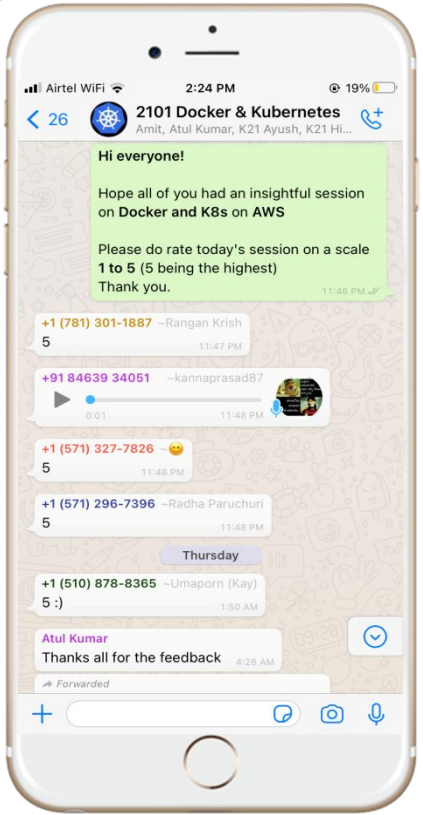
docker + kubernetes

WhatsApp & Ticketing System

K21Academy
Learn Cloud from Experts

support@k21academy.com

zendesk



Agenda: Module

- Introduction to Kubernetes
  - Kubernetes Architecture
  - Kubernetes Object
  - Kubernetes Pod
  - Kubernetes Networking
- Kubernetes Deep Dive
  - Highly Available and Scalable Application
  - Autoscaling with HPA
  - Persistent Storage
  - Advanced Scheduling
  - Advance Scheduling - Taint & Tolerations

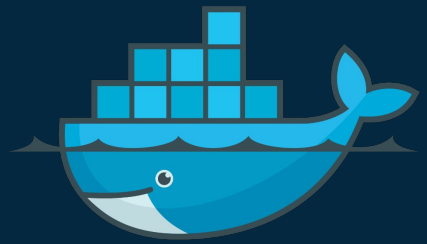
Agenda: Module

K21Academy
Learn Cloud from Experts

- Kubernetes Security
  - RBAC - Role Based Access Control
  - Role and Role Binding
  - Network Security
  - Security Context
  - ConfigMap
- Cluster Resource Limit
- MultiContainer Pattern - Sidecar
- Maintenance & Troubleshooting
- Working with Application Stack

Agenda: Module

- Deploy Stateful Application
- Deploy Daemon Process
- Working with Logging Stack
- Kubernetes Next Level
  - Managed Kubernetes on Cloud
  - Advance Scheduling - Taint & Tolerations
  - Advanced Networking
  - Helm and Helm Chart
- Dynamic Storage Provisioning



docker

+

kubernetes

K21Academy
Learn Cloud from Experts

Introduction to Kubernetes

What Is Kubernetes?

- An open-source container-orchestration system
- Automates application deployment, scaling, and management
- Originally designed by Google
- Maintained by CNCF

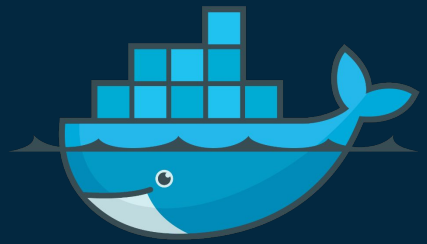
Why to use K8s??

- Can’t live with single server deployment
- Inter-host communication of Containers
- Deploying and updating software at scale
- Auto-healing
- Logging and Monitoring
- Better management through modularity

What It Provides?

- A set of building blocks to
  - Mechanism to deploy, maintain and scale applications
  - To edit and update a running application
  - Scheduling application as per need
  - Monitoring and Logging
  - RBAC

Questions



docker

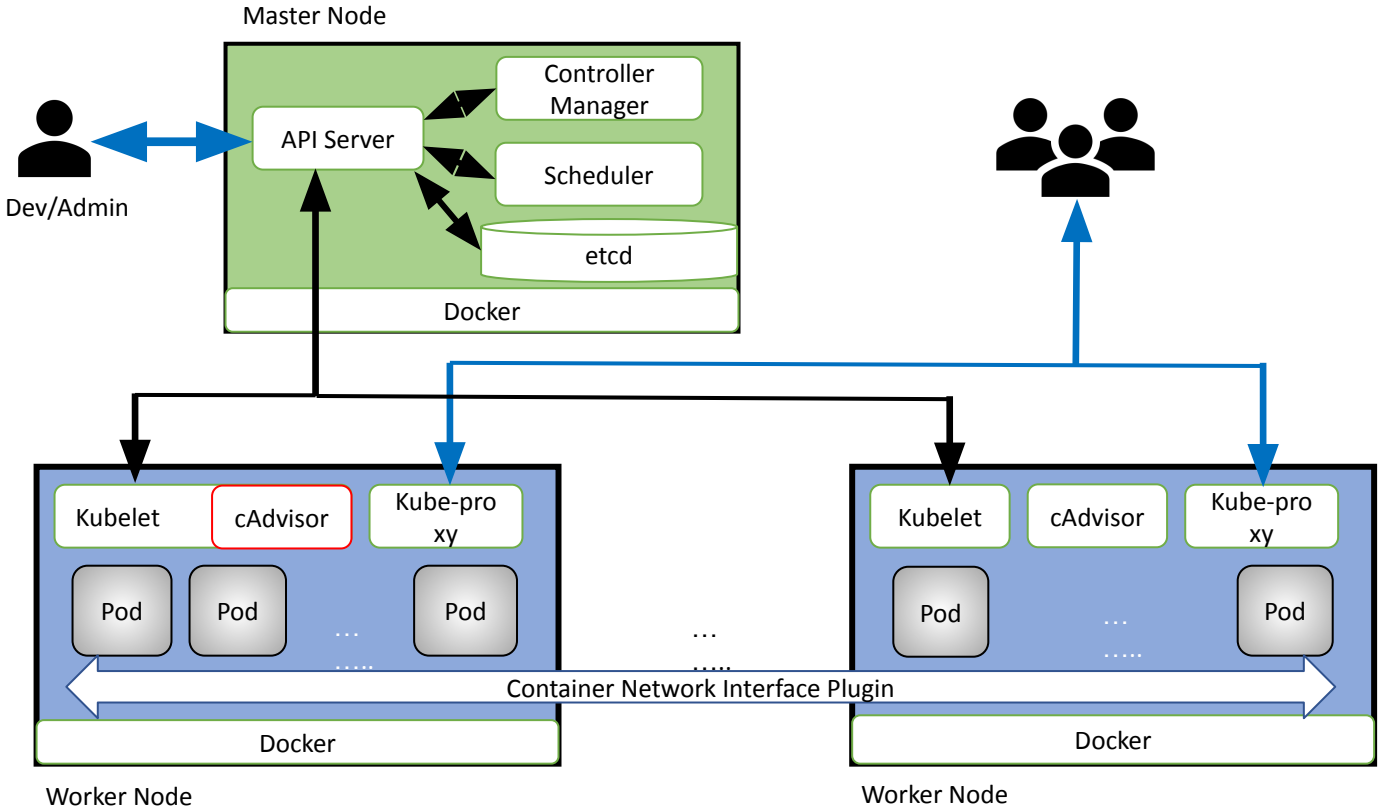
+

kubernetes

K21Academy
Learn Cloud from Experts

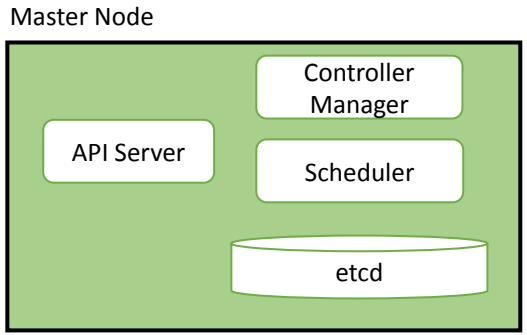
Kubernetes Architecture

Kubernetes Architecture



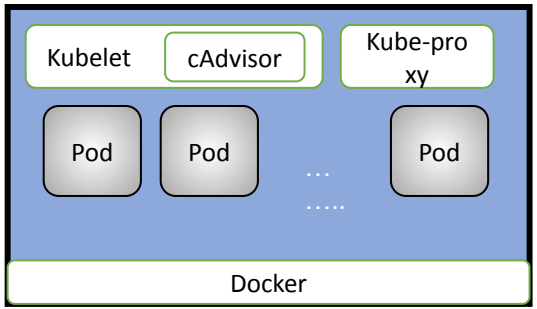
Master Node Architecture

- **API Server**: Configures and validates data for api objects like pods, services. Its a front-end of control plane
- **Scheduler**: It decides where in the cluster the workloads are to be run
- **etcd**: Stores all cluster-related data
- **Controller**: Daemon that embeds core control loops that regulates system state via routine tasks



Worker Node Architecture

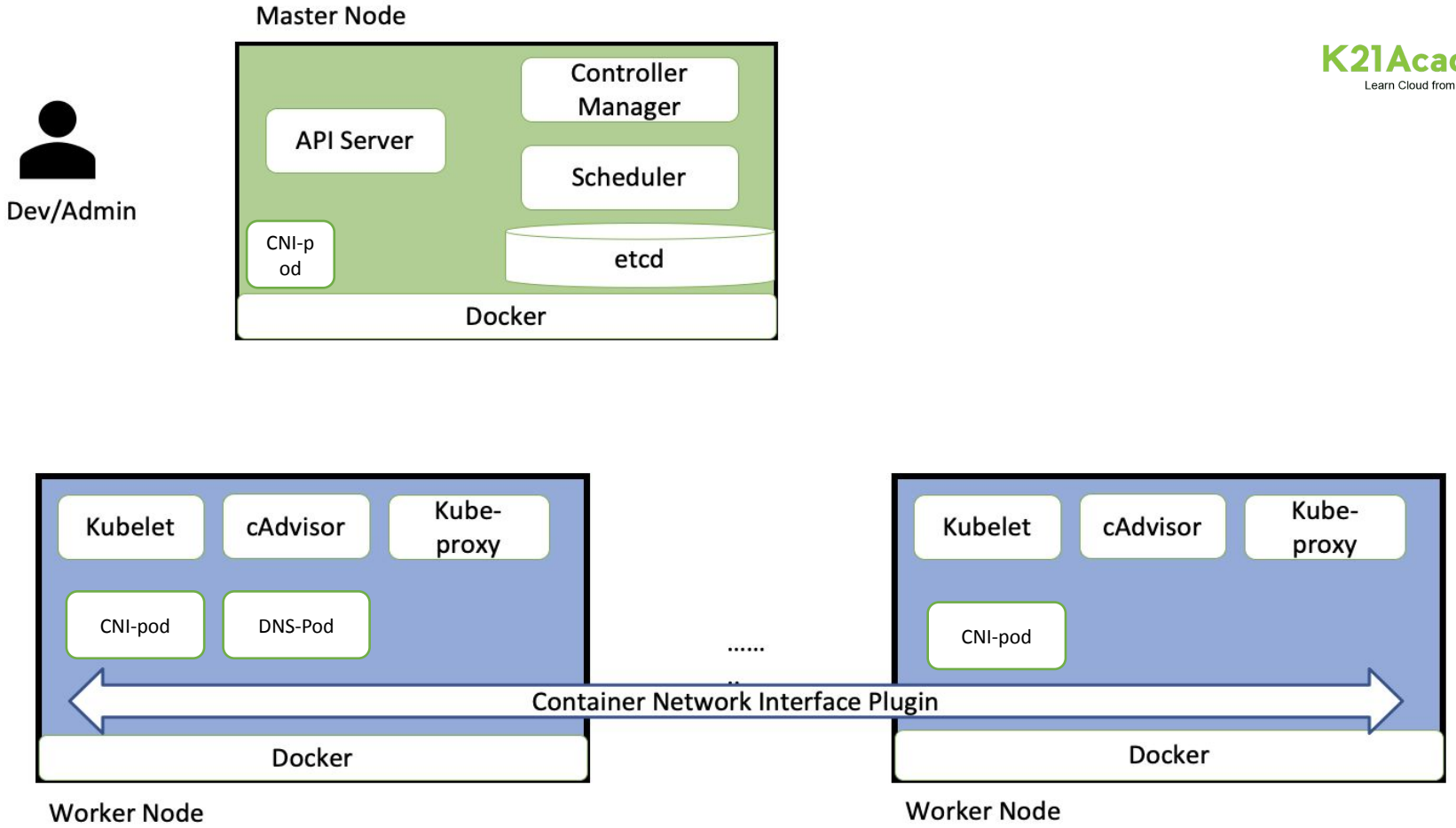
- **kubelet:** Primary node agent which performs various tasks like mounting volumes, running containers, etc. for pods assigned to the node
- **kube-proxy:** Provides service abstraction and connection forwarding
- **Docker:** Container engines for running containers
- **cAdvisor:** Provides container users an understanding of the resource usage and performance characteristics of their running containers



Worker Node

Kubernetes Basics

- Everything is an API call served by the *Kubernetes API server* (kube-apiserver)
- The API server is a gateway to an etcd datastore that maintains the desired state of your application cluster
- To update the state of a Kubernetes cluster, we make API calls to the API server describing our desired state
- Once we've declared the desired state of your cluster using the API server, controllers ensure that the cluster's current state matches the desired state by continuously watching the state of the API server and reacting to any changes
- Controllers operate using a simple loop that continuously checks the current state of the cluster against the desired state of the cluster
- If there are any differences, controllers perform tasks to make the current state match the desired state.



Questions

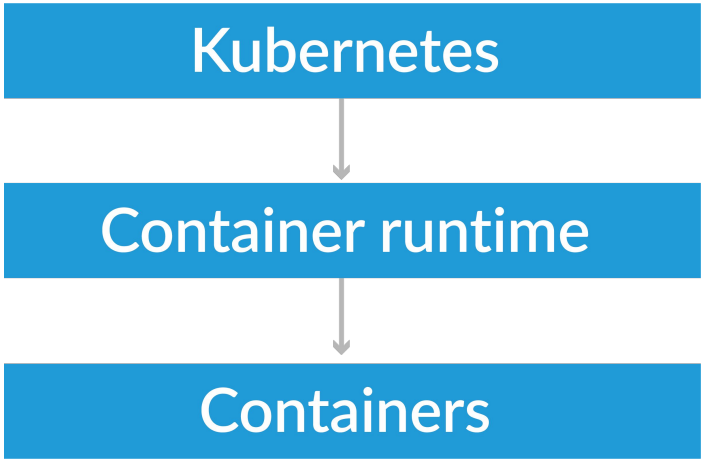
K21Academy
Learn Cloud from Experts

Kubernetes Deprecating Docker

Kubernetes is deprecating Docker

K21Academy
Learn Cloud from Experts

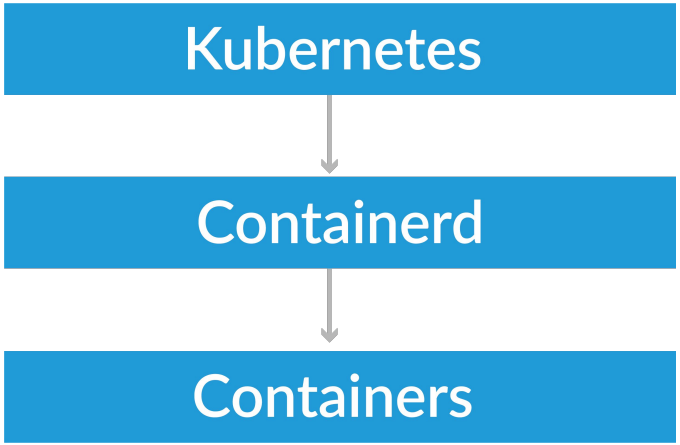
- Kubernetes is removing support for Docker as a container runtime.
- The container runtime runs containers on a host, and Kubernetes tells the container runtime on each host what to do.
- You will still be able to use Docker in other tasks that are relevant to Kubernetes, but you will not be able to use Docker as the container runtime underneath Kubernetes.



When Kubernetes is deprecating Docker

K21Academy
Learn Cloud from Experts

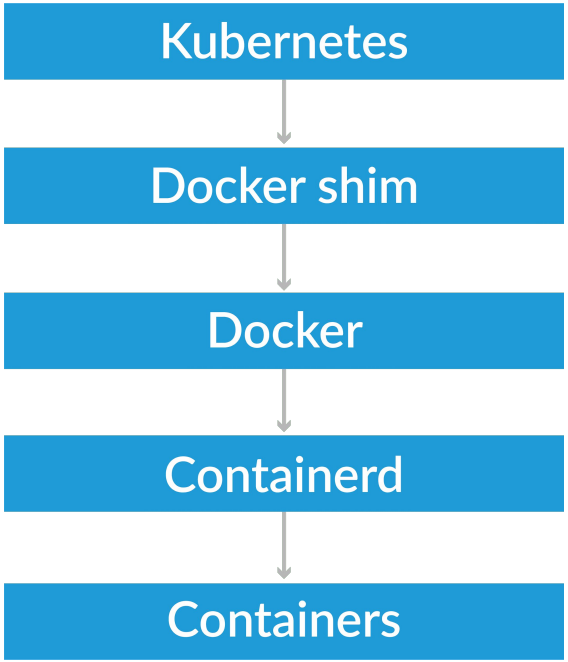
- Kubernetes works with all container runtimes that implement a standard known as the Container Runtime Interface (CRI). This is essentially a standard way of communicating between Kubernetes and the container runtime.
- Any runtime that supports CRI automatically works with Kubernetes.
- If you're creating new clusters. At v1.20, you will get a deprecation warning for Docker.
- When Docker runtime support is removed in a future release (currently planned for the 1.22 release in late 2021) of Kubernetes it will no longer be supported.
- you will need to switch to one of the other compliant container runtimes, like containerd or CRI-O.



Why Kubernetes is deprecating Docker

K21Academy
Learn Cloud from Experts

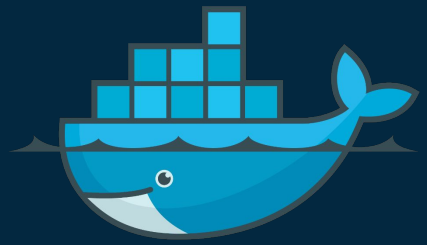
- Docker does not implement the Container Runtime Interface (CRI).
- Kubernetes implemented the Docker shim, an additional layer to serve as an interface between Kubernetes and Docker and Kubernetes manages (update and all binary) of Dockershim.
- Now, there are plenty of runtimes available that implement the CRI, so Kubernetes is removing Docker which takes special support (Dockershim) and no need to maintain the dockershim binary any more by kubernetes.



Role of Docker in Kubernetes

K21Academy
Learn Cloud from Experts

- Docker is still going strong as a tool for developing and building container images, as well as running them locally.
- Kubernetes can still run containers built using Docker's **Open Container Initiative (OCI)** image format, so you can still use Dockerfiles and build your container images using Docker.
- Kubernetes will also continue to be able to pull from Docker registries (such as Docker hub).



docker

+

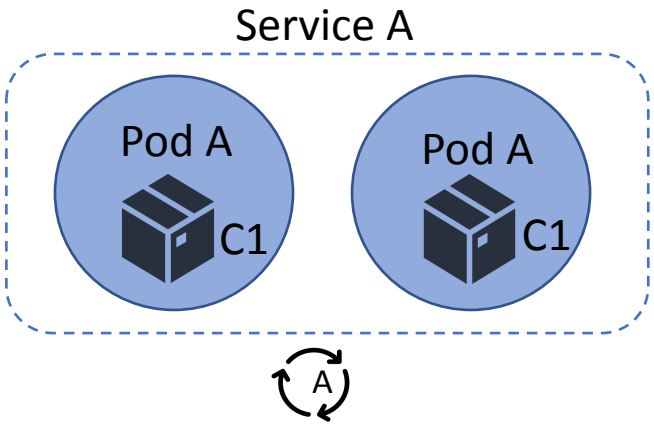
kubernetes

K21Academy
Learn Cloud from Experts

Kubernetes Building Blocks

Kubernetes Object

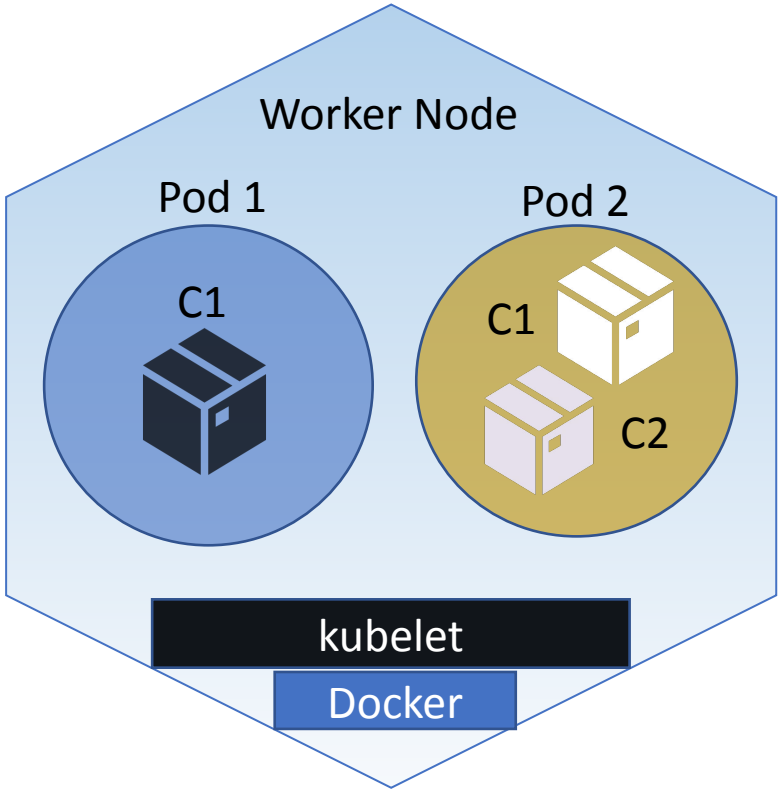
- Anything that persists in the system
- It defines desired state of the system
- Pod
- Deployment
- Service



Namespace

- A way to divide cluster resources
- Names of resources need to be unique within a namespace
- Kubernetes starts with three initial namespaces:
  - default - The default namespace for objects with no other namespace
  - kube-system - The namespace for objects created by the Kubernetes system
- Each Kubernetes resource can only be in one namespace

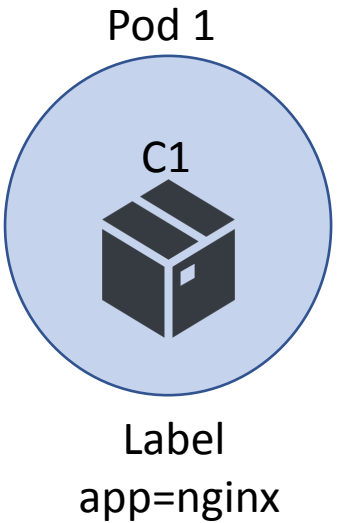
Kubernetes Pod



- Pod is the atomic unit on the K8s platform
- Pod can be a group of one or more application containers
- Pod has a unique Ip Address
- Containers in a pod share the IP address and port namespace

Label and Selector

- Labels are key/value pairs that are attached to objects
- Labels are intended to be used to specify identifying attributes of objects
- Labels can be used to organize and to select subsets of objects
- Selectors match the labels to connect to K8s objects



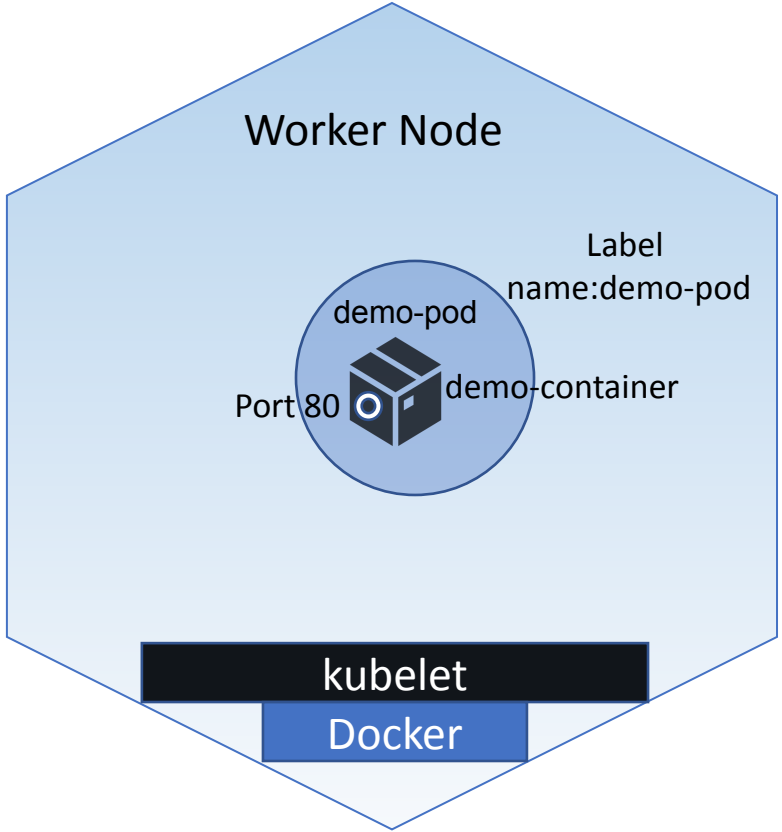
Annotations

- Kubernetes labels and annotations are both ways of adding metadata to Kubernetes objects
- To attach arbitrary non-identifying metadata to objects
- Labels allow you to identify, select and operate on Kubernetes objects
- Annotations are not used to identify and select objects
- The metadata in an annotation can be small or large, structured or unstructured, and can include characters not permitted by labels
- Eg: Build, release, or image information like timestamps, release IDs, git branch, PR numbers, image hashes, and registry address

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: annotations-demo
  annotations:
    imageregistry: "https://hub.docker.com/"
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

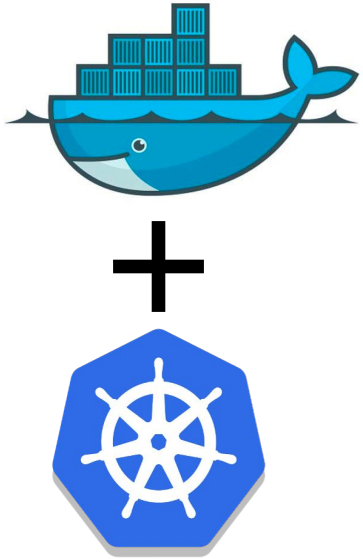
Pod Definition

```
apiVersion: v1
kind: Pod
metadata:
  name: demo-pod
  labels:
    name: demo-pod
spec:
  containers:
  - name: demo-container
    image: nginx
    ports:
    - containerPort: 80
~
~
~
~
```

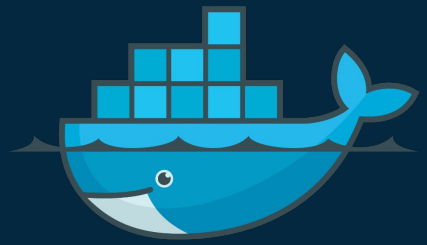


Pod Limitations

- IP Address is not persistent - Service
- Memory is not persistent - Persistent Volume
- Can restart or get deleted - Controller
- Need to have a management layer to manage



Questions



docker

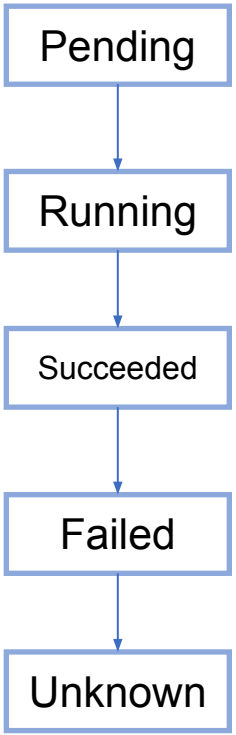
+

kubernetes

K21Academy
Learn Cloud from Experts

Pod Lifecycle

Pod Lifecycle



The Pod has been accepted by the Kubernetes cluster. The API Server has created a pod resource and stored it in etcd, but the pod has not been scheduled yet, nor have container images been pulled from the registry.

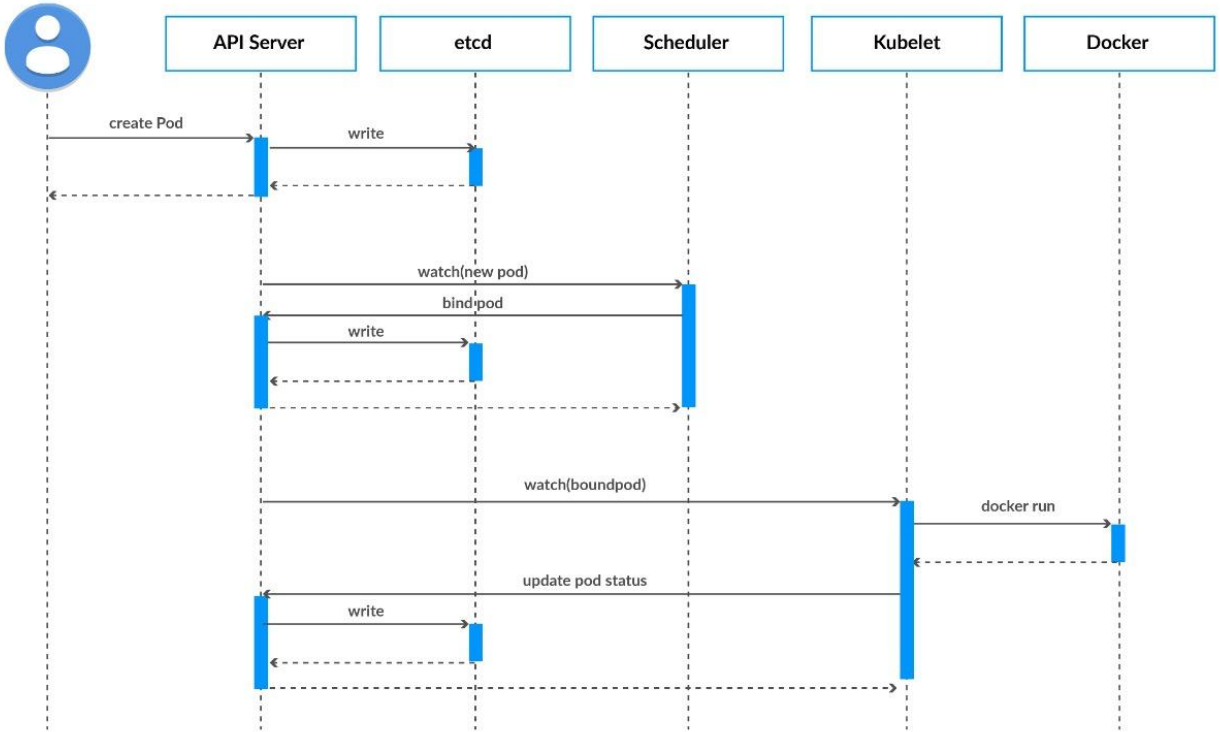
The pod has been scheduled to a node and all containers have been created by the kubelet.

All containers in the Pod have terminated in success, and will not be restarted.

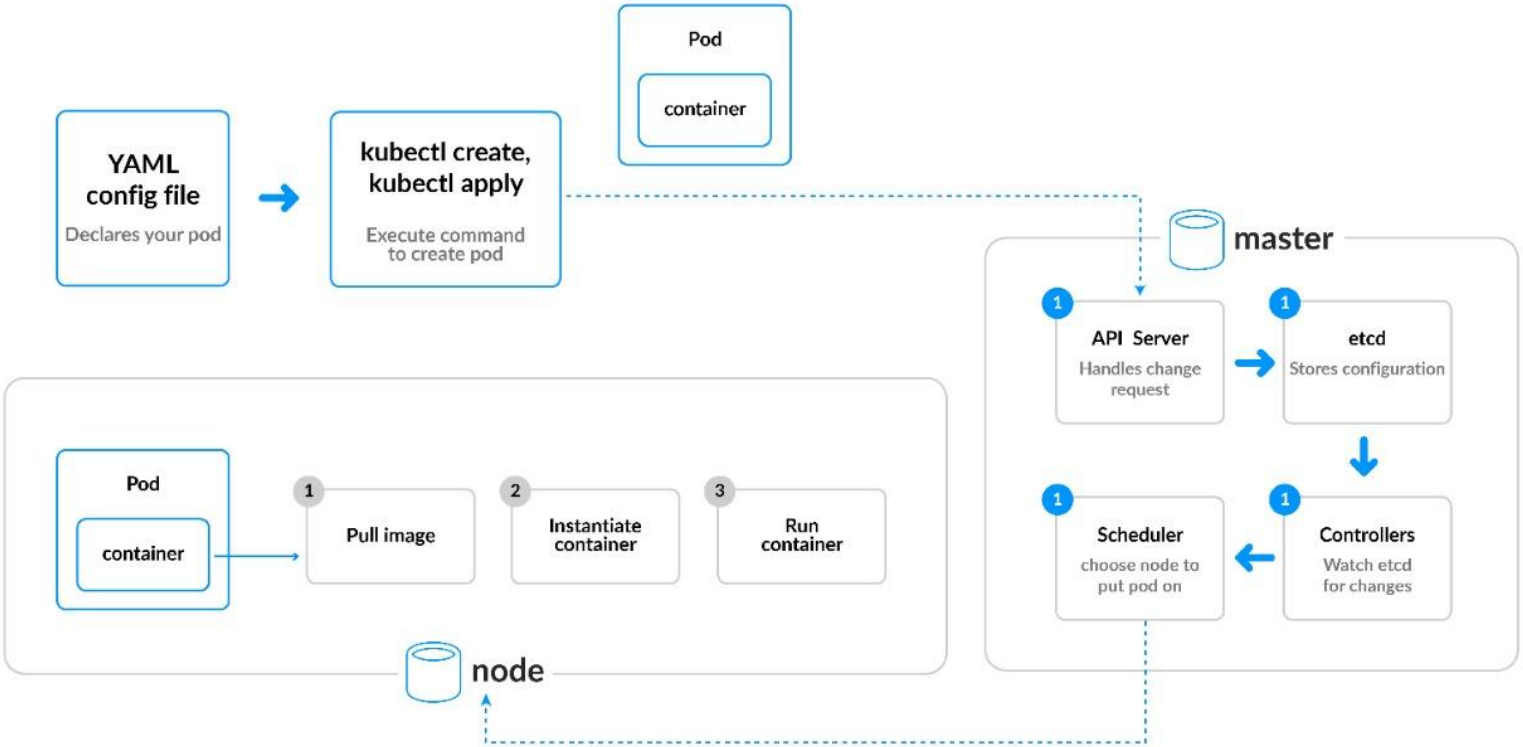
All containers in the Pod have terminated, and at least one container has terminated in failure.

For some reason the state of the Pod could not be obtained. This phase typically occurs due to an error in communicating with the node where the Pod should be running.

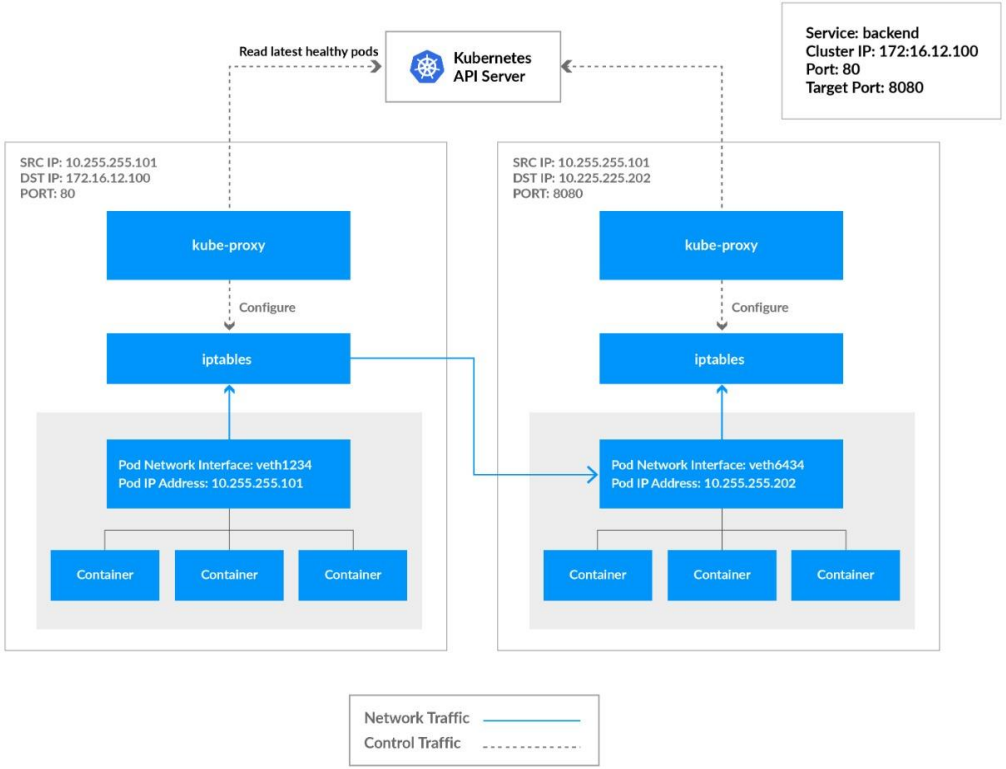
Pod Lifecycle

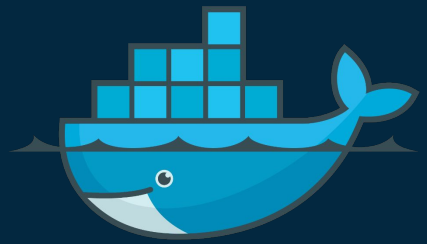


Kubernetes Pod Creation



Kubernetes Pod Networking





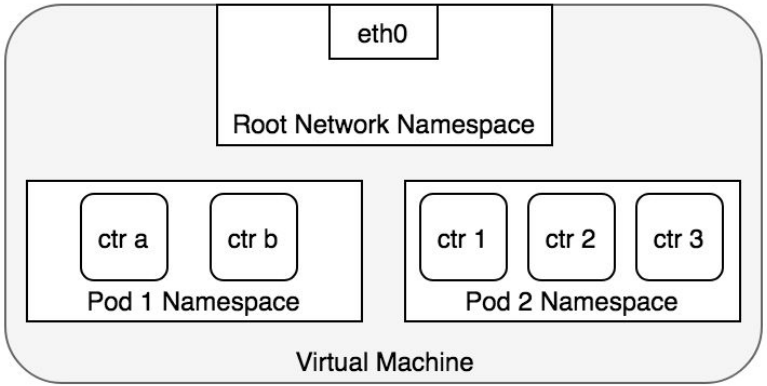
+

K21Academy
Learn Cloud from Experts

Kubernetes Networking

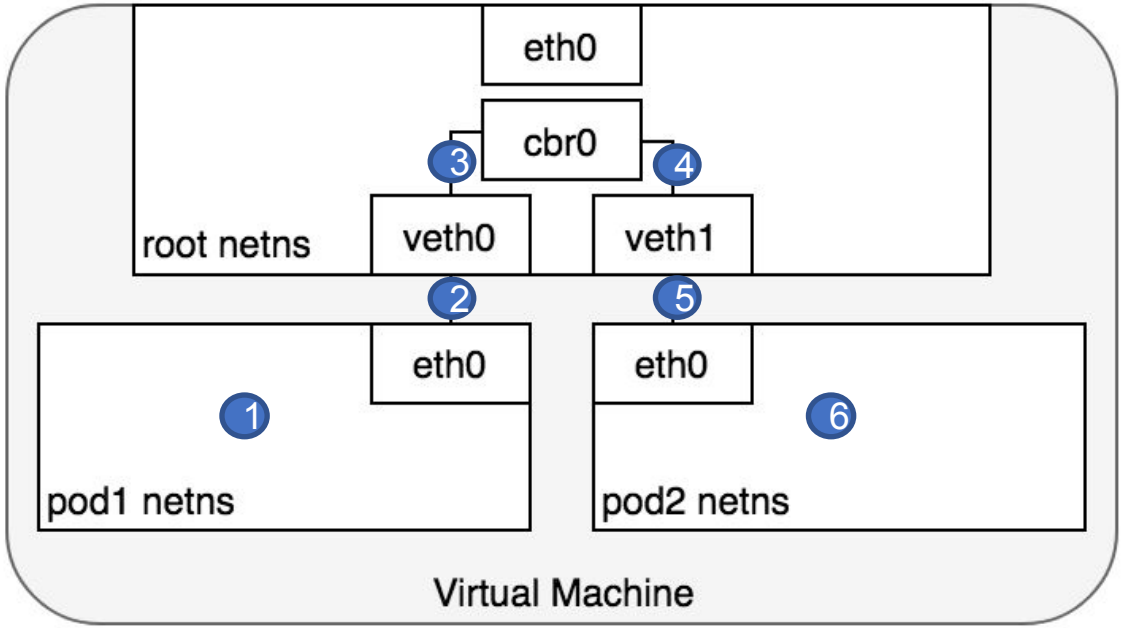
Kubernetes Networking Model

- All Pods can communicate with all other Pods without using network address translation (NAT)
- All Nodes can communicate with all Pods without NAT
- Containers within a Pod all have the same IP address and port space assigned through the network namespace assigned to the Pod
- They can find each other via localhost since they



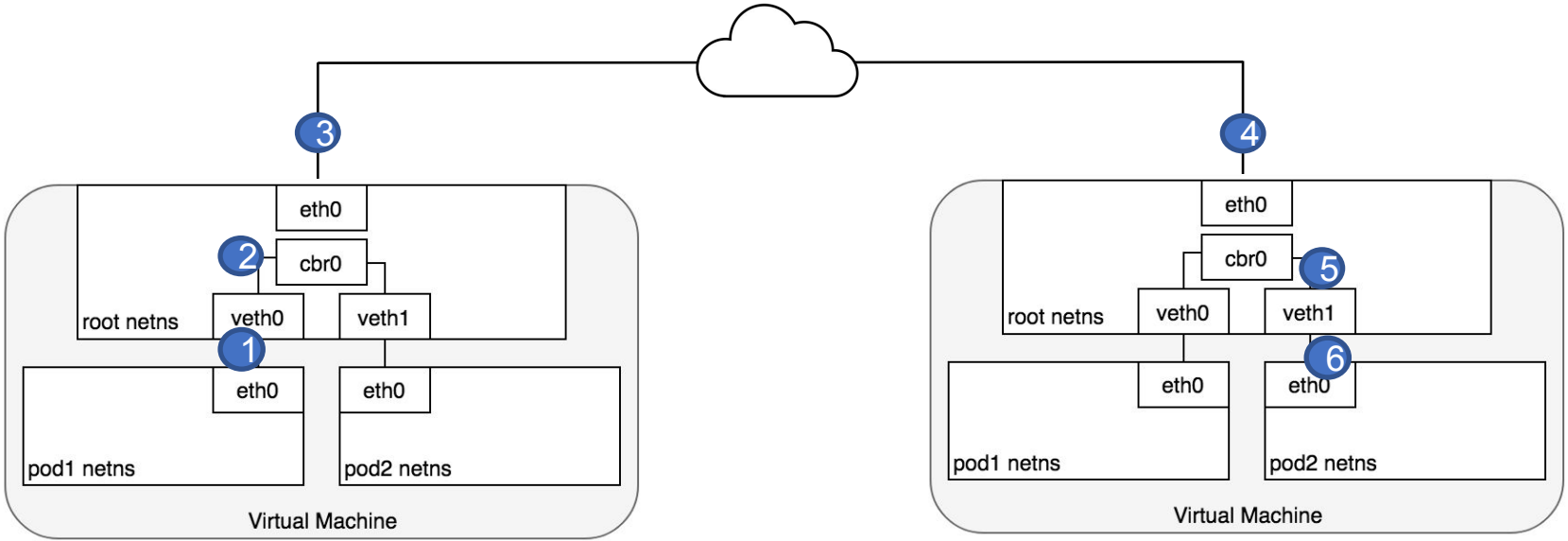
Pod-to-Pod Networking – Same Node

- Every Pod has a real IP address and each Pod communicates with other Pods using that IP address



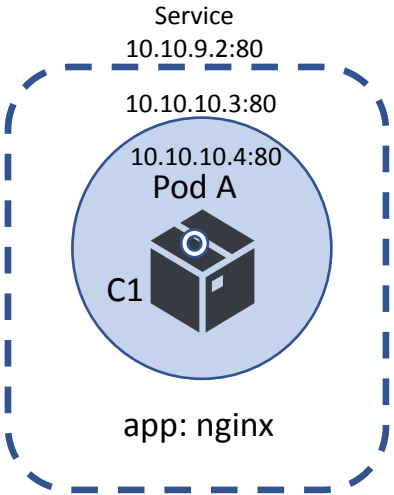
Pod-to-Pod Networking – Across Nodes

- Every Pod has a real IP address and each Pod communicates with other Pods using that IP address



Kubernetes Service

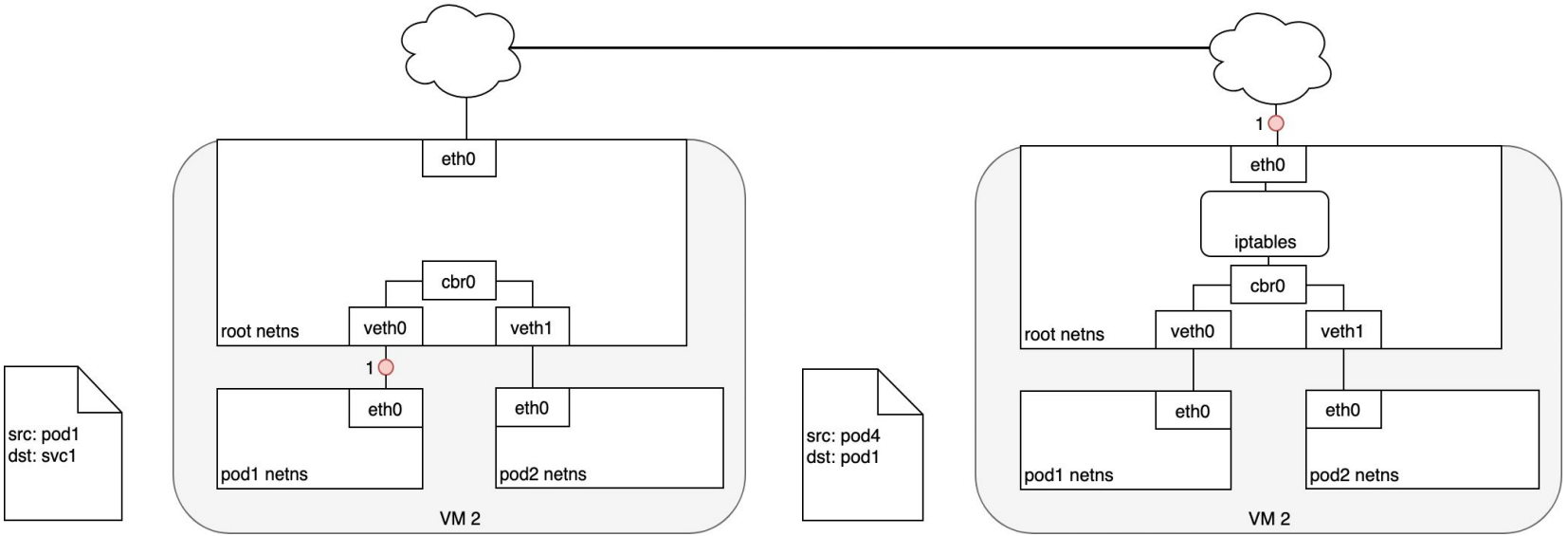
- Service in Kubernetes aims to solve the connectivity issue
- Service is an abstraction which defines a logical set of Pods along with policies with which to access them
- Service has a static IP address
- With Label/Selector, Pod and Service match
- Service exposes the application Pod to other applications and if needed to external world
- Does load balance and health check



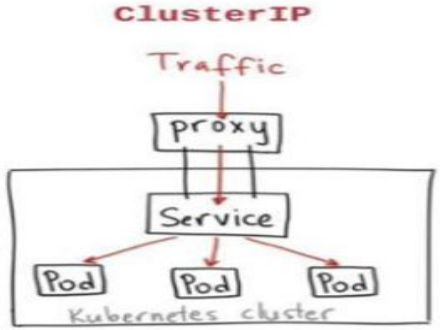
Kubernetes Service

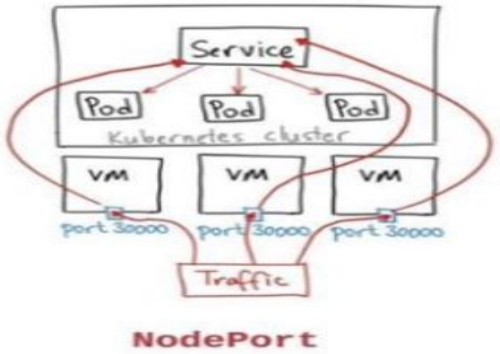
- A Kubernetes Service manages the state of a set of Pods, allowing us to track a set of Pod IP addresses that are dynamically changing over time
- Services act as an abstraction over Pods and assign a single virtual IP address to a group of Pod IP addresses
- Any traffic addressed to the virtual IP of the Service will be routed to the set of Pods that are associated with the virtual IP
- This allows the set of Pods associated with a Service to change at any time — clients only need to know the Service's virtual IP, which does not change

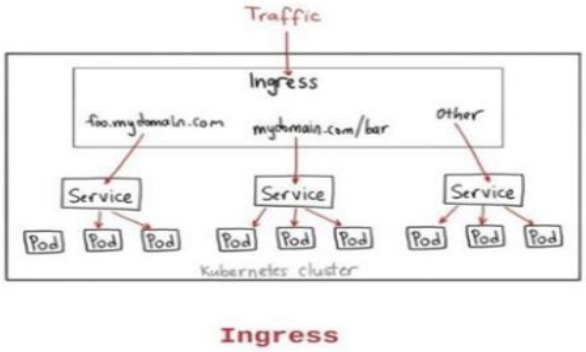
Pod-to-Svc-to-Pod Networking

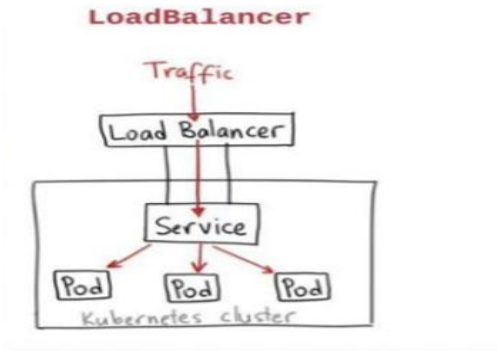


K8s Service Types



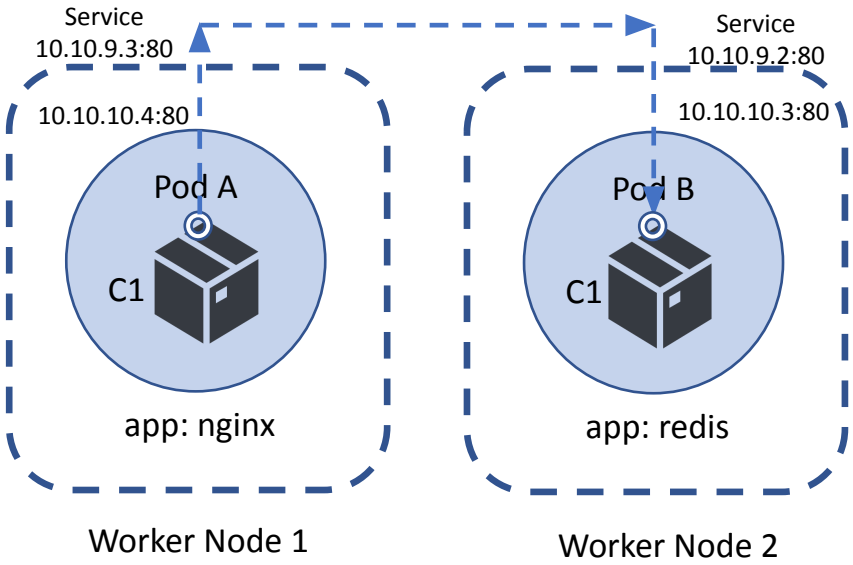






Kubernetes Service - ClusterIP

- Exposes the Pod to only within the cluster
- Assigns a Private IP Address in Cluster IP range
- Can talk to applications only within the cluster
- Doesn't have external connectivity



Kubernetes Service - ClusterIP

K21Academy
Learn Cloud from Experts

```
$ kubectl expose pod demo-pod --port 80 --target-port 80 --type ClusterIP
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
demo-pod     ClusterIP   10.108.93.193   <none>        80/TCP    6s
```

Kubernetes Service - ClusterIP

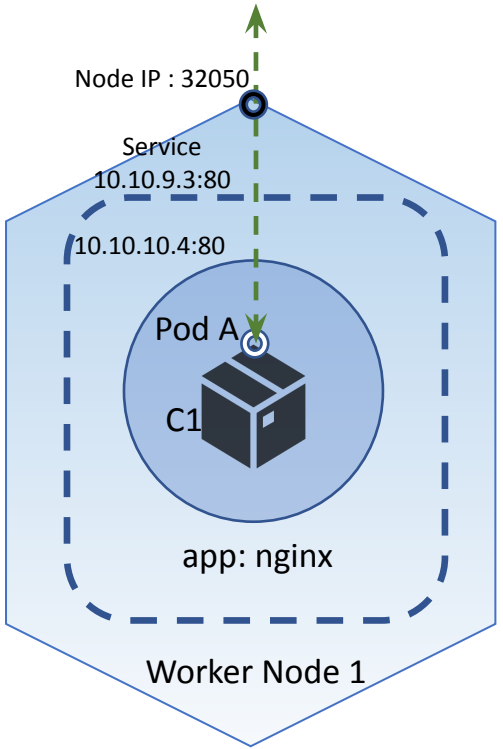
K21Academy
Learn Cloud from Experts

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe svc demo-pod
Name:              demo-pod
Namespace:         default
Labels:            name=demo-pod
Annotations:       <none>
Selector:          name=demo-pod
Type:              ClusterIP
IP:                10.108.93.193
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.46.0.1:80
Session Affinity:  None
Events:            <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
apiVersion: v1
kind: Service
metadata:
  name: demo-pod
  labels:
    name: demo-pod
spec:
  type: ClusterIP
  ports:
  - port: 80
    protocol: TCP
  selector:
    name: demo-pod
~
~
~
```

Kubernetes Service - NodePort

- Exposes the Pod to outside the cluster
- Assigns a Private IP Address in Cluster IP range
- Assigns a port on the worker nodes to expose it outside cluster
- Provides external connectivity



Kubernetes Service - NodePort

K21Academy
Learn Cloud from Experts

```
$ kubectl edit svc demo-pod
```

```
spec:
  clusterIP: 10.108.93.193
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    name: demo-pod
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
demo-pod     NodePort    10.108.93.193   <none>        80:32561/TCP   18m
```

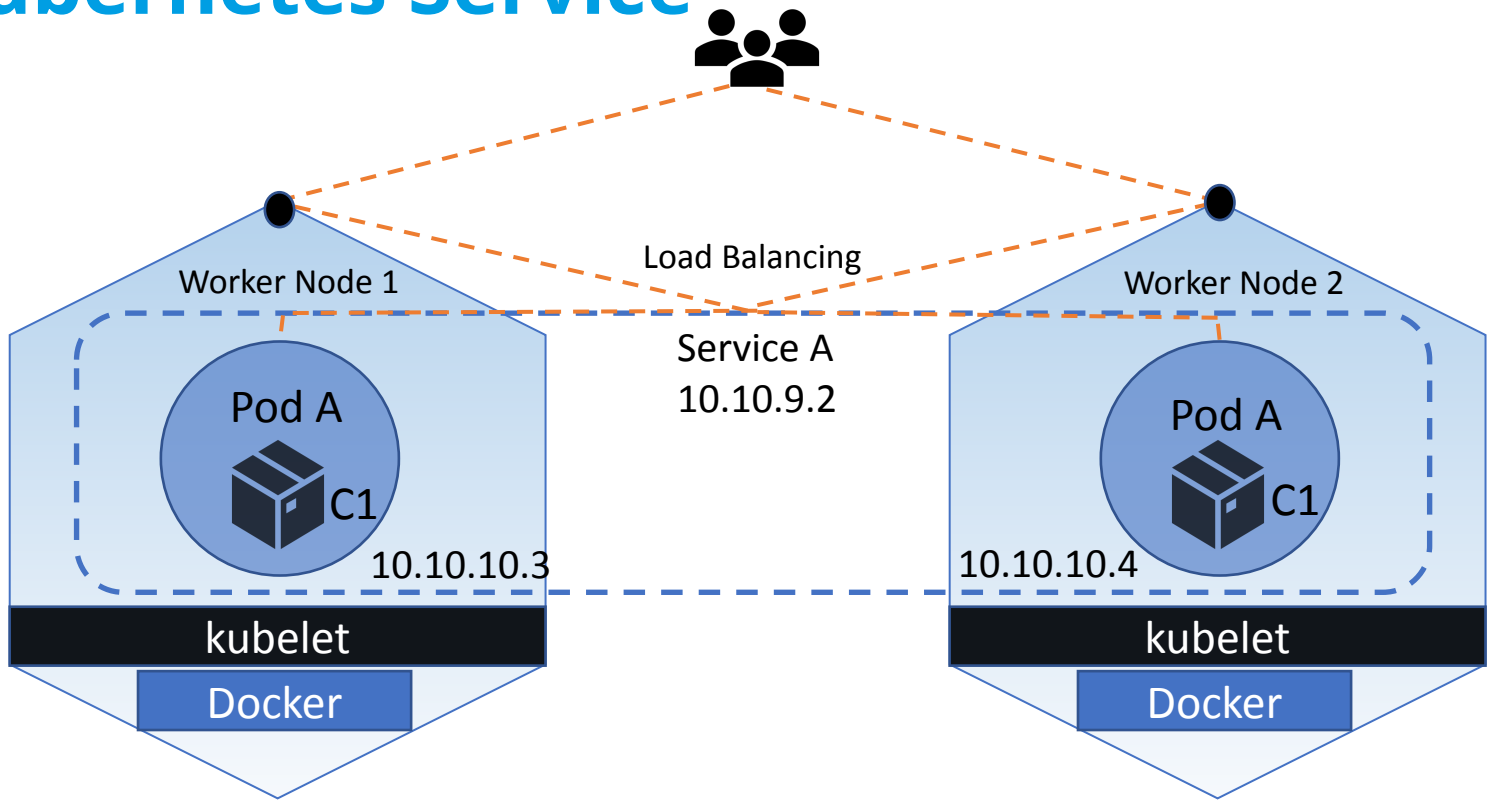
Kubernetes Service - NodePort

K21Academy
Learn Cloud from Experts

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe svc demo-pod
Name:                     demo-pod
Namespace:                default
Labels:                   name=demo-pod
Annotations:              <none>
Selector:                 name=demo-pod
Type:                     NodePort
IP:                       10.108.93.193
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  32561/TCP
Endpoints:                10.46.0.1:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
apiVersion: v1
kind: Service
metadata:
  name: demo-pod
  labels:
    name: demo-pod
spec:
  type: NodePort
  ports:
  - port: 80
    protocol: TCP
  selector:
    name: demo-pod
~
~
~
```

Kubernetes Service

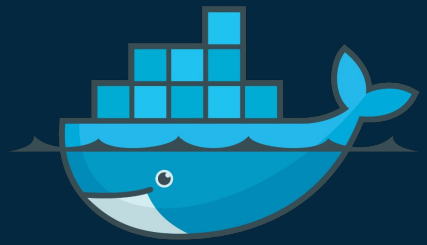


Kubernetes DNS Service

K21Academy
Learn Cloud from Experts

- Kubernetes clusters automatically configure an internal DNS service to provide a lightweight mechanism for service discovery
- The full DNS A record of a Kubernetes service will look like the following example: **service.namespace.svc.cluster.local**
- A pod would have a record in this format, reflecting the actual IP address of the pod: 10.32.0.125.namespace.pod.cluster.local
- If you're addressing a service in the same namespace, you can use just the service name to contact it: **other-service**
- If the service is in a different namespace, add it to the query: **other-service.other-namespace**

Questions



+

K21Academy
Learn Cloud from Experts

Deploy Highly Available Application

Kind: Deployment

- Deployment Controller defines the state of Deployment Objects, like Pods and ReplicaSets
- Deployments are used to create and deploy, scale, monitor and roll back
- Manage the state of Pods and ReplicaSets on the system
- Make application highly available, scalable and self-healing

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
~
~
~
~
~
~
~
~
```

Kind: Deployment

- kubectl get deployments

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           18s
```

- **NAME** lists the names of the Deployments in the namespace
- **READY** displays how many replicas of the application are available to the users. It follows the pattern ready/desired
- **UP-TO-DATE** displays the number of replicas that have been updated to achieve the desired state
- **AVAILABLE** displays how many replicas of the application are available to the users
- **AGE** displays the amount of time that the application has been running

ReplicaSet

K21Academy
Learn Cloud from Experts

- kubectl get rs

```
NAME                                DESIRED   CURRENT   READY   AGE
nginx-deployment-75675f5897   3         3         3       18s
```

- **NAME** lists the names of the ReplicaSets in the namespace
- **DESIRED** displays the desired number of replicas of the application, which you define when you create the Deployment. This is the desired state
- **CURRENT** displays how many replicas are currently running
- **READY** displays how many replicas of the application are available to the users
- **AGE** displays the amount of time that the application has been running

Pods

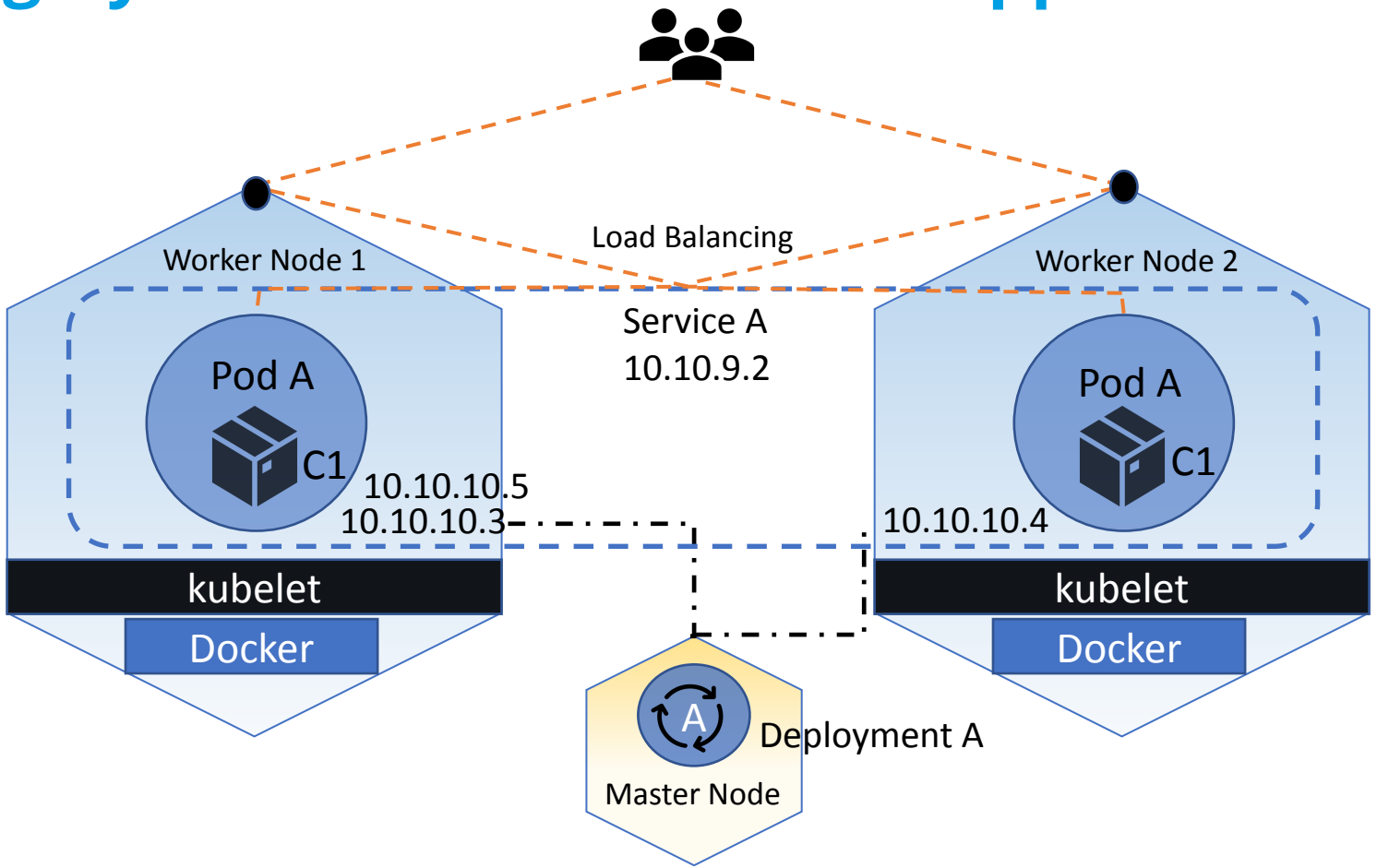
- kubectl get pods

```
NAME                                READY     STATUS    RESTARTS   AGE
nginx-deployment-75675f5897-7ci7o   1/1       Running   0          18s
nginx-deployment-75675f5897-kzszj   1/1       Running   0          18s
nginx-deployment-75675f5897-qqcnn   1/1       Running   0          18s
```

- **NAME** lists the names of the pods
- **READY** displays how many containers of the application are available to the users
- **STATUS** displays the where is the pod in the life-cycle
- **RESTARTS** shows number of time pod has been restarted by the kubelet
- **AGE** displays the amount of time that the application has been running

Highly Available and Scalable Application

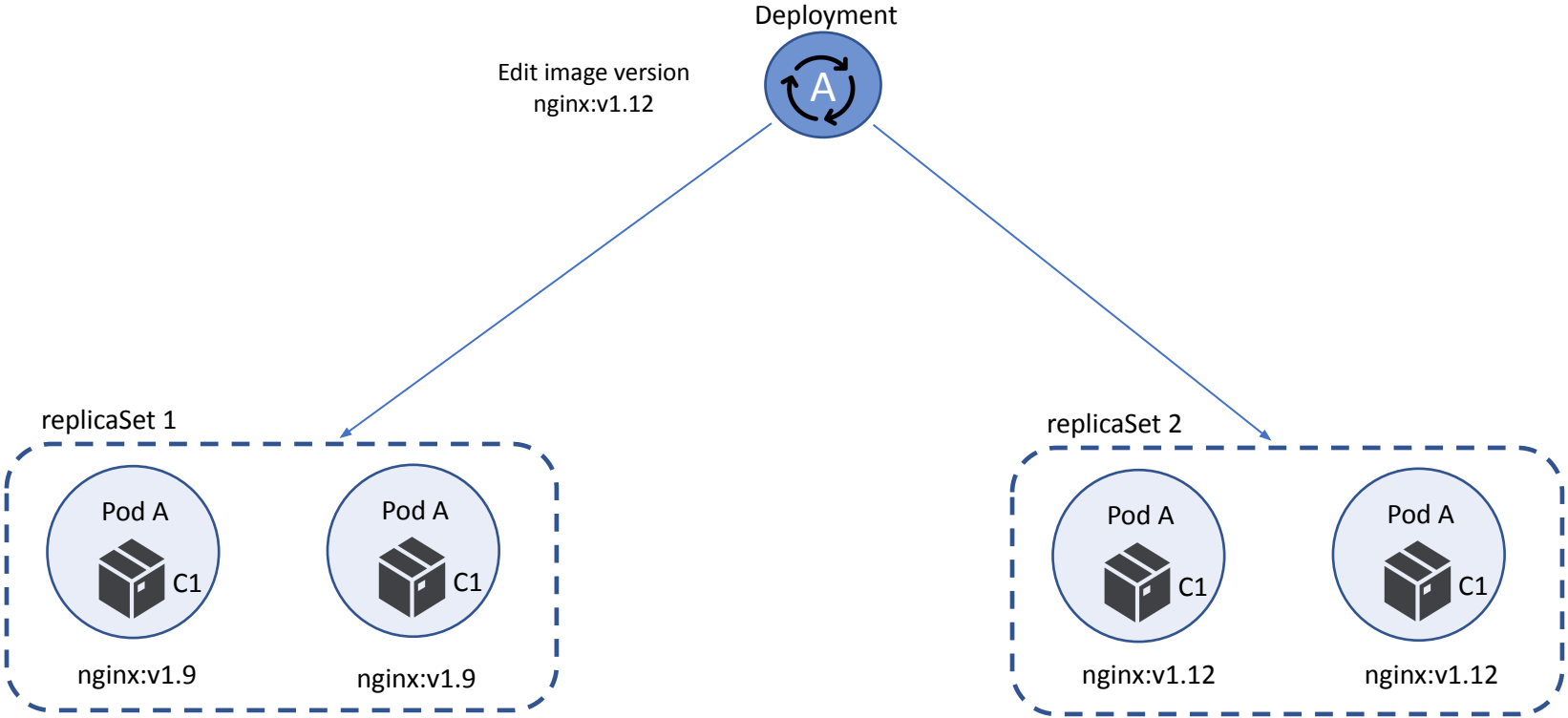
K21Academy
Learn Cloud from Experts



Responsibilities - Deployment

- Changing or updating Pod state
- Scaling up of the deployment
- To rollout, ReplicaSet to create Pods and monitor its status
- Roll back to a previous Deployment
- Pause Deployment to fix template

Rolling Changes - Deployment



Deployment Strategy

K21Academy
Learn Cloud from Experts

- StrategyType:
  - "Recreate" or "RollingUpdate”
  - "RollingUpdate" is the default value
- RollingUpdate
  - **maxUnavailable:** An optional field that specifies the maximum number of Pods that can be unavailable during the update process
  - **maxSurge:** An optional field that specifies the maximum number of Pods that can be created over the desired number of Pods
  - The value can be an absolute number or %

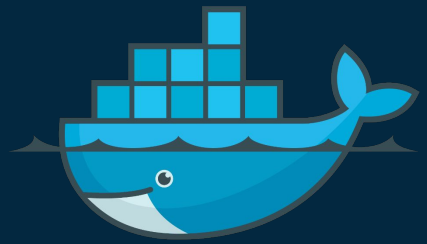
```
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Thu, 30 Nov 2017 10:56:25 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision=2
Selector:               app=nginx
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
   Containers:
    nginx:
      Image:        nginx:1.16.1
      Port:         80/TCP
      Environment:  <none>
      Mounts:       <none>
    Volumes:        <none>
```

Deployment Parameters

K21Academy
Learn Cloud from Experts

- **Progress Deadline Seconds:** Specifies the number of seconds we want to wait for our Deployment to progress before the system reports back that the Deployment has failed
- **Min Ready Seconds:** Specifies the minimum number of seconds for which a newly created Pod should be ready without any of its containers crashing, for it to be considered available
- **Revision History Limit:** Specifies the number of old ReplicaSets to retain to allow rollback. Default, 10 old ReplicaSets will be kept.
- **Paused:** Boolean field for pausing and resuming a Deployment. A Deployment is not paused by default when it is created.

Questions

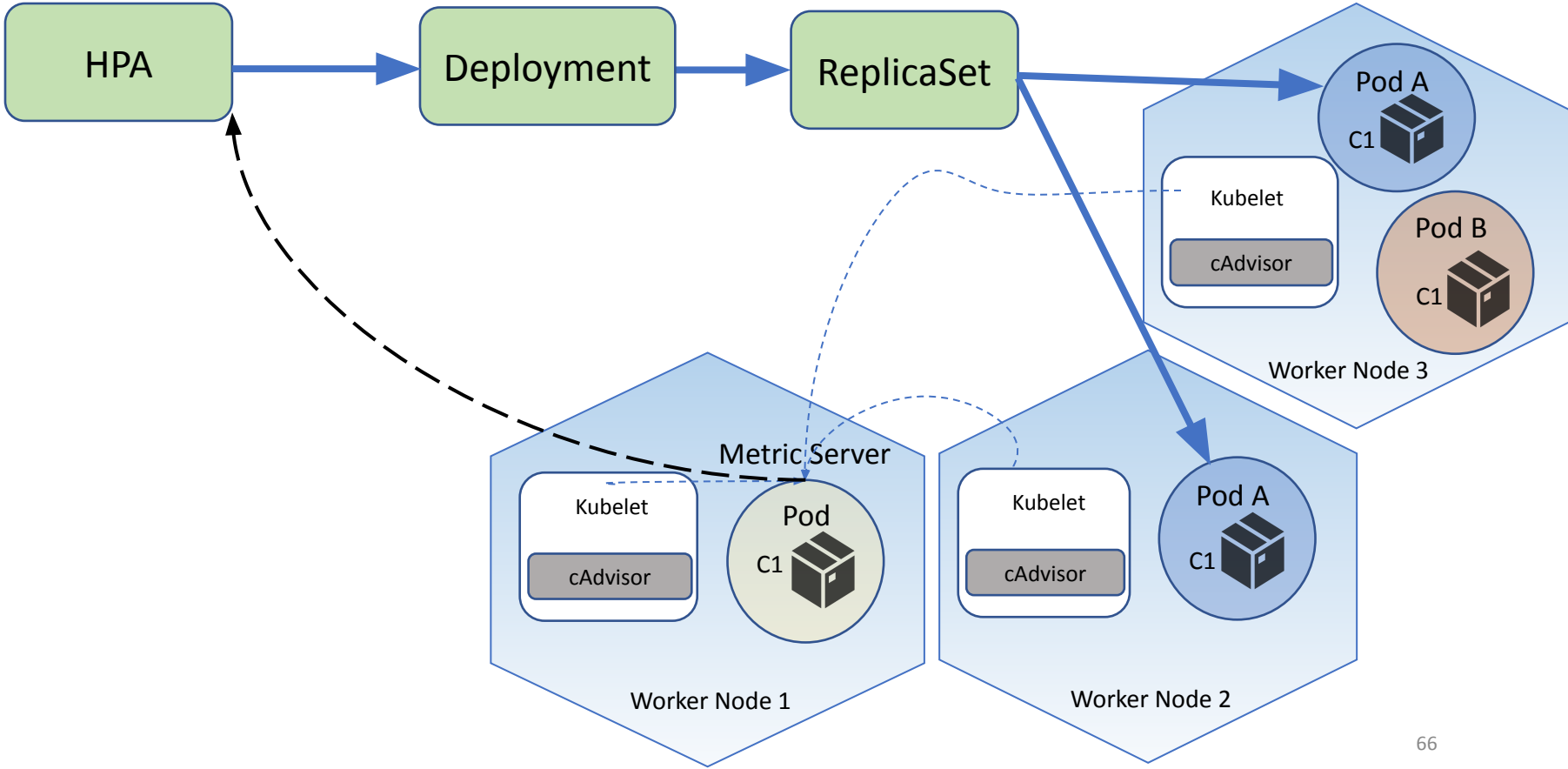


+

K21Academy
Learn Cloud from Experts

Application Autoscaling

Autoscaling with HPA

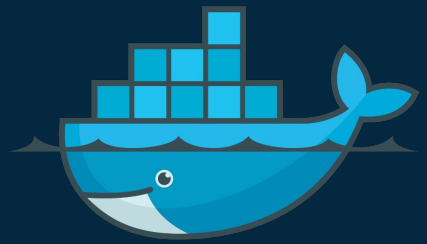


Horizontal Pod Autoscaler

K21Academy
Learn Cloud from Experts

- HPA automatically scales the number of pods in a deployment based on observed CPU utilization, pod or object metrics
- HPA does not apply to objects that can’t be scaled
- Metric server monitoring needs to be deployed in the cluster
- Metric server collects metrics from cAdvisor

Questions



+

K21Academy
Learn Cloud from Experts

Persistent Storage

Volume Types

K21Academy
Learn Cloud from Experts

- **EmptyDir**
  - An emptyDir volume is an empty directory created in the pod
  - Containers can read and write files in this directory
  - The directory's contents are removed once the pod is deleted

- **hostPath**
  - hostPath volumes directly mount the node's filesystem into the pod
  - Often not used in production

Volume Types

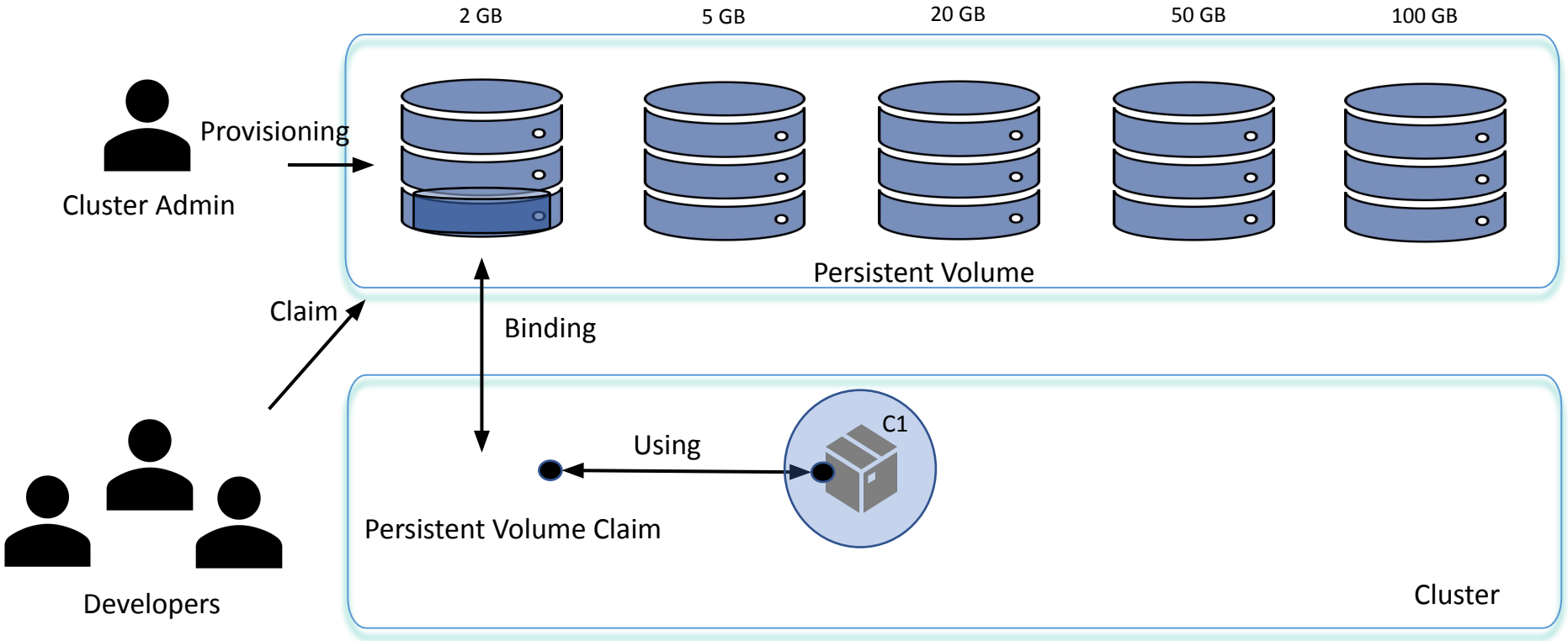
- **NFS**
  - NFS volumes are network file system shares that are mounted into the pod
  - An nfs volume's contents might exist beyond the pod's lifetime
  - Kubernetes doesn't do anything in regards to the management of the nfs

- **configMap and secret**
  - useful for inserting configuration and secret data into your containers

Persistent Storage

- **PersistentVolumes (PV)** are volume plugins like Volumes
- PVs lifecycle is managed by Kubernetes
- PVs are cluster resources that exist independently of Pods
- PVs can be provisioned by :
  - Static provisioning by an administrator
  - Dynamically provisioned using Storage Classes
- **PersistentVolumeClaim (PVC)** is a request for storage by a user

Lifecycle of a Volume and Claim



Volume and Access Modes

K21Academy
Learn Cloud from Experts

- **VolumeModes of PersistentVolumes:**
  - Filesystem : It's mounted into Pods into a directory
  - Block: Use a volume as a raw block device

- **Volume access modes are:**
  - ReadWriteOnce (RWO) – Mounted as read-write one
  - ReadOnlyMany (ROX) – Mounted read-only by many
  - ReadWriteMany (RWX) – Mounted as read-write by many

Yaml Definitions

K21Academy
Learn Cloud from Experts

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvvol-1
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    path: /opt/sfw
    server: kubeadm-master
    readOnly: false
~
~
~
~
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nfs-claim
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 500Mi
~
~
~
~
~
```

```
apiVersion: v1
kind: Pod
metadata:
  name: www
  labels:
    name: www
spec:
  containers:
  - name: www
    image: nginx:alpine
    ports:
      - containerPort: 80
        name: www
    volumeMounts:
      - name: nfs-vol
        mountPath: /usr/share/nginx/html
  volumes:
    - name: nfs-vol
      persistentVolumeClaim:
        claimName: nfs-claim
~
~
~
~
```

Persistent Storage – Phase

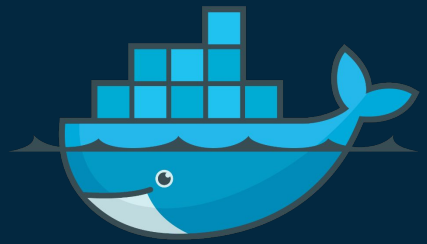
- Available -- a free resource that is not yet bound to a claim
- Bound -- the volume is bound to a claim
- Released -- the claim has been deleted, but the resource is not yet reclaimed by the cluster
- Failed -- the volume has failed its automatic reclamation

Important Points – Persistent Storage

K21Academy
Learn Cloud from Experts

- Volume abstraction solves issues of sharing storage between containers
- Saving from losing all data in case container is deleted and recreated
- Pods access storage by using the claim as a volume
- Claims must exist in the same namespace as the Pod using the claim

Questions



+

K21Academy
Learn Cloud from Experts

Advanced Scheduling

kube-scheduler

K21Academy
Learn Cloud from Experts

- Kubernetes default Scheduler
- Scheduling is an optimization process
- The task of the Kubernetes Scheduler is to choose a placement
- **Step 1: Filtering**
  - Scheduler determines the set of feasible placements, the set of placements that meet a set of given constraints
- **Step 2: Scoring**
  - Scheduler determines the set of viable placements, the set of feasible placements with the highest score

Advanced Scheduling

K21Academy
Learn Cloud from Experts

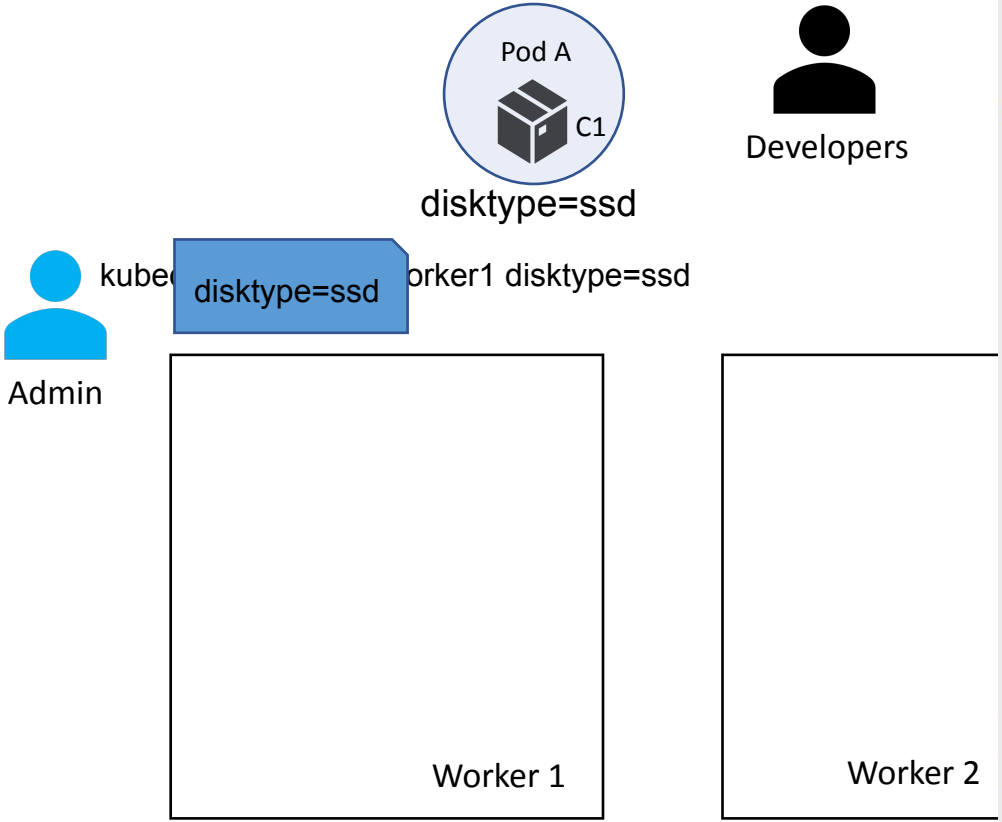
- From Kubernetes 1.6 it offers four advanced scheduling features:
  - Node Selector
  - Node Affinity
  - Pod Affinity/Anti-Affinity
  - Taints and Tolerations

Node Selector

- The simplest recommended form of node selection constraint.
- nodeSelector is a field of PodSpec
- It specifies a map of key-value pairs
-

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      nodeSelector:
        disktype: ssd
~
~
~
~
```

Node Selector




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      nodeSelector:
        disktype: ssd
~
~
~
~
```

Advanced Scheduling

K21Academy
Learn Cloud from Experts

- From Kubernetes 1.6 it offers four advanced scheduling features:
  - Node Selector
  - NodeAffinity
  - Pod Affinity/Anti-Affinity
  - Taints and Tolerations

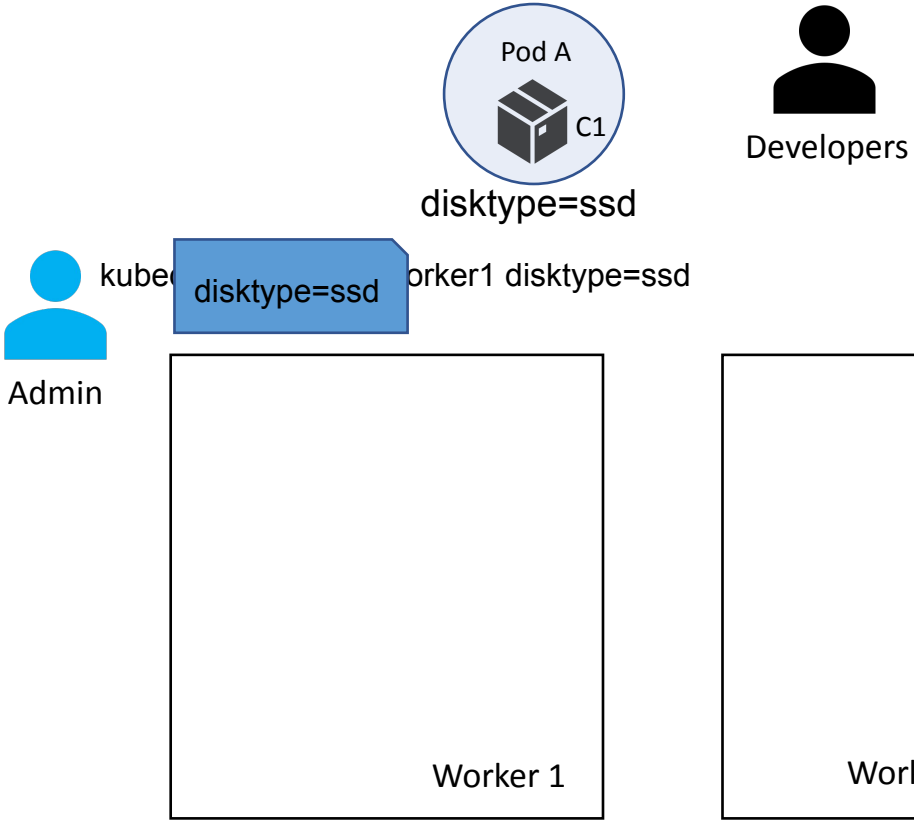
Node Affinity

- Similar to nodeSelector
- Allows to constrain based on labels on the node
- Types of node affinity:
  - preferredDuringSchedulingIgnoredDuringExecution
  - requiredDuringSchedulingIgnoredDuringExecution

Node Affinity

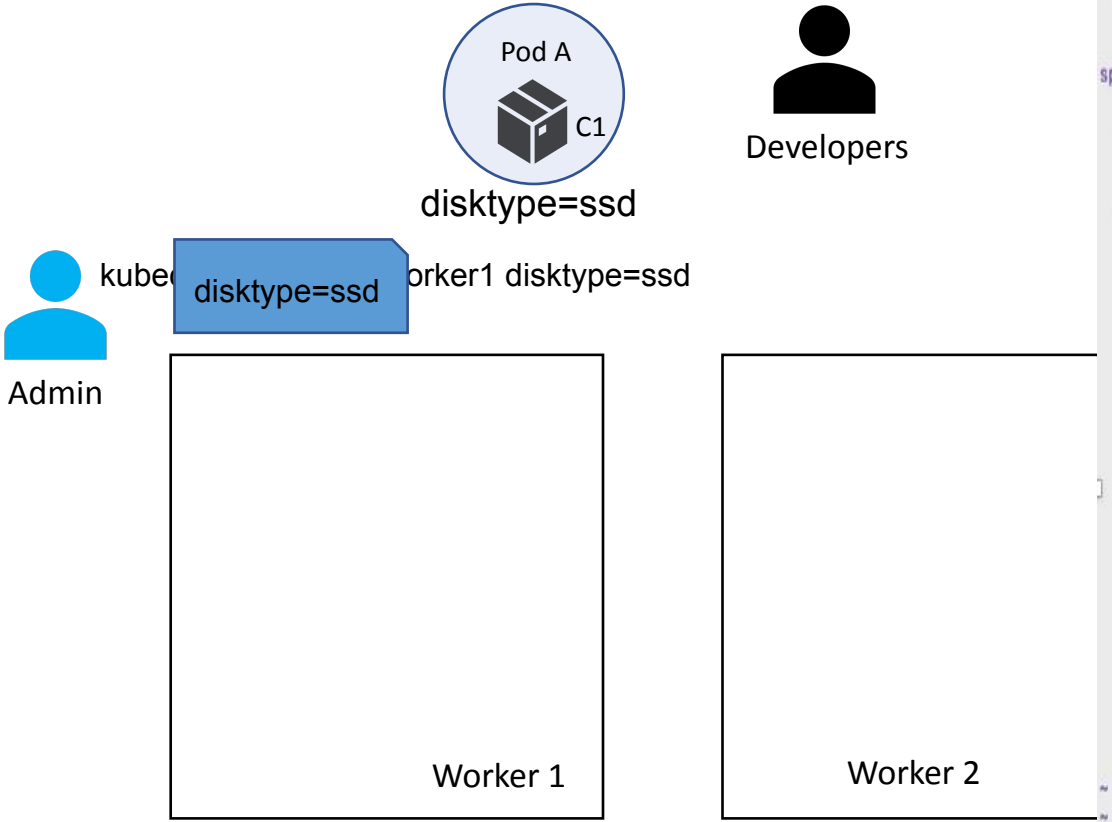
- Similar to nodeSelector
- Allows to constrain based on labels on the node
- Types of node affinity:
  - preferredDuringSchedulingIgnoredDuringExecution
  - requiredDuringSchedulingIgnoredDuringExecution
- Weight: int64 the numeric multiplier for the node scores that the prioritize call generates. The weight should be a positive integer

Node Affinity – Reqd-In




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: disktype
                operator: In
                values:
                - ssd
```

Node Affinity – Pref-In




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 1
            preference:
              matchExpressions:
              - key: disktype
                operator: In
                values:
                - ssd
```

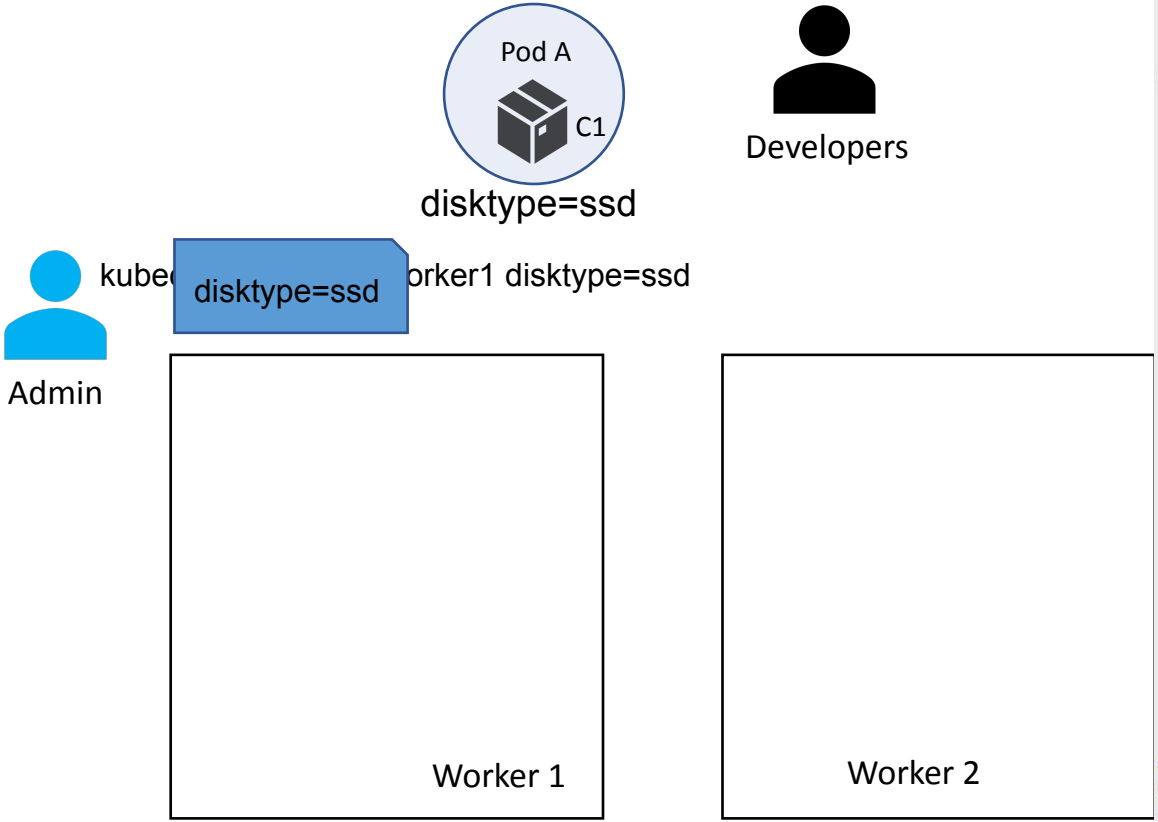
Node Affinity/Anti-Affinity

K21Academy
Learn Cloud from Experts

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 1
            preference:
              matchExpressions:
              - key: disktype
                operator: In
                values:
                - ssd
```

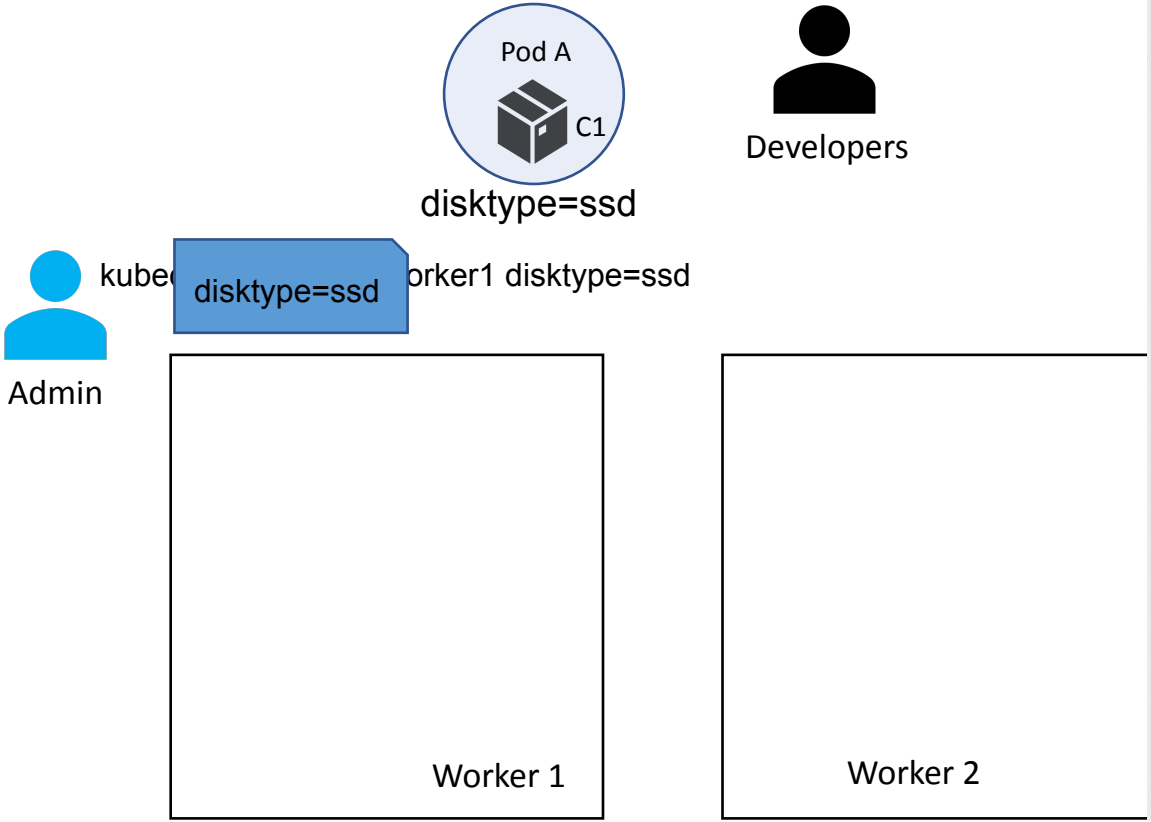
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: disktype
                operator: NotIn
                values:
                - ssd
```

Node Affinity – Reqd-NotIn

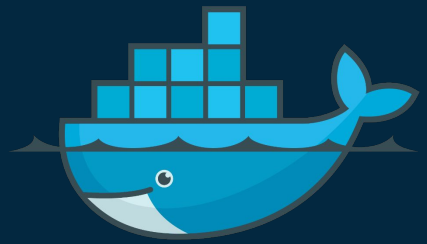



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: disktype
                operator: NotIn
                values:
                - ssd
~
~
~
```

Node Affinity – Pref-NotIn




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 1
            preference:
              matchExpressions:
              - key: disktype
                operator: NotIn
                values:
                - ssd
~
~
~
~
```



docker

+

kubernetes

K21Academy
Learn Cloud from Experts

Taint & Tolerations

Adv Scheduling - Taint & Tolerations

K21Academy
Learn Cloud from Experts

- Node Affinity was a property of Pods that attracts them to a set of nodes (either as a preference or a hard requirement)
- Taints are the opposite -- they allow a node to repel a set of pods
- Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints
- Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes

```
$
$ kubectl taint node aks-agentpool-40017546-vmss000001 disktype=magnetic:NoSchedule
node/aks-agentpool-40017546-vmss000001 tainted
```

```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod1
spec:
  containers:
   - name: nginx
     image: nginx
  tolerations:
  - key: "disktype"
    operator: "Equal"
    value: "magnetic"
    effect: "NoSchedule"
~
~
~
~
```

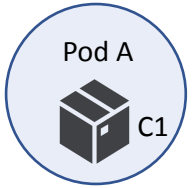
Taint Effect and Operator

K21Academy
Learn Cloud from Experts

- **Effect:**
  - **NoSchedule:** New pods that do not match the taint are not scheduled onto that node but the existing pods on the node remain
  - **PreferNoSchedule:** New pods that do not match the taint might be scheduled onto that node, but the scheduler tries not to and the existing pods on the node remain
  - **NoExecute:** New pods that do not match the taint cannot be scheduled onto that node and the existing pods on the node that donot have a matching toleration are removed
- **Operator:**
  - **Equal:** The key/value/effect all three parameters must match. This is the default
  - **Exists:**

```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod1
spec:
  containers:
   - name: nginx
     image: nginx
  tolerations:
  - key: "disktype"
    operator: "Equal"
    value: "magnetic"
    effect: "NoSchedule"
~
~
~
~
```

Taint– NoSchedule

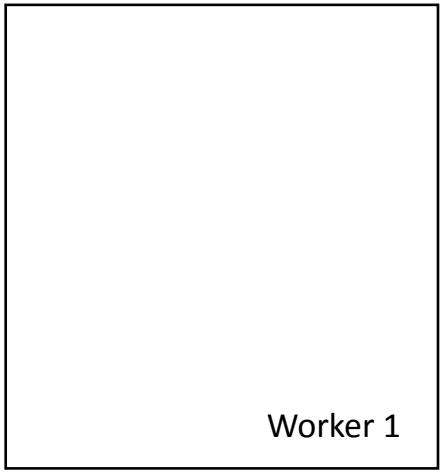


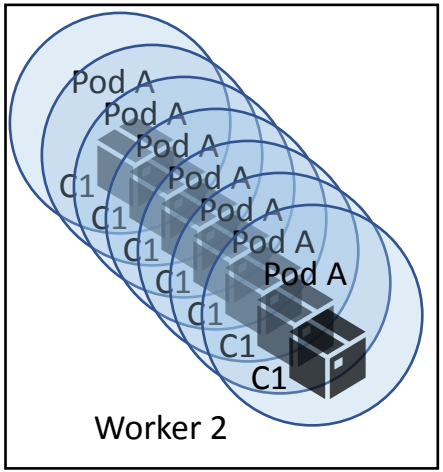
Developers

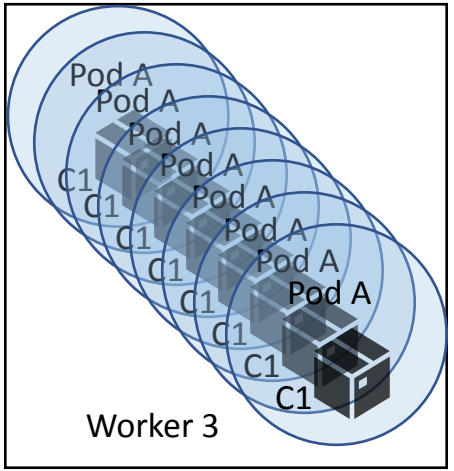
```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod1
spec:
  containers:
   - name: nginx
     image: nginx
```

Admin

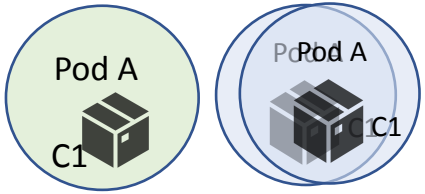
kubectl taint node worker1 disktype=magnetic:NoSchedule







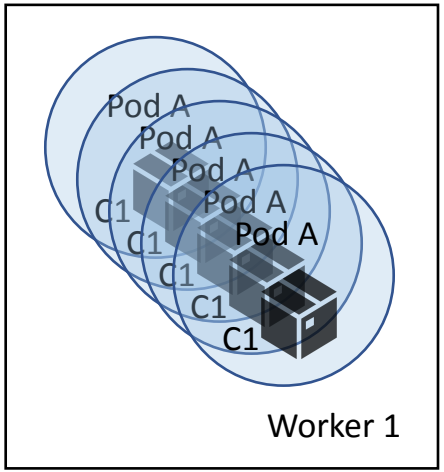
Taint & Toleration -NoSchedule

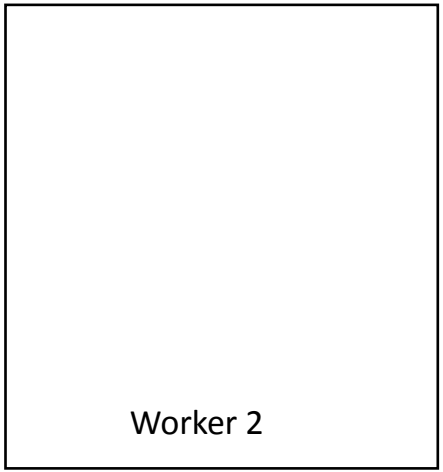


Developers

Admin

kubectl taint node worker1 disktype=magnetic:NoSchedule






```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod1
spec:
  containers:
   - name: nginx
     image: nginx
  tolerations:
  - key: "disktype"
    operator: "Equal"
    value: "magnetic"
    effect: "NoSchedule"
~
~
~
~
```

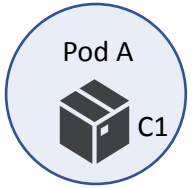
Taint Effect and Operator

K21Academy
Learn Cloud from Experts

- **Effect:**
  - **NoSchedule:** New pods that do not match the taint are not scheduled onto that node but the existing pods on the node remain
  - **PreferNoSchedule:** New pods that do not match the taint might be scheduled onto that node, but the scheduler tries not to and the existing pods on the node remain
  - **NoExecute:** New pods that do not match the taint cannot be scheduled onto that node and the existing pods on the node that donot have a matching toleration are removed
- **Operator:**
  - **Equal:** The key/value/effect all three parameters must match. This is the default
  - **Exists:**

```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod1
spec:
  containers:
   - name: nginx
     image: nginx
  tolerations:
  - key: "disktype"
    operator: "Equal"
    value: "magnetic"
    effect: "NoSchedule"
~
~
~
~
```

Taint– NoExecute

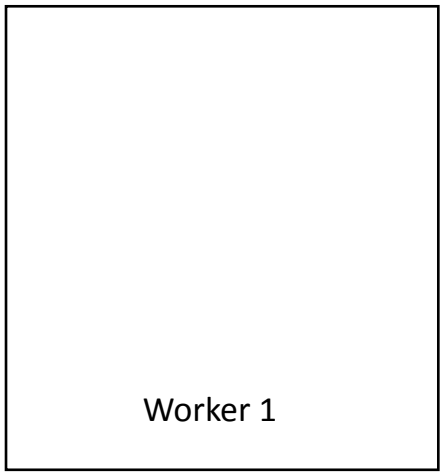


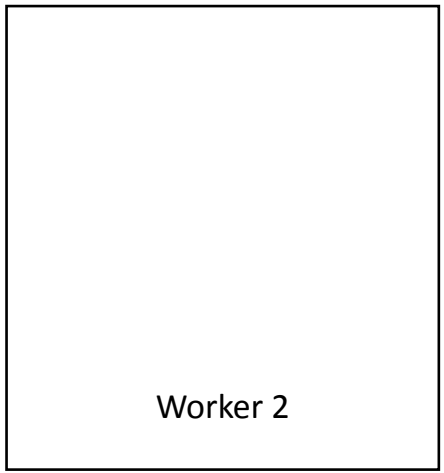
Developers

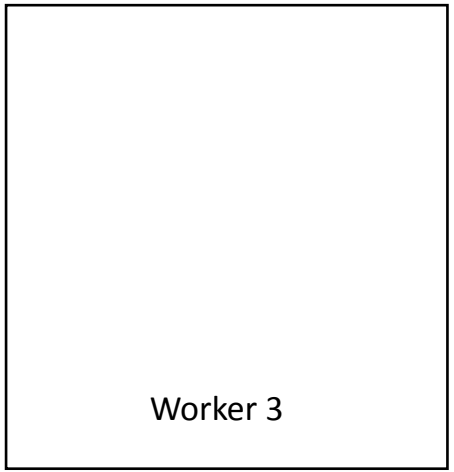
```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod1
spec:
  containers:
   - name: nginx
     image: nginx
```

kubectl taint node worker1 disktype=magnetic:NoExecute

Admin







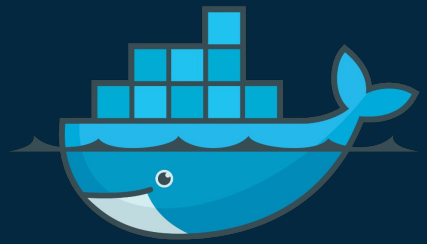
Taint Effect and Operator

K21Academy
Learn Cloud from Experts

- **Effect:**
  - **NoSchedule:** New pods that do not match the taint are not scheduled onto that node but the existing pods on the node remain
  - **PreferNoSchedule:** New pods that do not match the taint might be scheduled onto that node, but the scheduler tries not to and the existing pods on the node remain
  - **NoExecute:** New pods that do not match the taint cannot be scheduled onto that node and the existing pods on the node that donot have a matching toleration are removed
- **Operator:**
  - **Equal:** The key/value/effect all three parameters must match. This is the default
  - **Exists:**

```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod1
spec:
  containers:
   - name: nginx
     image: nginx
  tolerations:
  - key: "disktype"
    operator: "Equal"
    value: "magnetic"
    effect: "NoSchedule"
~
~
~
~
```

Questions



docker

+

kubernetes

K21Academy
Learn Cloud from Experts

Kubernetes Security

3 Step Security Check

K21Academy
Learn Cloud from Experts

- The kube-apiserver goes through 3 step process for every API call
  - **Authentication:** Can be achieved with an X509 certificate, a token, or a webhook to an LDAP server
  - **Authorization:** RBAC handles this, it checks if the API call should be allowed or not
  - **Admission Control**

RBAC - Role Based Access Control

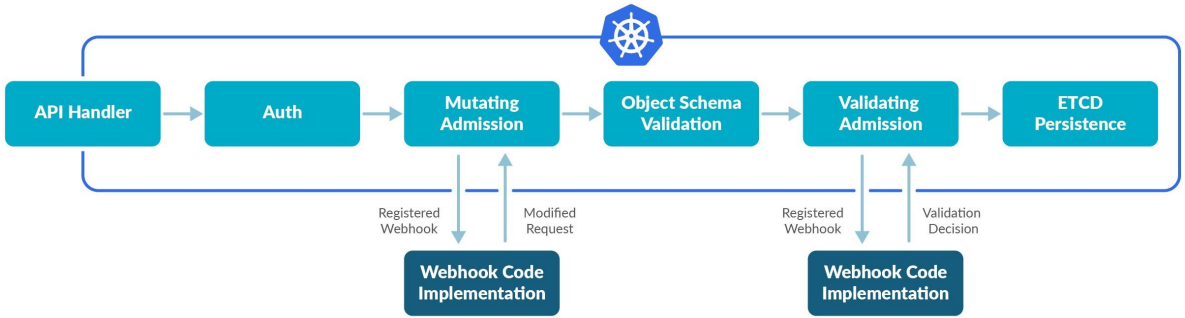
K21Academy
Learn Cloud from Experts

- Determine or create namespace
- Create certificate credentials for user
- Set the credentials for the user to the namespace using a context
- Create a role for the expected task set
- Bind the user to the role
- Verify the user has limited access

Admission Controllers

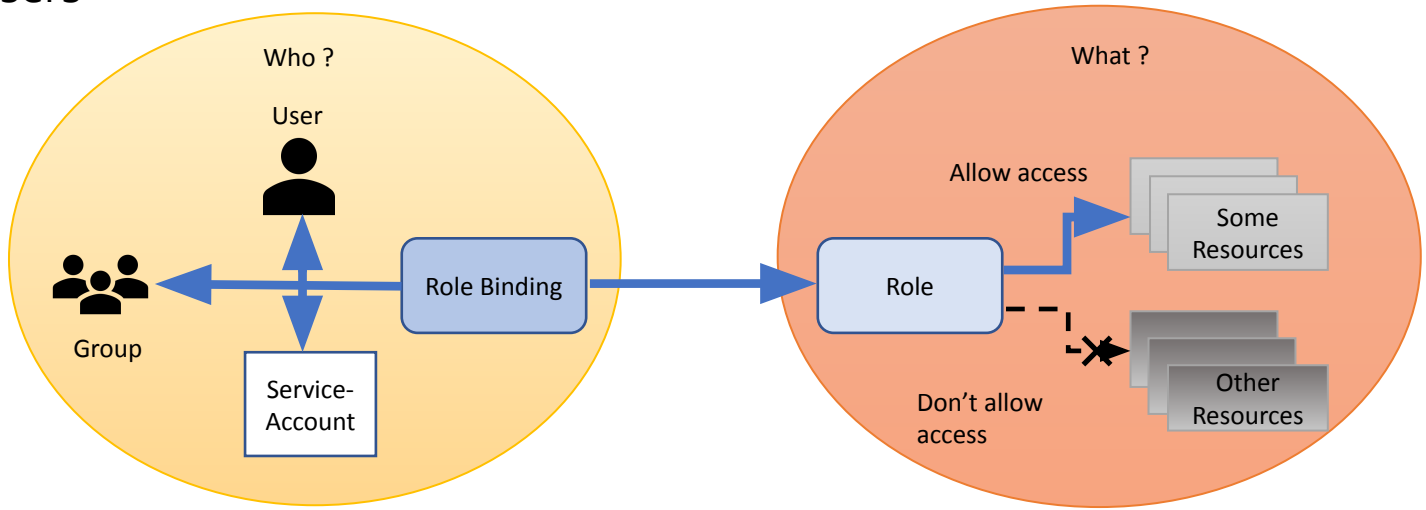
- An admission controller intercepts and processes requests to the Kubernetes API prior to persistence of the object, but after the request is authenticated and authorized.
- The controllers are compiled into the kube-apiserver binary, and may only be configured by the cluster administrator.

- **Admission Controller Example:**
- AlwaysPullImages
- CertificateApproval
- CertificateSigning
- DenyServiceExternalIPs



Role and RoleBinding

- Role is a set of rules that represent a set of purely additive permissions
- Role is used to grant access to resources within a single namespace
- RoleBinding binds the permissions defined in a role to a user or set of users



Cluster Role and RoleBinding

K21Academy
Learn Cloud from Experts

- **ClusterRole** : A role which has a cluster-wide scope. ClusterRole can also be used to grant access to:
  - define permissions on namespaced resources and be granted within individual namespace(s)
  - define permissions on namespaced resources and be granted across all namespaces
  - define permissions on cluster-scoped resources

- **ClusterRoleBinding** references a ClusterRole to grant the permissions to namespaced resources defined in the ClusterRole
- This allows administrators to define a set of common roles for the entire cluster, then reuse them within multiple namespaces

Network Security

K21Academy
Learn Cloud from Experts

- Network policies are namespaced resources, we create policy for each namespace
- By default, pods are non-isolated; they accept traffic from any source
- NetworkPolicy uses labels to select pods and define rules which specify what traffic is allowed to the selected pods
- **If no network policies apply to a pod, then all network connections to and from it are permitted**
- Once NetworkPolicy is attached to a pod, that pod will reject any connections that are not allowed by any NetworkPolicy
- **Pods are "isolated" if at least one network policy applies to them; if no policies apply, they are "non-isolated"**

DENY all Ingress traffic

K21Academy
Learn Cloud from Experts

```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-ingress
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

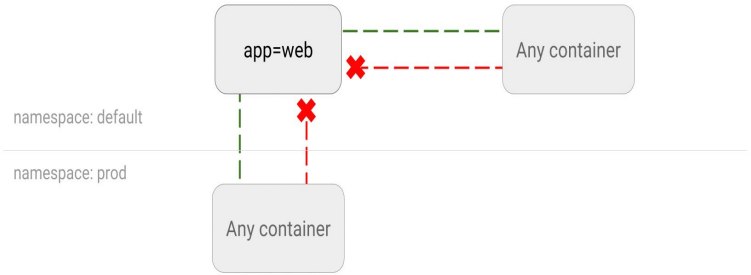
- "default" isolation policy for a namespace by creating a NetworkPolicy
- It selects all pods but does not allow any ingress traffic to those pods

DENY all Ingress traffic – specific pod

K21Academy
Learn Cloud from Experts

```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-deny-all
spec:
  podSelector:
    matchLabels:
      app: web
  ingress: []
```

- Target Pods with app=web label to police the network
- This NetworkPolicy will drop all traffic to pods of an application, selected using Pod Selectors



Allow all Ingress traffic

K21Academy
Learn Cloud from Experts

- This manifest file is having empty the spec.ingress field
- Allow any traffic into all the Pod
- Target Pods with app=web label to police the network
- This NetworkPolicy will allow all traffic to pods of an application, selected using Pod Selectors
- Empty ingress rule ({}) allows traffic from all pods in the current namespace, as well as other namespaces. It corresponds to:

```
- from:

    podSelector: {}

     namespaceSelector: {}
```

```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-ingress
spec:
  podSelector: {}
  ingress:
  - {}
  policyTypes:
  - Ingress
```

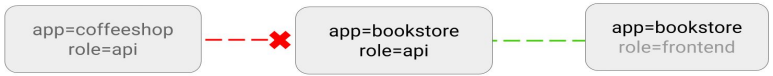
```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-all
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: web
  ingress:
  - {}
  policyTypes:
  - Ingress
~
~
~
```

Allow to and from (same namespace)

K21Academy
Learn Cloud from Experts

- Each network policy has a **podSelector** field, which selects a group of (zero or more) pods
- When a pod is selected by a network policy, the network policy is said to apply to it
- When the network policy is created, all the pods that it applies to are allowed to make or accept the connections listed in it
- A network policy is a list of allowed connections

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

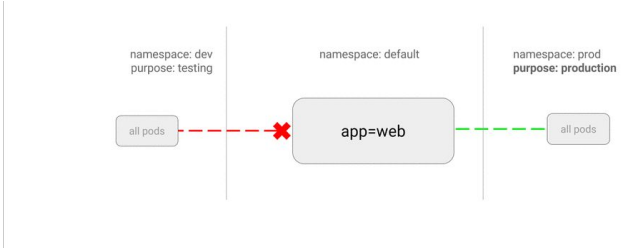


Allow to and from (namespaceSelector)

K21Academy
Learn Cloud from Experts

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: secondary
  name: web-allow-all-namespaces
spec:
  podSelector:
    matchLabels:
      app: web
  policyType:
  - Ingress
  ingress:
  - from:
    - namespaceSelector: {}
~
~
~
```

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-prod
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          purpose: production
~
```



ALLOW traffic from specific pods in a namespace

K21Academy
Learn Cloud from Experts

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-all-ns-monitoring
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
    - from:
      - namespaceSelector:     # chooses all pods in namespaces labelled with team=operations
          matchLabels:
            team: operations
        podSelector:           # chooses pods with type=monitoring
          matchLabels:
            type: monitoring
~
~
~
~
```

Egress and Ingress

K21Academy
Learn Cloud from Experts

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

- **Egress rules:**

Allows connections from any pod in the "default" namespace with the label "role=db" to CIDR 10.0.0.0/24 on TCP port 5978

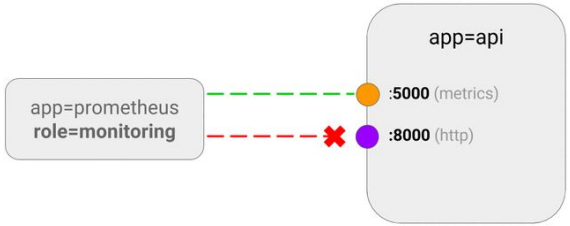
- **Ingress rules :**

Allows connections to all pods in the "default" namespace with the label "role=db" on TCP port 6379 from:

- Any pod in the "default" namespace with the label "role=frontend"
- Any pod in a namespace with the label "project=myproject"
- IP addresses in the ranges 172.17.0.0–172.17.0.255 and 172.17.2.0–172.17.255.255 (ie, all of 172.17.0.0/16 except 172.17.1.0/24)

ALLOW traffic only to a port of a Pod

K21Academy
Learn Cloud from Experts




```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: api-allow-5000
spec:
  podSelector:
    matchLabels:
      app: apiserver
  policyTypes:
  - Ingress
  ingress:
  - ports:
    - port: 5000
    from:
    - podSelector:
        matchLabels:
          role: monitoring
~
~
```

Security Context

K21Academy
Learn Cloud from Experts

- A security context defines privilege and access control settings for a Pod or Container
- Permission to access an object, like a file, is based on user ID (UID) and group ID (GID)
- Running as privileged or unprivileged user
- Give a process some privileges, but not all the privileges of the root user

Security Context

K21Academy
Learn Cloud from Experts

```
apiVersion: v1
kind: Pod
metadata:
  name: alpine-user-context
spec:
  containers:
  - name: main
    image: alpine
    command: ["/bin/sleep", "999999"]
    securityContext:
      runAsUser: 405
~
~
~
~
~
```

```
apiVersion: v1
kind: Pod
metadata:
  name: alpine-nonroot
spec:
  containers:
  - name: main
    image: alpine
    command: ["/bin/sleep", "999999"]
    securityContext:
      runAsNonRoot: true
~
~
~
~
~
~
```

```
apiVersion: v1
kind: Pod
metadata:
  name: privileged-pod
spec:
  containers:
  - name: main
    image: alpine
    command: ["/bin/sleep", "999999"]
    securityContext:
      privileged: true
~
~
```

Security Context

K21Academy
Learn Cloud from Experts

```
apiVersion: v1
kind: Pod
metadata:
  name: kernelchange-pod
spec:
  containers:
  - name: main
    image: alpine
    command: ["/bin/sleep", "999999"]
    securityContext:
      capabilities:
        add:
        - SYS_TIME
~
~
~
~
~
~
~
```

```
apiVersion: v1
kind: Pod
metadata:
  name: remove-capabilities
spec:
  containers:
  - name: main
    image: alpine
    command: ["/bin/sleep", "999999"]
    securityContext:
      capabilities:
        drop:
        - CHOWN
~
~
~
~
~
~
```

```
apiVersion: v1
kind: Pod
metadata:
  name: readonly-pod
spec:
  containers:
  - name: main
    image: alpine
    command: ["/bin/sleep", "999999"]
    securityContext:
      readOnlyRootFilesystem: true
    volumeMounts:
    - name: my-volume
      mountPath: /volume
      readOnly: false
  volumes:
  - name: my-volume
    emptyDir:
~
~
~
~
~
```

Configuration Parameters

- Application is tunable using configuration settings
- Environment variables and flags change logic in our app
- In containers we can not depend on the host's environment variables
- Each container has its own environment

Drawbacks & Solution

- The drawback of Docker and Kubernetes environment variables is that they are tied to the container or deployment
- If we wish to change them, we have to rebuild the container or modify the deployment
- Even worse, if we wish to use the variable with multiple containers or deployments, we have to duplicate the data!
- Kubernetes has solved this problem with Secrets (for confidential data) and ConfigMaps (for non-confidential data)

ConfigMap

- ConfigMap API stores the configuration data in key-value pairs
- It can be either consumed in the Pods or it can be used to store the data of a system component
- ConfigMap resources store the configuration data in an organized manner
- We can change the configuration of the container without having to rebuild the container
- This configuration data can then be used as:
  - Environment variables
  - Command-line arguments for a container
  - Config files in a volume

Env Variable - ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
  namespace: default
data:
  mydata: hello_world

~
~
~
~
```

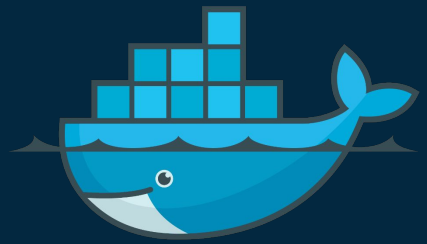
```
apiVersion: v1
kind: Pod
metadata:
  name: cm-pod
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    env:
      - name: cm
        valueFrom:
          configMapKeyRef:
            name: my-config
            key: mydata

~
~
```

ConfigMap as Volumes

```
apiVersion: v1
data:
  redis-config: |
    maxmemory 2mb
    maxmemory-policy allkeys-lru
kind: ConfigMap
metadata:
  name: example-redis-config
  namespace: default
~
~
~
~
```

```
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: redis
    env:
    - name: MASTER
      value: "true"
    ports:
    - containerPort: 6379
    resources:
      limits:
        cpu: "0.1"
    volumeMounts:
    - mountPath: /redis-master-data
      name: data
    - mountPath: /redis-master
      name: config
  volumes:
    - name: data
      emptyDir: {}
    - name: config
      configMap:
        name: example-redis-config
        items:
        - key: redis-config
          path: redis.conf
~
~
~
~
~
~
```

Questions



+

K21Academy
Learn Cloud from Experts

Cluster Resource Limit

Resource Quota

K21Academy
Learn Cloud from Experts

- Defines resource consumption quota per namespace
- The quota system keeps track of resources created by the users and ensures that the hard limits are not exceeded
- Compute resources like CPU and Memory can be limited in a namespace
- Number of objects of a kind can be limited per namespace
- When a cluster is shared among several teams, it becomes necessary to limit the amount of resources that can be used by them

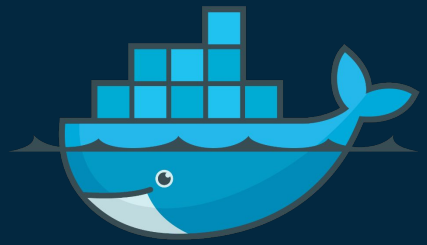
```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: quota
  namespace: quotas
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Limiting Resources

- Containers in a Pod can specify the resource requirements
- **Limits:** Specifies the max the container would go up to
- **Requests:** The minimum need for the container

```
apiVersion: v1
kind: Pod
metadata:
  name: quota-pod
  namespace: quotas
spec:
  containers:
  - name: quota-container
    image: nginx
    resources:
      limits:
        memory: "800Mi"
        cpu: "1000m"
      requests:
        memory: "600Mi"
        cpu: "350m"
~
~
~
```

Questions



docker

+

kubernetes

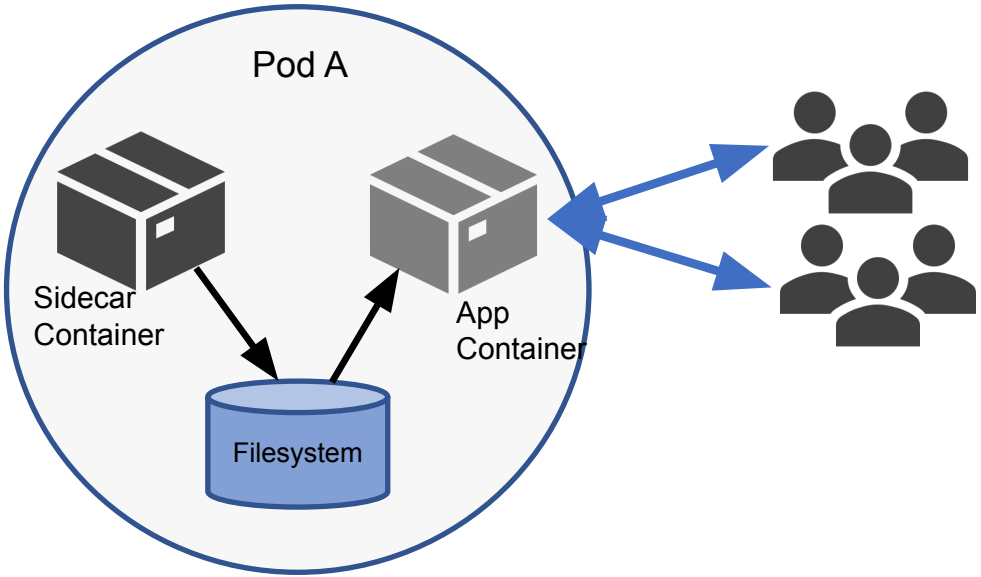
K21Academy
Learn Cloud from Experts

Multi Container - Sidecar Pattern

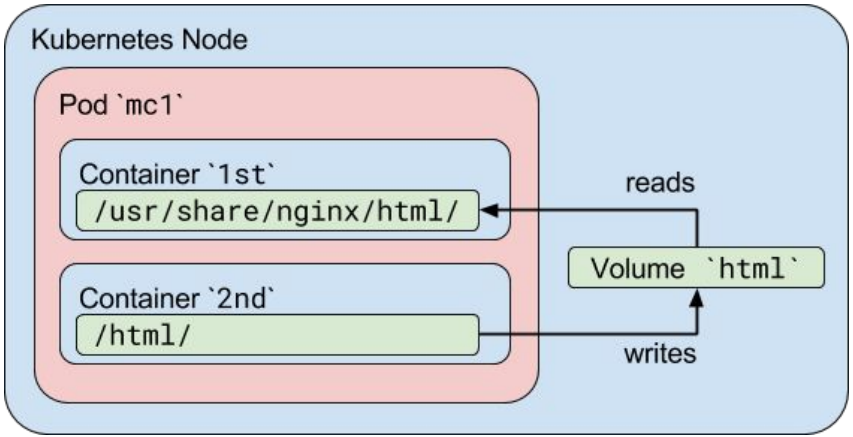
Multi Container - Sidecar Pattern

- Most common pattern for multi-container deployments
- Sidecars extend and enhance the main container
- Advantages:
  - The sidecar container is the unit of resource accounting and allocation
  - The container can be the unit of reuse, so sidecar containers can be paired with numerous different “main” containers
  - The container provides a failure containment boundary
  - The container is the unit of deployment, so can be upgraded or rolled independently

Multi Container - Sidecar Pattern

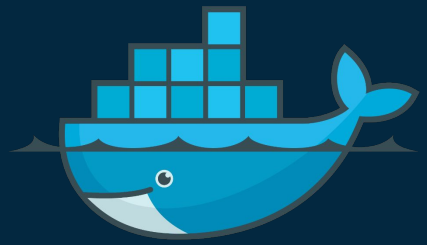


Multi Container - Sidecar Pattern




```
apiVersion: v1
kind: Pod
metadata:
  name: mc1
spec:
  volumes:
  - name: html
    emptyDir: {}
  containers:
  - name: 1st
    image: nginx
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
  - name: 2nd
    image: debian
    volumeMounts:
    - name: html
      mountPath: /html
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          date >> /html/index.html;
          sleep 1;
        done
~
```

Questions



docker

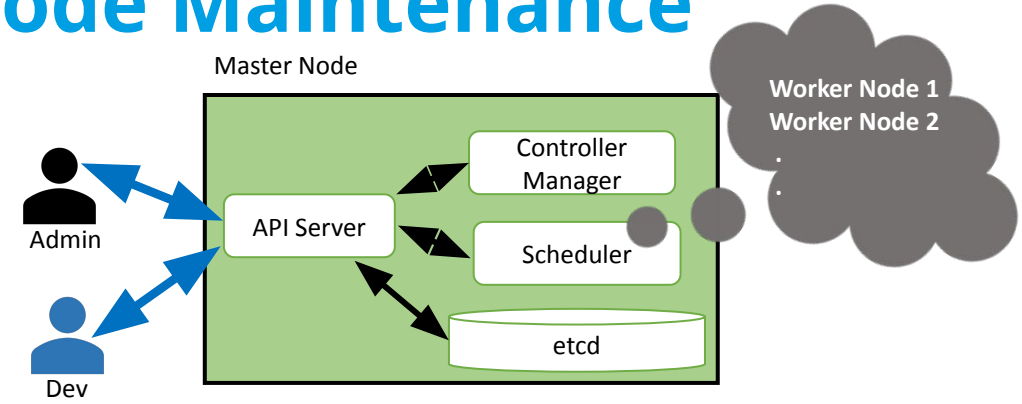
+

kubernetes

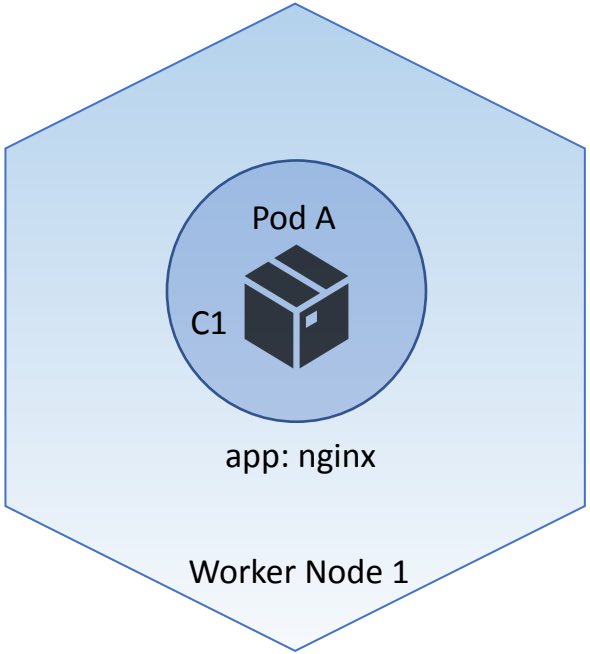
K21Academy
Learn Cloud from Experts

Maintenance & Troubleshooting

Node Maintenance



- Use **kubectl cordon** to mark the node as unschedulable
- Use **kubectl drain** to gracefully terminate all pods on the node while marking the node as unschedulable
- Use **kubectl uncordon** to mark the node schedulable again



K8s Control Plane Troubleshooting

K21Academy
Learn Cloud from Experts

- **Control Plane issues**
  - Kube-system processes
  - Kubelet daemon
  - Docker daemon

K8s Control Plane Log Path

- **Daemon logs – Kubelet and Docker**
  - Every node @ /var/log/syslog
- **Kube-system Pod logs on Master**
  - Master node @ /var/log/pods (or)
  - Master node @ /var/lib/docker/containers find respective container logs
- **Kube-system and App Pod logs on Worker**
  - Every node @ /var/log/pods (or)
  - Every node @ /var/lib/docker/containers find respective container logs

kube-system Process Logs

K21Academy
Learn Cloud from Experts

```
root@kubeadm-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu# kubectl logs kube-apiserver-kubeadm-master -n kube-system
Flag --insecure-port has been deprecated, This flag will be removed in a future version.
I0718 14:27:03.294371       1 server.go:618] external host was not specified, using 10.0.0.4
I0718 14:27:03.294756       1 server.go:148] Version: v1.18.6
I0718 14:27:03.814046       1 plugins.go:158] Loaded 12 mutating admission controller(s) successfully in the following order: NamespaceLifecycle,LimitRanger,ServiceAccount,N
odeRestriction,TaintNodesByCondition,Priority,DefaultTolerationSeconds,DefaultStorageClass,StorageObjectInUseProtection,RuntimeClass,DefaultIngressClass,MutatingAdmissionWeb
hook.
I0718 14:27:03.814302       1 plugins.go:161] Loaded 10 validating admission controller(s) successfully in the following order: LimitRanger,ServiceAccount,Priority,Persisten
tVolumeClaimResize,RuntimeClass,CertificateApproval,CertificateSigning,CertificateSubjectRestriction,ValidatingAdmissionWebhook,ResourceQuota.
I0718 14:27:03.822264       1 plugins.go:158] Loaded 12 mutating admission controller(s) successfully in the following order: NamespaceLifecycle,LimitRanger,ServiceAccount,N
odeRestriction,TaintNodesByCondition,Priority,DefaultTolerationSeconds,DefaultStorageClass,StorageObjectInUseProtection,RuntimeClass,DefaultIngressClass,MutatingAdmissionWeb
hook.
I0718 14:27:03.830255       1 plugins.go:161] Loaded 10 validating admission controller(s) successfully in the following order: LimitRanger,ServiceAccount,Priority,Persisten
tVolumeClaimResize,RuntimeClass,CertificateApproval,CertificateSigning,CertificateSubjectRestriction,ValidatingAdmissionWebhook,ResourceQuota.
I0718 14:27:03.833907       1 client.go:361] parsed scheme: "endpoint"
```

```
root@kubeadm-master:/home/ubuntu# kubectl logs etcd-kubeadm-master -n kube-system
[WARNING] Deprecated '--logger=capnslog' flag is set; use '--logger=zap' flag instead
2020-07-18 14:27:04.344001 I | etcdmain: etcd Version: 3.4.3
2020-07-18 14:27:04.344167 I | etcdmain: Git SHA: 3cf2f69b5
2020-07-18 14:27:04.344415 I | etcdmain: Go Version: go1.12.12
2020-07-18 14:27:04.344502 I | etcdmain: Go OS/Arch: linux/amd64
2020-07-18 14:27:04.344577 I | etcdmain: setting maximum number of CPUs to 2, total number of available CPUs is 2
[WARNING] Deprecated '--logger=capnslog' flag is set; use '--logger=zap' flag instead
```

```
root@kubeadm-master:/home/ubuntu# kubectl logs coredns-66bff467f8-6n4g8 -n kube-system
.:53
[INFO] plugin/reload: Running configuration MD5 = 4e235fcc3696966e76816bcd9034ebc7
CoreDNS-1.6.7
linux/amd64, go1.13.6, da7f65b
root@kubeadm-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu#
```

kube-system Process Logs

K21Academy
Learn Cloud from Experts

```
root@kubeadm-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu# kubectl logs kube-proxy-49gft -n kube-system
W0718 14:33:36.398781       1 server_others.go:559] Unknown proxy mode "", assuming iptables proxy
I0718 14:33:36.410772       1 node.go:136] Successfully retrieved node IP: 10.0.0.5
I0718 14:33:36.410937       1 server_others.go:186] Using iptables Proxier.
W0718 14:33:36.410965       1 server_others.go:436] detect-local-mode set to ClusterCIDR, but no cluster CIDR defined
I0718 14:33:36.410972       1 server_others.go:447] detect-local-mode: ClusterCIDR , defaulting to no-op detect-local
I0718 14:33:36.411420       1 server.go:583] Version: v1.18.6
I0718 14:33:36.411959       1 conntrack.go:100] Set sysctl 'net/netfilter/nf_conntrack_max' to 131072
I0718 14:33:36.411995       1 conntrack.go:52] Setting nf_conntrack_max to 131072
I0718 14:33:36.412168       1 conntrack.go:100] Set sysctl 'net/netfilter/nf_conntrack_tcp_timeout_established' to 86400
I0718 14:33:36.412247       1 conntrack.go:100] Set sysctl 'net/netfilter/nf_conntrack_tcp_timeout_close_wait' to 3600
I0718 14:33:36.414454       1 config.go:315] Starting service config controller
I0718 14:33:36.414484       1 shared_informer.go:223] Waiting for caches to sync for service config
I0718 14:33:36.414883       1 config.go:133] Starting endpoints config controller
```

```
root@kubeadm-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu# kubectl logs kube-scheduler-kubeadm-master -n kube-system
I0718 14:27:03.854038       1 registry.go:150] Registering EvenPodsSpread predicate and priority function
I0718 14:27:03.854115       1 registry.go:150] Registering EvenPodsSpread predicate and priority function
I0718 14:27:04.946783       1 serving.go:313] Generated self-signed cert in-memory
W0718 14:27:09.252268       1 authentication.go:349] Unable to get configmap/extension-apiserver-authentication in kube-system.  Usually fixed by 'kubectl create rolebinding
 -n kube-system ROLEBINDING_NAME --role=extension-apiserver-authentication-reader --serviceaccount=YOUR_NS:YOUR_SA'
W0718 14:27:09.252316       1 authentication.go:297] Error looking up in-cluster authentication configuration: configmaps "extension-apiserver-authentication" is forbidden:
User "system:kube-scheduler" cannot get resource "configmaps" in API group "" in the namespace "kube-system"
W0718 14:27:09.252328       1 authentication.go:298] Continuing without authentication configuration. This may treat all requests as anonymous.
W0718 14:27:09.252336       1 authentication.go:299] To require authentication configuration lookup to succeed, set --authentication-tolerate-lookup-failure=false
I0718 14:27:09.276928       1 registry.go:150] Registering EvenPodsSpread predicate and priority function
I0718 14:27:09.276957       1 registry.go:150] Registering EvenPodsSpread predicate and priority function
I0718 14:27:09.281087       1 secure_serving.go:178] Serving securely on 127.0.0.1:10259
I0718 14:27:09.281359       1 configmap_cafile_content.go:202] Starting client-ca::kube-system::extension-apiserver-authentication::client-ca-file
I0718 14:27:09.281377       1 shared_informer.go:223] Waiting for caches to sync for client-ca::kube-system::extension-apiserver-authentication::client-ca-file
I0718 14:27:09.281542       1 tlsconfig.go:240] Starting DynamicServingCertificateController
```

kube-system Process Logs

K21Academy
Learn Cloud from Experts

```
root@kubeadm-master:/home/ubuntu# kubectl logs weave-net-xwfkg weave -n kube-system
INFO: 2020/07/18 14:37:38.578798 Command line options: map[conn-limit:200 datapath:datapath db-prefix:/weavedb/weave-net docker-api: expect-npc:true host-root:/host http-addr:127.0.0.1:6784 ipalloc-init:consensus=2 ipalloc-range:10.32.0.0/12 metrics-addr:0.0.0.0:6782 name:76:ec:70:b2:aa:58 nickname:kubeadm-master no-dns:true port:6783]
INFO: 2020/07/18 14:37:38.582673 weave  2.6.5
INFO: 2020/07/18 14:37:38.840907 Bridge type is bridged_fastdp
INFO: 2020/07/18 14:37:38.840940 Communication between peers is unencrypted.
INFO: 2020/07/18 14:37:38.850825 Our name is 76:ec:70:b2:aa:58(kubeadm-master)
INFO: 2020/07/18 14:37:38.850873 Launch detected - using supplied peer list: [10.0.0.5 10.0.0.6]
INFO: 2020/07/18 14:37:38.865968 Unable to fetch ConfigMap kube-system/weave-net to infer unique cluster ID
INFO: 2020/07/18 14:37:38.866051 Checking for pre-existing addresses on weave bridge
INFO: 2020/07/18 14:37:38.919332 [allocator 76:ec:70:b2:aa:58] No valid persisted data
INFO: 2020/07/18 14:37:38.939904 [allocator 76:ec:70:b2:aa:58] Initialising via deferred consensus
```

```
root@kubeadm-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu# kubectl logs weave-net-xwfkg weave-npc -n kube-system
INFO: 2020/07/18 14:37:42.769249 Starting Weaveworks NPC 2.6.5; node name "kubeadm-master"
INFO: 2020/07/18 14:37:42.769725 Serving /metrics on :6781
Sat Jul 18 14:37:42 2020 <5> ulogd.c:408 registering plugin `NFLOG'
Sat Jul 18 14:37:42 2020 <5> ulogd.c:408 registering plugin `BASE'
Sat Jul 18 14:37:42 2020 <5> ulogd.c:408 registering plugin `PCAP'
Sat Jul 18 14:37:42 2020 <5> ulogd.c:981 building new pluginstance stack: 'log1:NFLOG,base1:BASE,pcap1:PCAP'
WARNING: scheduler configuration failed: Function not implemented
DEBU: 2020/07/18 14:37:42.822888 Got list of ipsets: []
```

K8s App and Network Troubleshooting

- **Network issue**
  - Service issues
  - Service <-> Pod mapping issue
  - Port mapping issue
- **Application issues**
  - Image pull error
  - App error

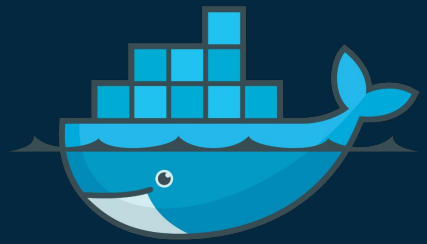
Application Logs

K21Academy
Learn Cloud from Experts

```
root@kubeadm-master:/home/ubuntu/Kubernetes-basics#
root@kubeadm-master:/home/ubuntu/Kubernetes-basics# kubectl logs demo-pod
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```

Questions

K21Academy
Learn Cloud from Experts



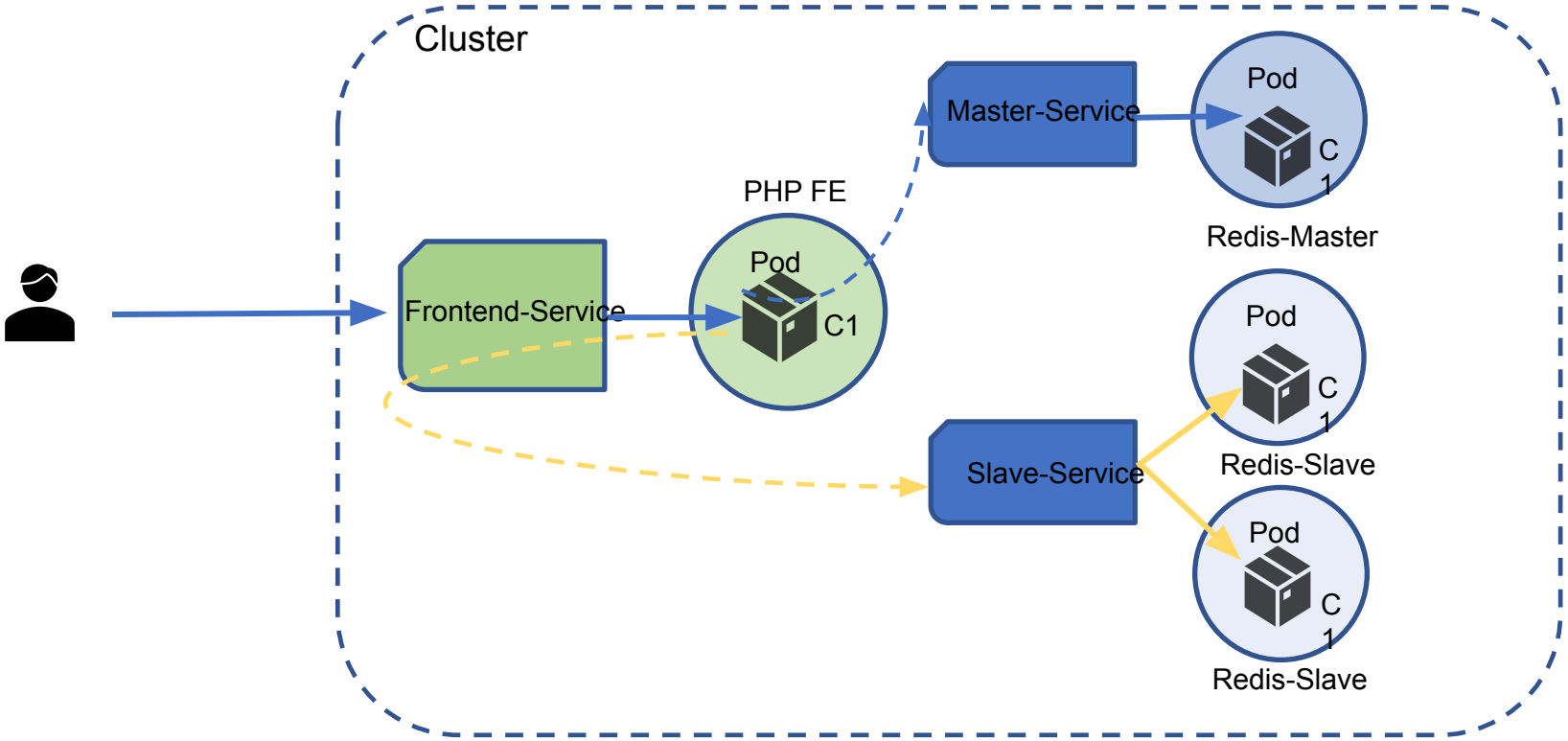
docker

+

kubernetes

Working with Application Stack

Guestbook Application



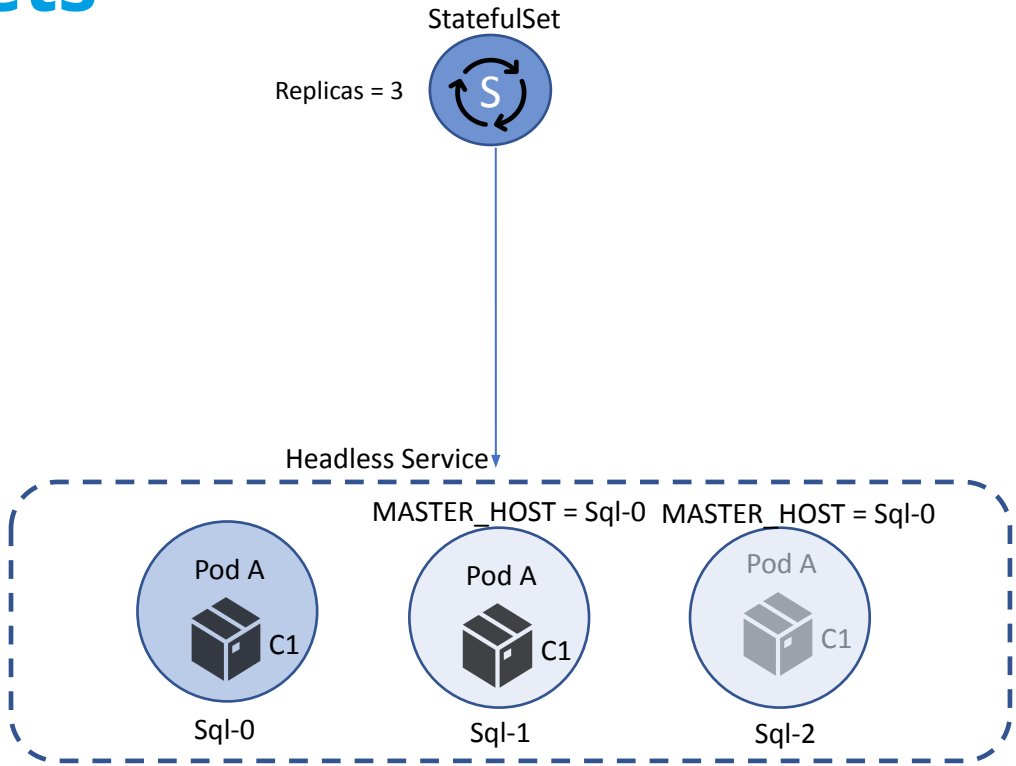
K21Academy
Learn Cloud from Experts

Deploy Stateful Application

Database App as Deployment

- Provides HA
- Self Healing
- ClusterIP service will expose it within the cluster
- But all pods come up at once
- Pod can’t be marked as master or slave
- Service by default does load balancing across replicas
- Can’t mark a replicas as Reap replica or Write replica

StatefulSets



StatefulSets

K21Academy
Learn Cloud from Experts

- Used for Stateful application and Distributed systems
- For a StatefulSet with N replicas, Pods are created sequentially, in order from {0..N-1}
- Pods in a StatefulSet have a unique ordinal index and a stable network identity
- The RollingUpdate strategy updates all Pods in a StatefulSet, in reverse ordinal order
- The controller deletes one Pod at a time, in reverse order with respect to its ordinal index

Headless Service

K21Academy
Learn Cloud from Experts

- Don't need load-balancing and a single Service IP
- Explicitly specifying "None" for the cluster IP
- Allows us to interact directly with the Pods instead of a proxy
- A headless service is a service with a service IP but instead of load-balancing it will return the IPs of our associated Pods

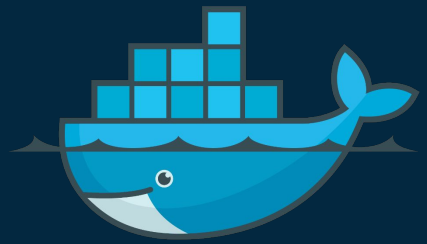
```
kind: Service
apiVersion: v1
metadata:
  name: elasticsearch
  namespace: kube-logging
  labels:
    app: elasticsearch
spec:
  selector:
    app: elasticsearch
  clusterIP: None
  ports:
    - port: 9200
      name: rest
    - port: 9300
      name: inter-node
~
~
~
```

Init-Containers

K21Academy
Learn Cloud from Experts

- Specialized containers that run before app containers in a Pod
- Init containers can contain utilities or setup scripts not present in an app image
- Init containers are exactly like regular containers, except:
- Init containers always run to completion.
- Each Init container must complete successfully before the next one starts
- If a Pod's Init container fails, Kubernetes repeatedly restarts the Pod until the init container succeeds
- If you specify multiple Init containers for a Pod, Kubelet runs each init container sequentially
- Kubelet initializes the application containers for the Pod after all Init container are completed

Questions



docker

+

kubernetes

K21Academy
Learn Cloud from Experts

Deploy Daemon Process

DaemonSet

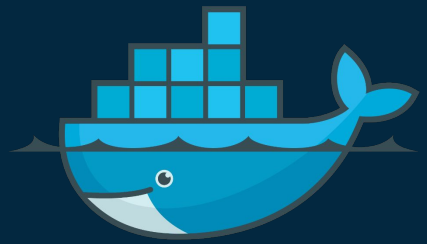
K21Academy
Learn Cloud from Experts

- A daemonset ensures that an instance of a specific pod is running on all nodes in a cluster
- It creates pods on every cluster node and these pods are garbage collected when nodes are removed from the cluster
- We can use daemonsets for running daemons and other tools that need to run on all nodes of a cluster
- Cluster bootstrapping is another use case. The DaemonSet controller can make Pods even when the scheduler has not been started
- We can perform rolling update on DaemonSet

Node specific Daemons

- If we plan to have different logging, monitoring, or storage solutions on different nodes of the cluster
- We want to deploy the daemons only to a specific set of nodes instead of all Node selector can be used to specify a subset of nodes for the daemon set
- DaemonSet controller will create Pods on nodes which match that node affinity

Questions



docker

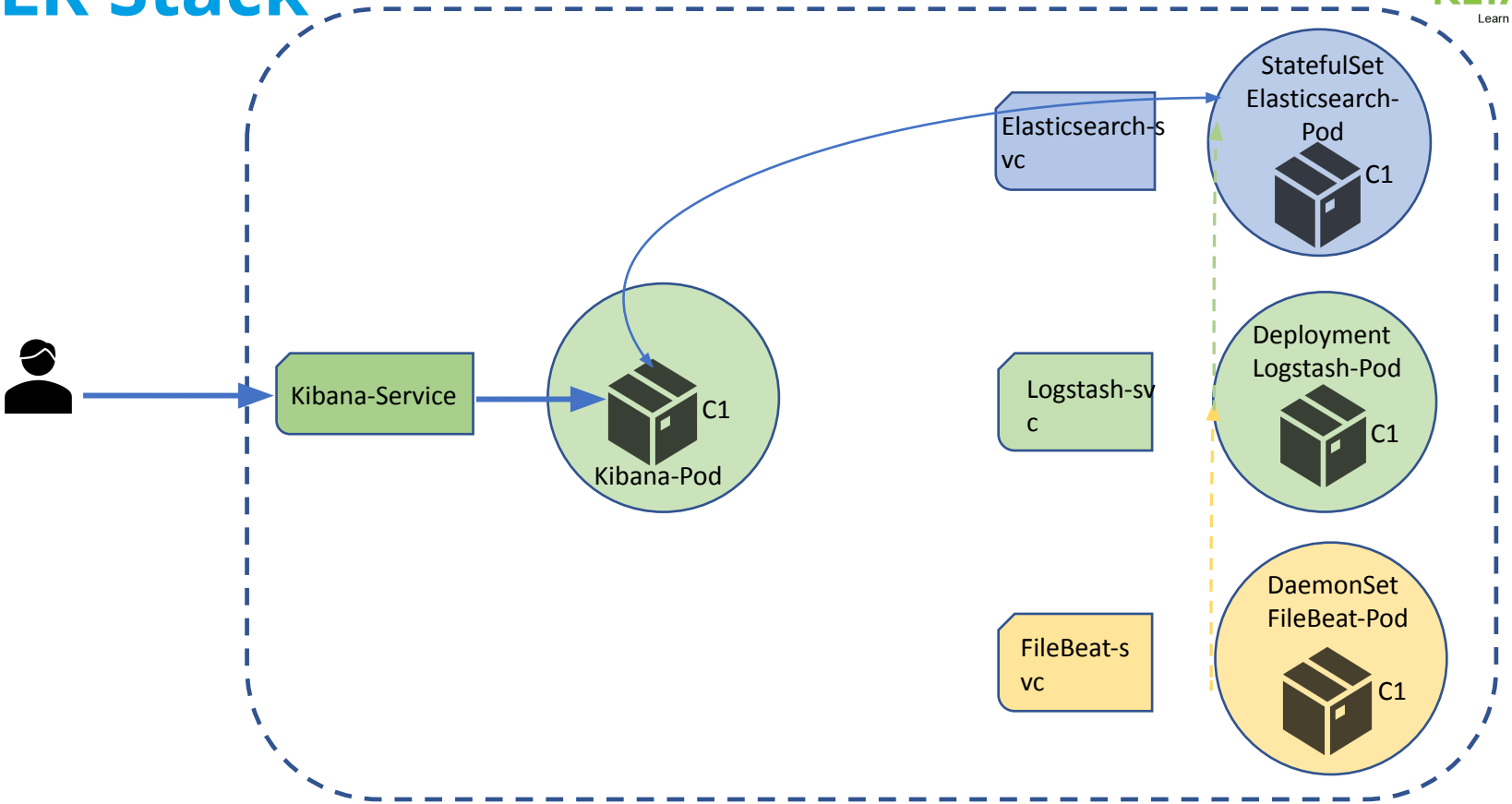
+

kubernetes

K21Academy
Learn Cloud from Experts

Working with Logging Stack

ELK Stack



Aggregating Logs

- Log aggregation system uses a push mechanism to collect the data
- The ELK stack is a popular log aggregation and visualization solution that is maintained by elasticsearch
- Stack consists of:
  - Kibana
  - Elasticsearch
  - Logstash
  - Filebeat

Elasticsearch-StatefulSet

- A real-time, distributed, and scalable search engine
- Allows for full-text and structured search, as well as analytics
- Commonly used to index and search through large volumes of log data
- Elasticsearch simplifies data ingest, visualization, and reporting
- It helps in fast and Incisive search against large volumes of data
- Real-time data and real-time analytics
- Elasticsearch comes with a wide set of features like scalability, high-availability and multi-tenant

Logstash-Deployment

- Logstash is a popular open-source data collector that we'll set up on our Kubernetes nodes to tail container log files
- Logstash to collect, transform, and ship log data to the Elasticsearch backend
- We can filter and transform the log data, and deliver it to the Elasticsearch cluster, where it will be indexed and stored

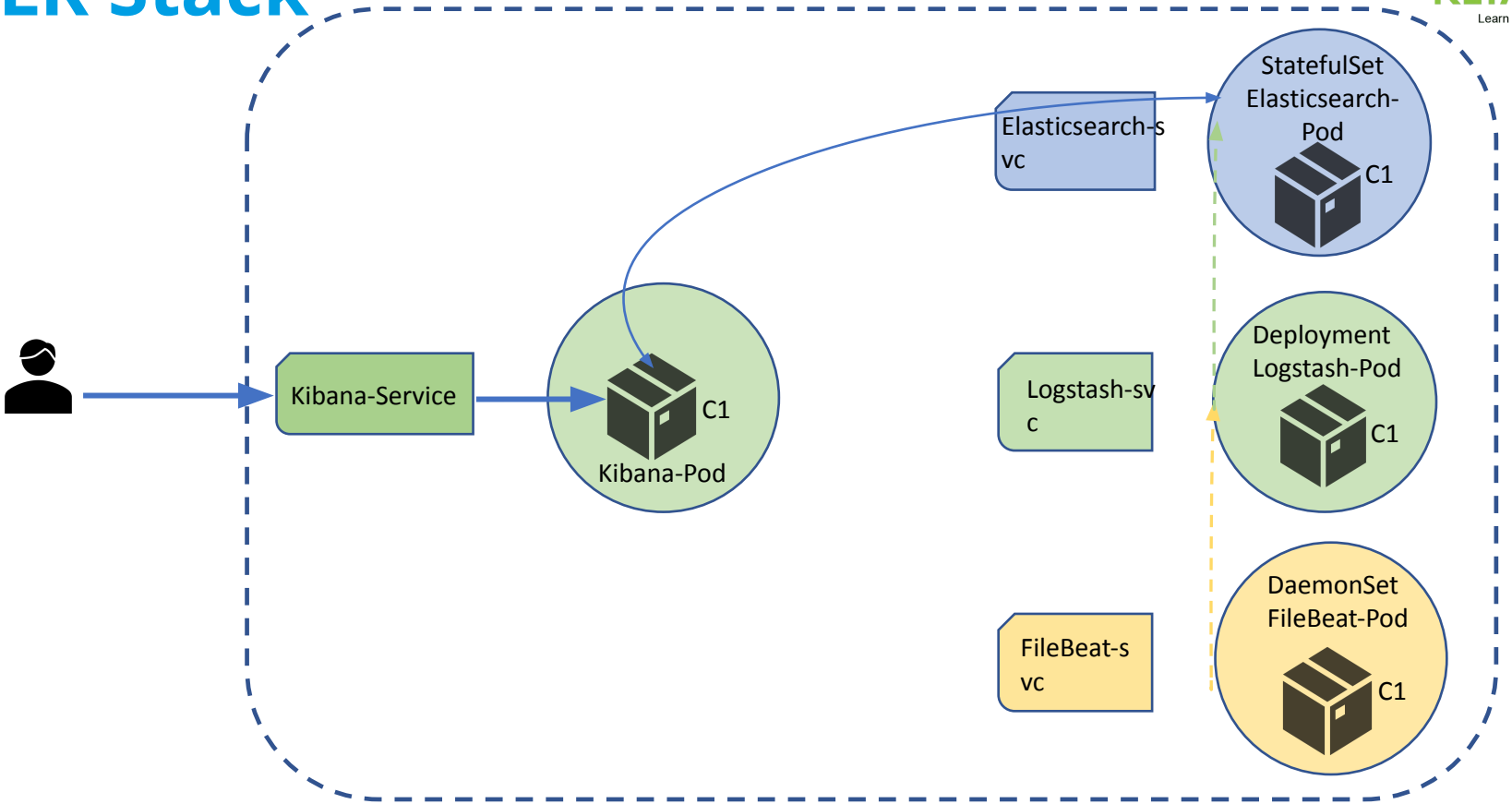
Filebeat-DaemonSet

- Filebeat is the agent to ship logs to Logstash
- It runs like a DaemonSet on all nodes in the cluster to collect logs
- Filebeat is a lightweight shipper for forwarding and centralizing log data

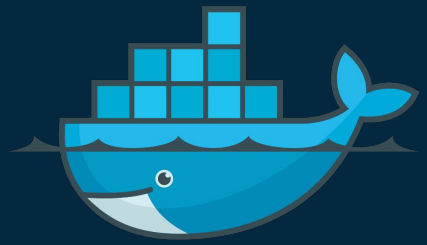
Kibana-Deployment

- Kibana is a data visualization and management tool for Elasticsearch that provides real-time histograms, line graphs, pie charts, and maps
- Kibana allows you to explore your Elasticsearch log data through a web interface
- We can build dashboards and queries to quickly answer questions and gain insight into your Kubernetes applications

ELK Stack



Questions



docker

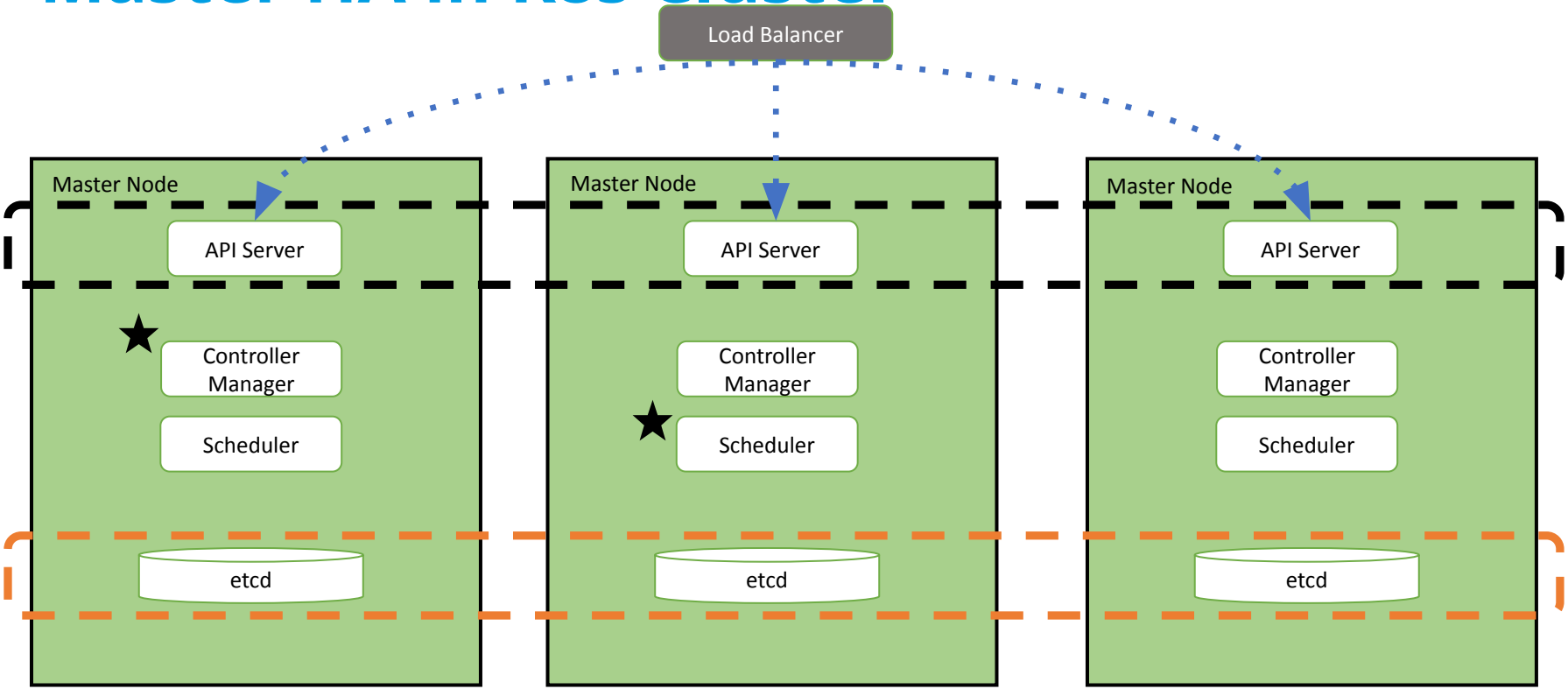
+

kubernetes

K21Academy
Learn Cloud from Experts

Highly Available Cluster & Upgrade

Master HA in K8s Cluster



Leader Election

- Instance who firstly creates a service endpoint is the leader of this service at very beginning
- This instance then adds (control plane.alpha.kubernetes.io/leader) annotation to expose leadership information to other followers or application in the cluster
- The leader instance shall constantly renew its lease time to indicate its existence

```
root@kubeadm-master:/# kubectl describe ep -n kube-system kube-controller-manager
Name:         kube-controller-manager
Namespace:    kube-system
Labels:       <none>
Annotations:  control-plane.alpha.kubernetes.io/leader:
                {"holderIdentity":"kubeadm-master_0b72f545-9b54-4ded-a217-cfc10e920cb7","leaseDurationSeconds":15,"acquireTime":"2020-06-10T00:00:50Z","re...
Subsets:
Events:  <none>
root@kubeadm-master:/#
```

Leader Re-Election

- Followers will constantly check the existence of service endpoint
- If it is created already by the leader then the followers will do an additional lease (renewTime) renew checking against the current time
- If lease time (renewTime) is older than **Now** which means leader failed to update its lease duration, hence suggests Leader is crashed
- In that case, a new leader election process is started until a follower successfully claims leadership by updating endpoint with its own Identity and lease duration

Master High Availability

- Having multiple master nodes ensures that services remain available should master node(s) fail
- In order to facilitate availability of master services, they should be deployed with odd numbers (e.g. 3,5,7,9 etc.)
- We want quorum (master node majority) to be maintained should one or more masters fail
- Provides highly available master services, ensuring that the loss of up to (n – 1)/2 master nodes will not affect cluster operations

Backing Up and Restoring Cluster

K21Academy
Learn Cloud from Experts

- All Kubernetes objects are stored on etcd
- Periodically backing up the etcd cluster data is important to recover Kubernetes clusters under disaster scenarios
- The snapshot file contains all the Kubernetes states and critical information
- etcd supports built-in snapshot, so backing up an etcd cluster is easy
- Restore from the snapshot

```
root@kubeadm-master:/# sudo ETCDCTL_API=3 etcdctl snapshot save snapshot.db --cacert /etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/server.crt
 --key /etc/kubernetes/pki/etcd/server.key
Snapshot saved at snapshot.db
root@kubeadm-master:/#
root@kubeadm-master:/#
root@kubeadm-master:/#
```

Cluster Certificates

- First need to have CA Authority to create Certs
- Client certificates
  - Admin Client certificate
  - Kubelet Client certificate
  - Controller Manager Client
  - Kube Proxy Client
  - Kube Scheduler Client
- API Server Certificates
- Service Account Key pairs
- Move the files onto the appropriate servers

Upgrading kubeadm Maters

K21Academy
Learn Cloud from Experts

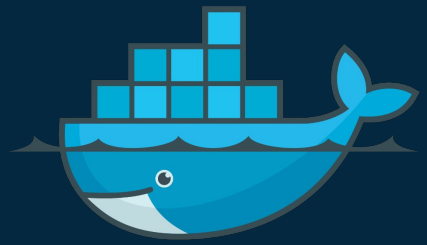
- Determine which version to upgrade to
- Upgrading control plane nodes
  - On your first control plane node, upgrade kubeadm:
  - Drain the control plane node:
  - On the control plane node, run: **sudo kubeadm upgrade plan**
  - Choose a version to upgrade to and run : **sudo kubeadm upgrade apply v1.18.x**
  - kubectl uncordon <cp-node-name>
- Upgrade additional control plane nodes
  - sudo kubeadm upgrade node

Upgrading kubeadm Worker

K21Academy
Learn Cloud from Experts

- Upgrade kubeadm on all worker nodes
- Drain the node: kubectl drain <node-to-drain> --ignore-daemonsets
- Execute the upgrade cmd: sudo kubeadm upgrade node
- Upgrade kubelet and kubectl
- Restart the kubelet
- Uncordon the node
- Verify the status of the cluster

Questions



docker

+

kubernetes

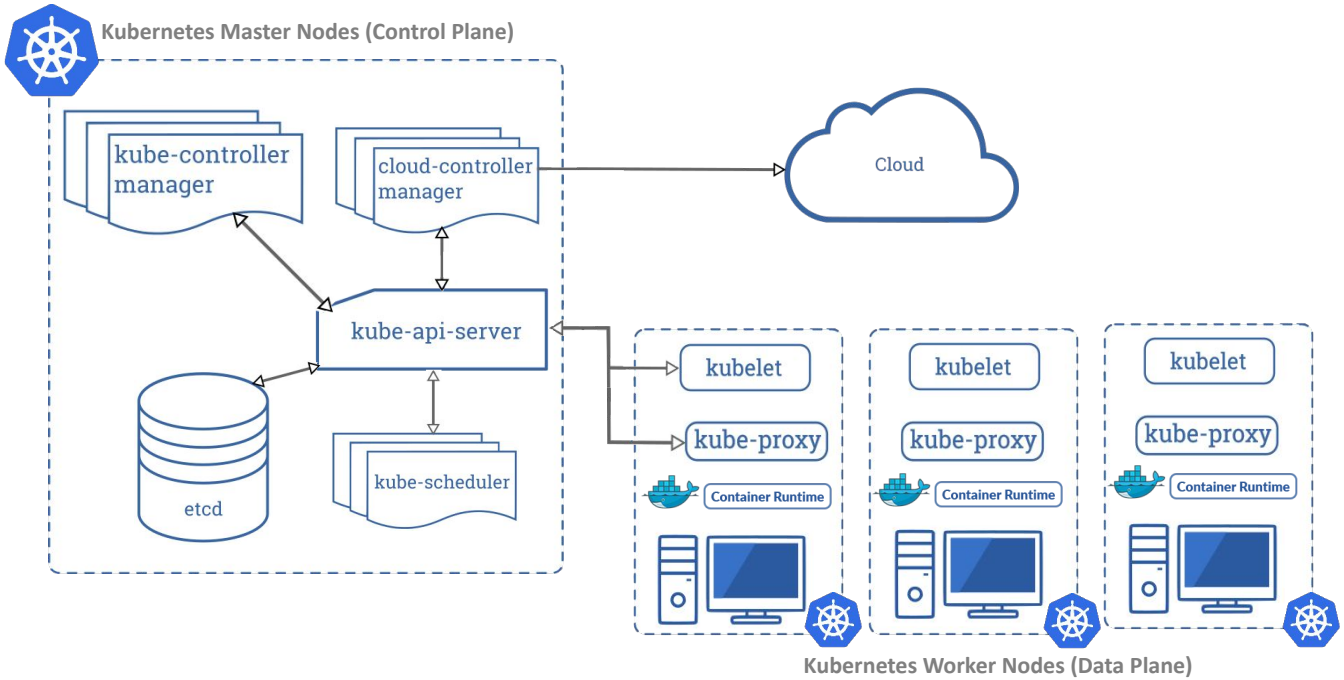
K21Academy
Learn Cloud from Experts

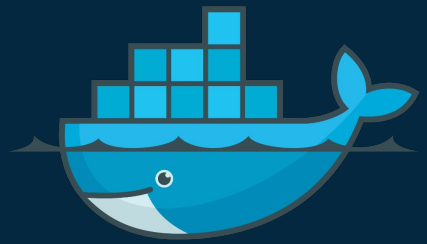
Managed Kubernetes on Cloud

Kubernetes Adoption in Cloud

- AWS - EKS
- Microsoft Azure - AKS
- Google Cloud Platform - GKE
- Oracle Cloud - OKE
- Digital Ocean - DOKS

Kubernetes Architecture





docker

+

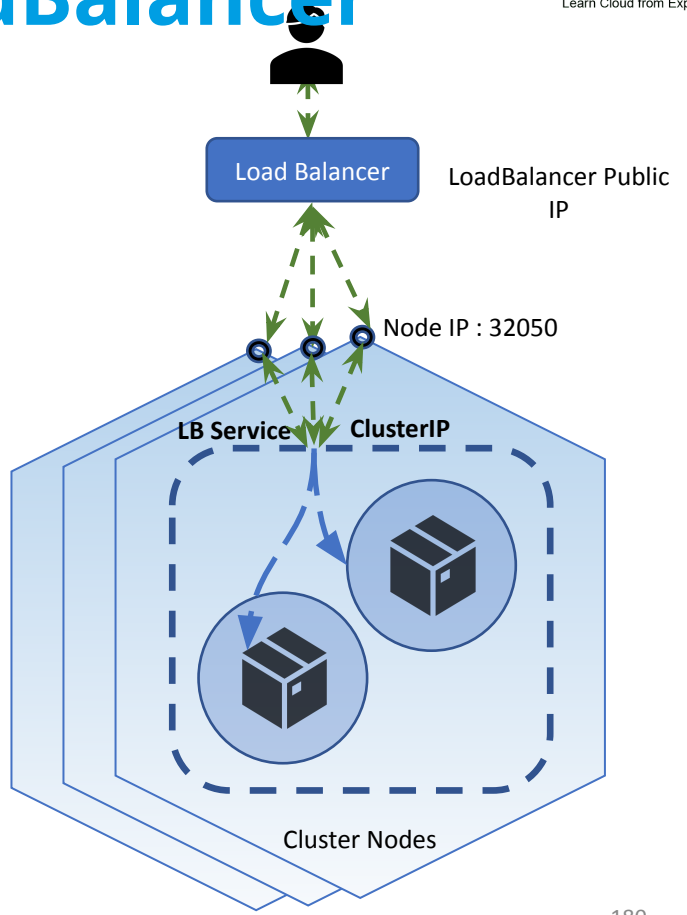
kubernetes

K21Academy
Learn Cloud from Experts

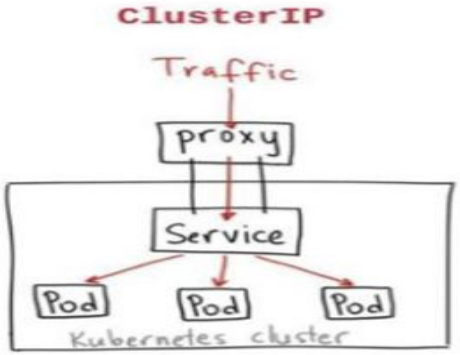
Advanced Networking

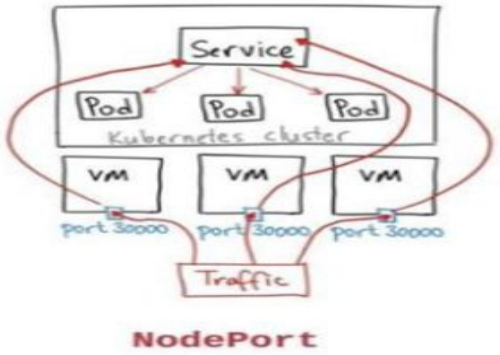
Kubernetes Service - LoadBalancer

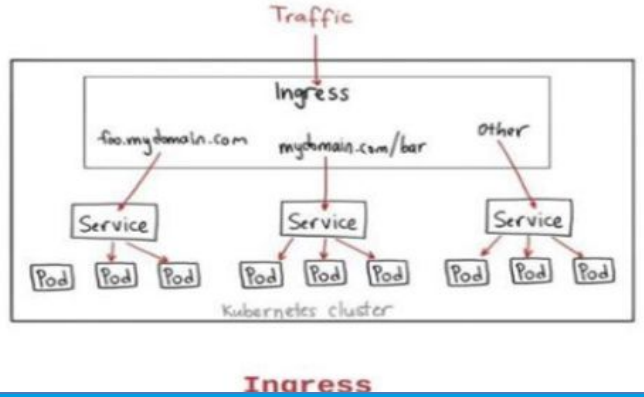
- Exposes the Pod to outside the cluster
- Assigns a Private IP Address in Cluster IP range
- Assigns a port on the worker nodes to expose it outside cluster
- Create a Load Balancer in Cloud

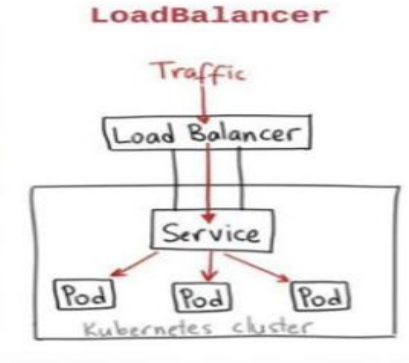


K8s Service Types

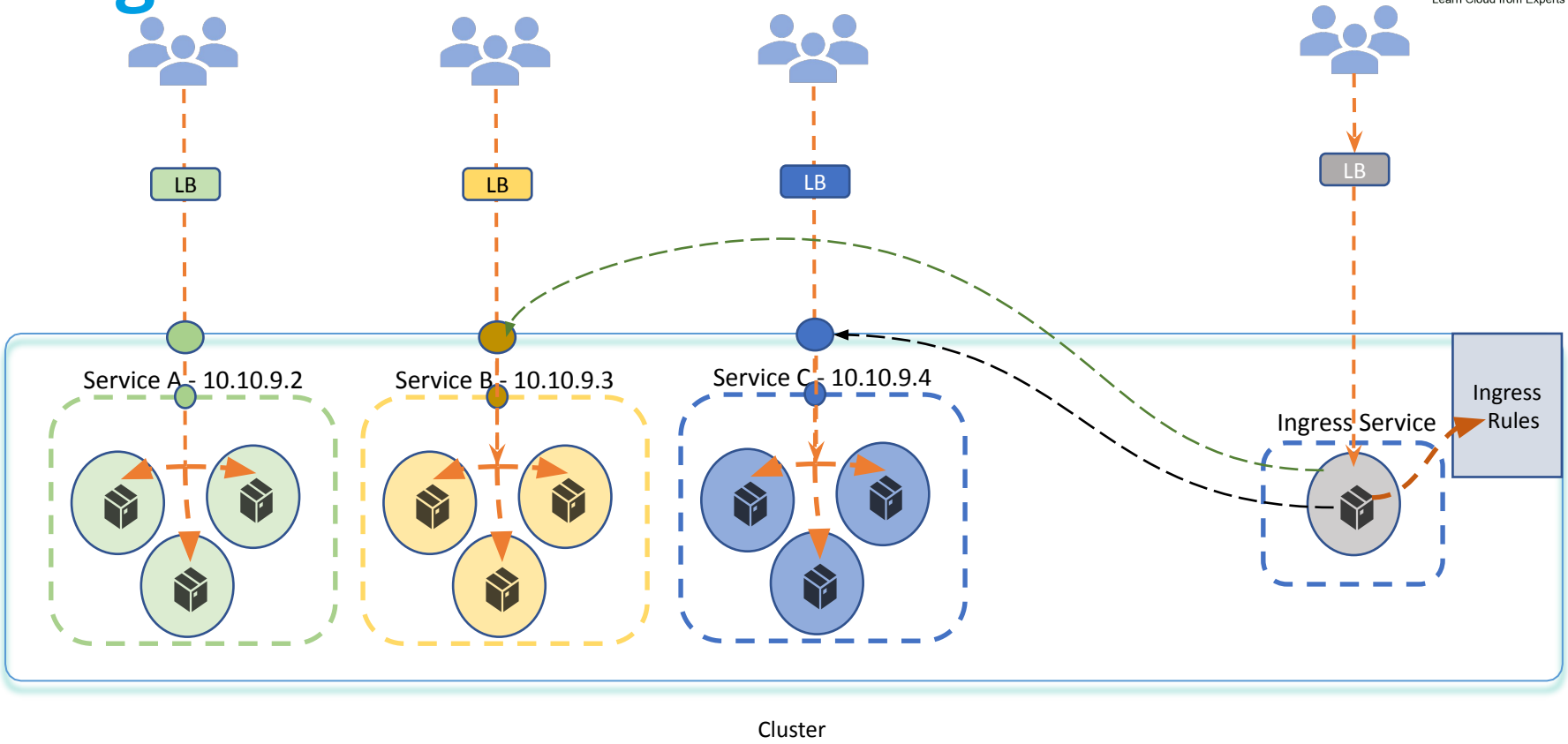








Ingress Controller



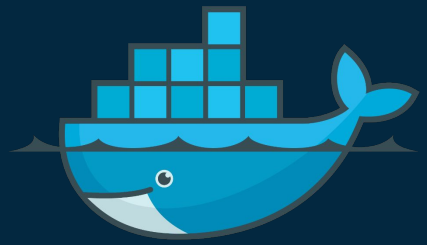
Ingress Controller

- Intelligent Load Balancer
- Layer 7 Load Balancer
- Nginx or Nginx +
- It doesn’t run as part of kube-controller-manager
- With Ingress, users don't connect directly to a Service
- Users reach the Ingress endpoint, and, from there, the request is forwarded to the respective Service

Ingress

- An API object that manages external access to the services in a cluster, typically HTTP
- Ingress may provide load balancing, SSL termination and name-based virtual hosting
- Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster
- Traffic routing is controlled by rules defined on the Ingress resource
- For it to work we need to deploy an Ingress controller such as ingress-nginx

Questions



docker

+

kubernetes

K21Academy
Learn Cloud from Experts

Helm
&
Helm Charts

Helm Terminology

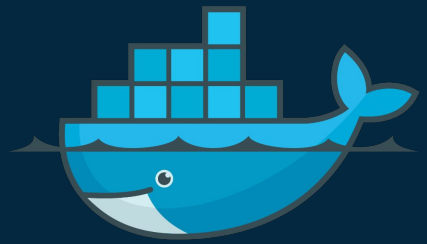
- A Chart is a Helm package. It contains all of the resource definitions necessary to run an application, tool, or service inside of a Kubernetes cluster
- A Repository is the place where charts can be collected and shared
- A Release is an instance of a chart running in a Kubernetes cluster

Helm Charts

- Package manager for Kubernetes
- Helm is the best way to find, share, and use software built for Kubernetes
- Helm helps us manage Kubernetes applications — Helm Charts
- Helm Chart is collection of files that describe a related set of K8s resources
- Helps us define, install, and upgrade even the most complex Kubernetes application
- Charts are easy to create, version control, share, and publish

Helm Commands

- helm search
- helm repo add
- helm install
- helm list
- helm status
- helm show values
- helm install
- helm upgrade
- helm get values
- helm rollback
- helm uninstall



+

K21Academy
Learn Cloud from Experts

Dynamic Storage Provisioning

Yaml Definitions

K21Academy
Learn Cloud from Experts

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

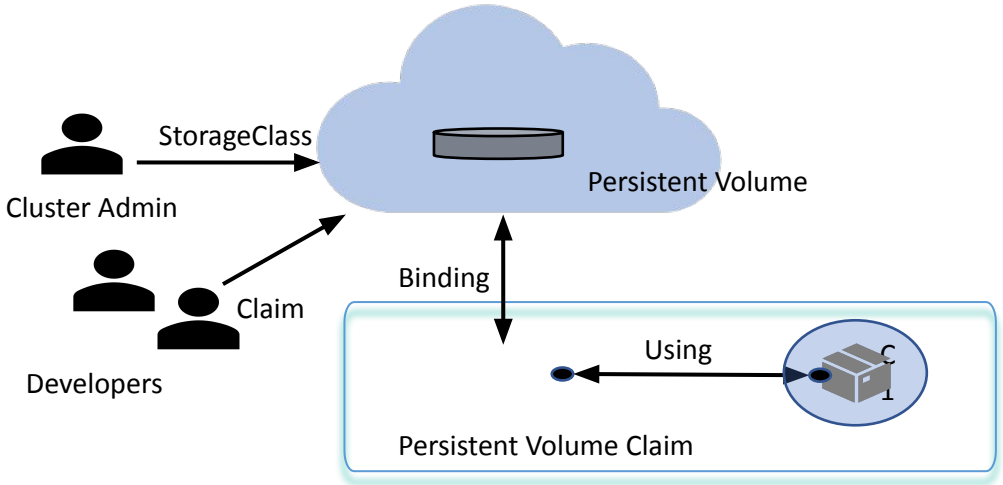
Storage Class

- A StorageClass provides a way for administrators to describe the "classes" of storage they offer
- Different classes might map to quality-of-service levels, or to backup policies
- Each StorageClass

```
mamta@Azure:~/Kubernetes$ kubectl describe sc managed-premium
Name:            managed-premium
IsDefaultClass:  No
Annotations:     kubectl.kubernetes.io/last-applied-configuration={"allowVolumeExpansion":true,"apiV
ersion":"storage.k8s.io/v1beta1","kind":"StorageClass","metadata":{"annotations":{},"labels":{"kuber
netes.io/cluster-service":"true"},"name":"managed-premium"},"parameters":{"cachingmode":"ReadOnly","
kind":"Managed","storageaccounttype":"Premium_LRS"},"provisioner":"kubernetes.io/azure-disk"}

Provisioner:           kubernetes.io/azure-disk
Parameters:            cachingmode=ReadOnly,kind=Managed,storageaccounttype=Premium_LRS
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>
mamta@Azure:~/Kubernetes$
```

Dynamic Volume Provisioning

- Dynamic volume provisioning allows storage volumes to be created on-demand
- Dynamic volume provisioning is based on the API object StorageClass
- To enable dynamic provisioning, a cluster administrator pre-creates one or more StorageClass objects for users
- StorageClass objects define which provisioner should be used and what parameters should be passed to that provisioner when dynamic provisioning is invoked



Questions

Summary: Module

K21Academy
Learn Cloud from Experts

- Introduction to Kubernetes
  - Kubernetes Architecture
  - Kubernetes Object
  - Kubernetes Pod
  - Kubernetes Networking
- Kubernetes Deep Dive
  - Highly Available and Scalable Application
  - Autoscaling with HPA
  - Persistent Storage
  - Advanced Scheduling

Summary: Module

K21Academy
Learn Cloud from Experts

- Kubernetes Security
  - RBAC - Role Based Access Control
  - Role and Role Binding
  - Network Security
  - Security Context
  - ConfigMap
- Cluster Resource Limit
- MultiContainer Pattern - Sidecar
- Maintenance & Troubleshooting
- Working with Application Stack

Agenda: Module

- Deploy Stateful Application
- Deploy Daemon Process
- Working with Logging Stack
- Kubernetes Next Level
  - Managed Kubernetes on Cloud
  - Advance Scheduling - Taint & Tolerations
  - Advanced Networking
  - Helm and Helm Chart
- Dynamic Storage Provisioning

Find Us

https://www.facebook.com/K21Academy

http://twitter.com/k21Academy

https://www.linkedin.com/company/k21academy

https://www.youtube.com/dockerkubernetes

https://www.instagram.com/k21academy/