

Docker & Certified Kubernetes Administrator (CKA)

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

16 Module | 60+ Lessons | 33 Hands-On Labs | Exam Preparation | On-Job Support







Atul Kumar





Cloud Expert

- 20+ Years working on Oracle as Certified Cloud Architect
- Help Customers Migrating to Cloud & Integrations
- Oracle ACE since 2006
- Helped thousands of individuals to learn Cloud













Mamta: Docker Kubernetes Expert





- > 16+ years of IT experience, seasoned Cloud Solution Architect with 7+Years of hands-on experience
- Automating, and Optimizing mission-critical Deployments in several Cloud Platforms
- Leveraging Configuration Management, CI/CD, and DevOps Processes.



Getting Help



support@k21academy.com



Agenda: Module



Introduction to Kubernetes

- Kubernetes Architecture
- Kubernetes Object
- Kubernetes Pod
- Kubernetes Networking

Kubernetes Deep Dive

- Kubernetes Service
- Highly Available and Scalable Application
- Autoscaling with HPA
- Persistent Storage
- Advanced Scheduling

Agenda: Module



- Maintenance & Troubleshooting
- Kubernetes Security
 - RBAC Role Based Access Control
 - Role and RoleBinding
 - Network Security
- ➤ Working with Application Stack
- Deploy Stateful Application
- Kubernetes Cluster HA
 - Backing Up and Restoring Cluster
 - Master HA in K8s Cluster

Agenda: Module



- Deploy Daemon Process
- Working with Logging Stack
- Kubernetes Next Level
 - Adv Scheduling Taint & Tolerations
 - Probing Parameters
 - Kubernetes Service LoadBalancer
 - Managed Kubernetes on Cloud
 - Advanced Networking
 - > Helm Chart
- Dynamic Storage Provisioning





Introduction to Kubernetes

What Is Kubernetes?



- An open-source containerorchestration 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

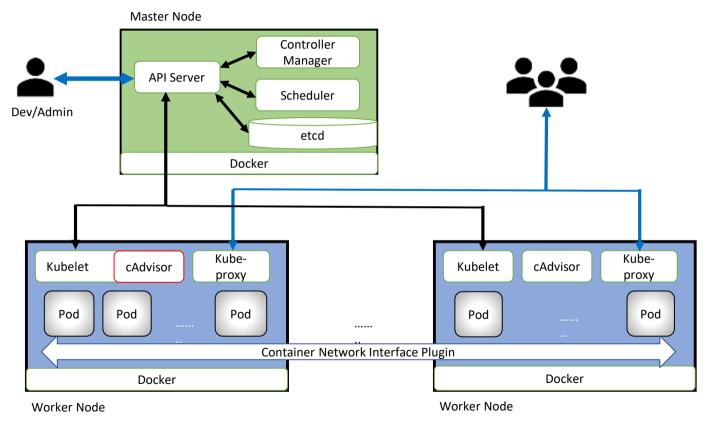




Kubernetes Architecture

Kubernetes Architecture

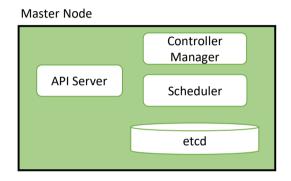




Master Node Architecture



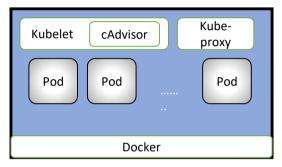
- ➤ **API Server**: Configures and validates data for api objects like pods, services. Its a front-end of control plane
- > **Scheduler**: It decides where in the cluster the workloads are to be run
- > etcd: Stores all cluster-related data
- Controller: Daemon that embeds core control loops that regulates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

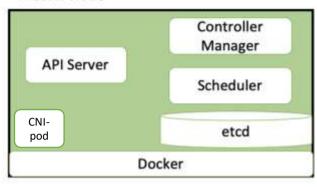


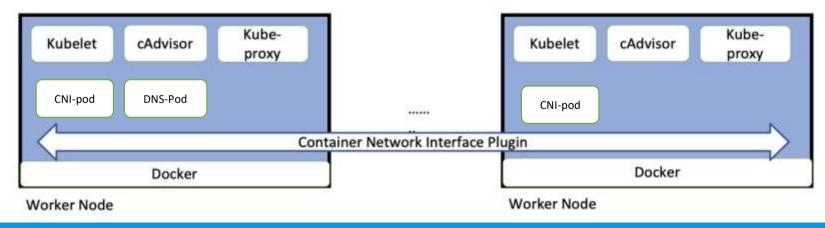
- > Everything is an API call served by the *Kubernetes API server* (kube-apiserver)
- The API server is a gateway to an etcd datastore that maintains the desired state of your application cluster
- To update the state of a Kubernetes cluster, we make API calls to the API server describing our desired 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.

Master Node















Questions



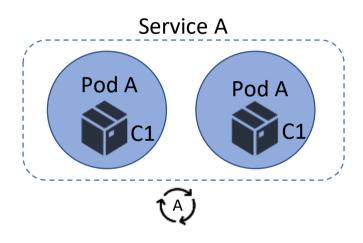


Kubernetes Building Blocks

Kubernetes Object



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



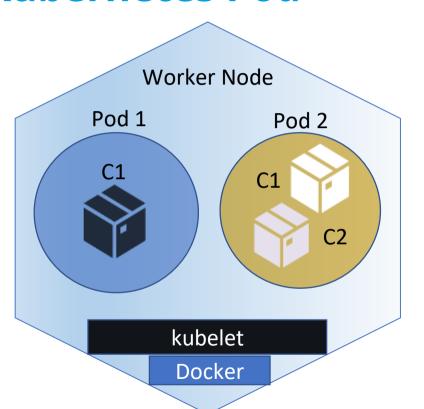
Namespace



- 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



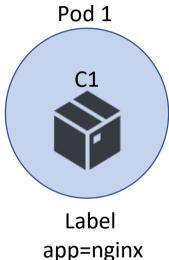


- Pod is the atomic unit on the K8s platform
- Pod can be a group of one or more application containers
- Pod has a unique Ip Address
- Containers in a pod share the IP address and port namespace

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

```
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

K21Academy

apiVersion: v1 kind: Pod

metadata:

name: nginx

labels:

app: nginx

env: dev

spec:

containers:

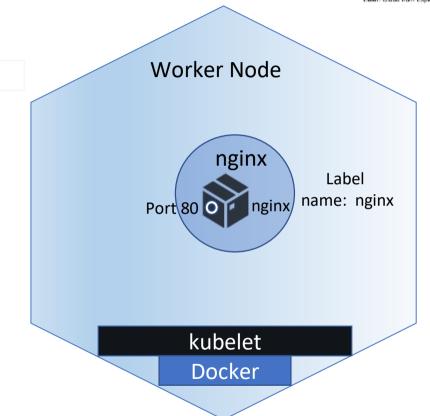
- name: nginx

image: nginx

ports:

- containerPort: 80

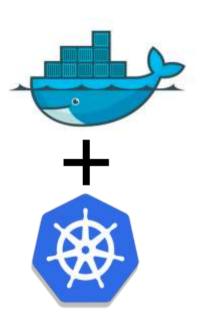
kind: Pod



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



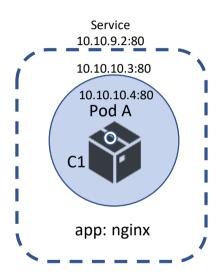


Kubernetes Networking

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



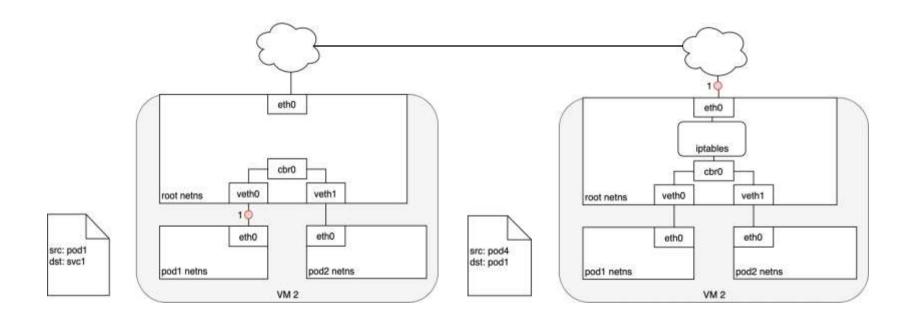
Kubernetes Service



- 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

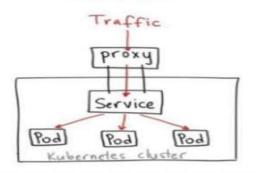


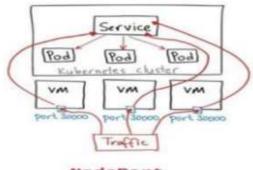


K8s Service Types

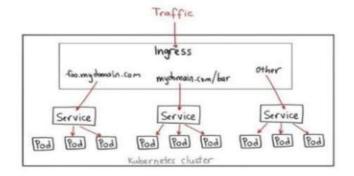




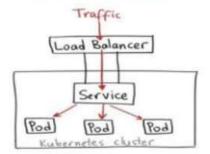




NodePort



LoadBalancer

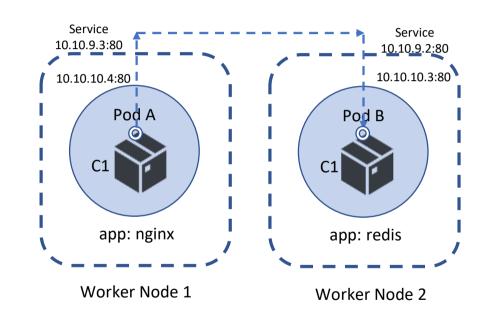


Ingress

Kubernetes Service - ClusterIP



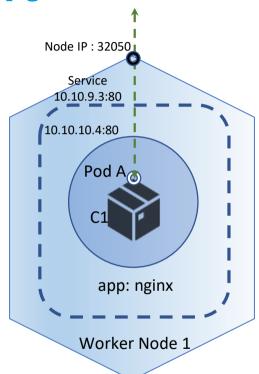
- Exposes the Pod to only within the cluster
- Assigns a Private IP Address in Cluster IP range
- Can talk to applications only within the cluster
- Doesn't have external connectivity



Kubernetes Service - NodePort

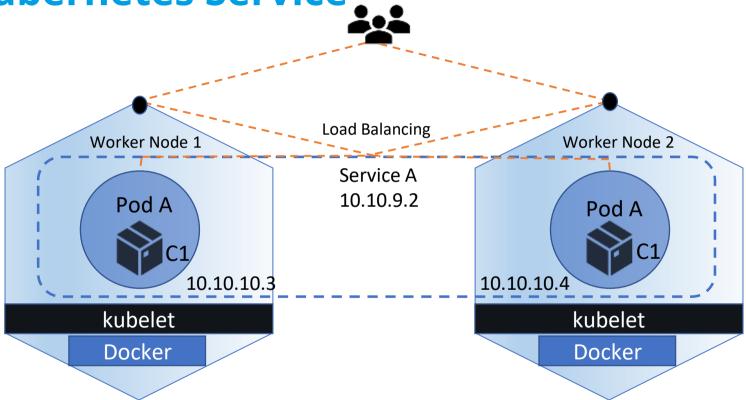


- Exposes the Pod to outside the cluster
- Assigns a Private IP Address in Cluster IP range
- Assigns a port on the worker nodes to expose it outside cluster
- Provides external connectivity



Kubernetes Service





Kubernetes DNS Service



- 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





Deploy Highly Available Application

Kind: Deployment

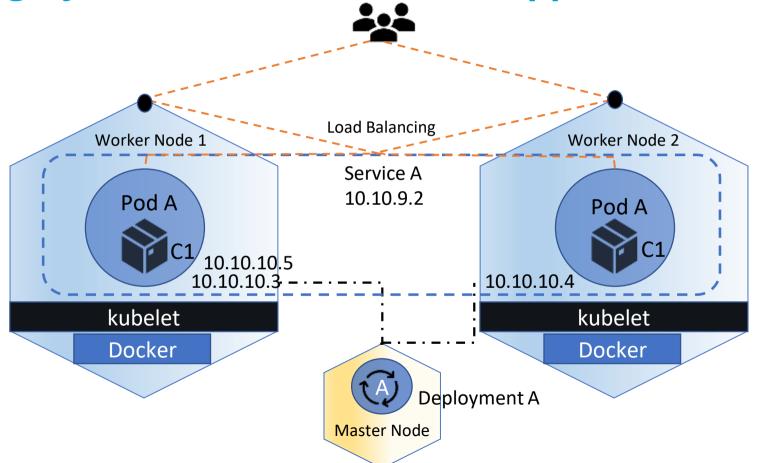


- Deployment Controller defines the state of Deployment Objects, like Pods and ReplicaSets
- Deployments are used to create and deploy, scale, monitor and roll back
- Manage the state of Pods and ReplicaSets on the system
- Make application highly available, scalable and self-healing

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

Highly Available and Scalable Application





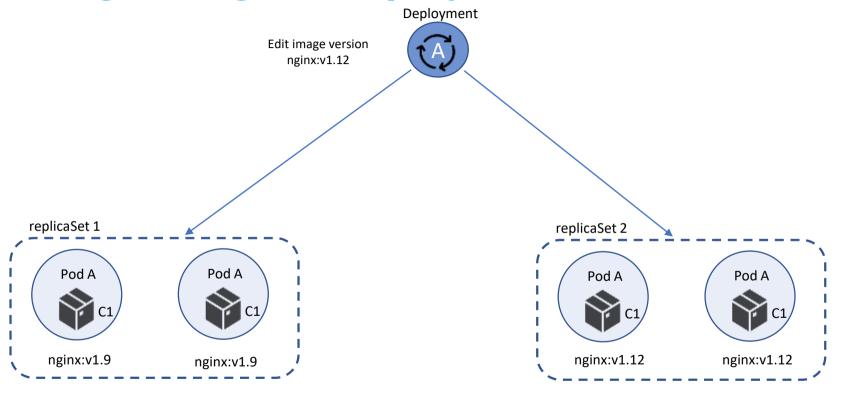
Responsibilities - Deployment



- > 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











Questions





Application Autoscaling

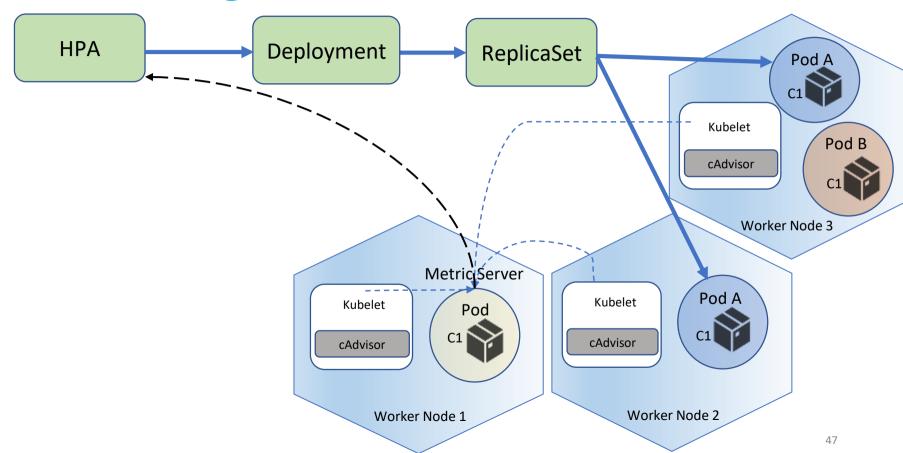
Horizontal Pod Autoscaler



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

Autoscaling with HPA











Questions





Persistent Storage

Persistent Storage

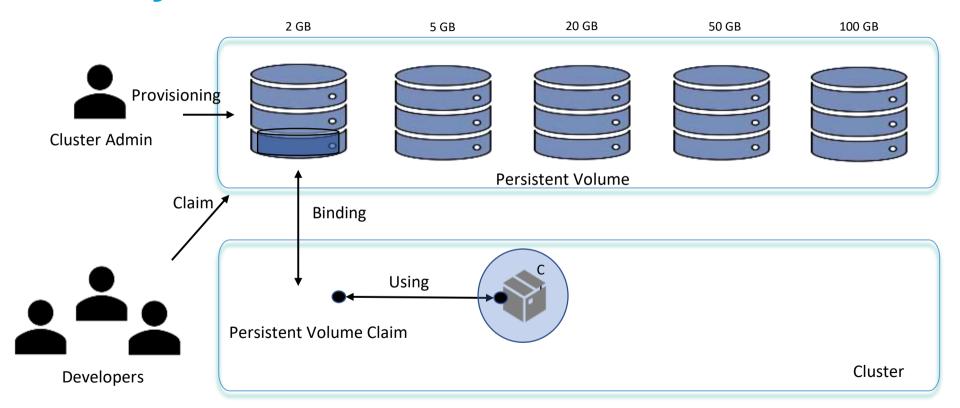


- PersistentVolumes (PV) are volume plugins like Volumes
- PVs lifecycle is managed by Kubernetes
- PVs are cluster resources that exist independently of Pods
- PVs can be provisioned by :
 - Static provisioning by an administrator
 - Dynamically provisioned using Storage Classes

- > PersistentVolumeClaim (PVC) is a request for storage by a user
- Claims can request specific size and access modes

Lifecycle of a Volume and Claim





Volume and Access Modes



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

Volume Types



> 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

> 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

Volume Types



hostPath

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

configMap and secret

useful for inserting configuration and secret data into your containers

Yaml Definitions



```
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
```

```
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]}
```

```
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
```

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



- 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





Advanced Scheduling

kube-scheduler



- 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



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

Node Selector

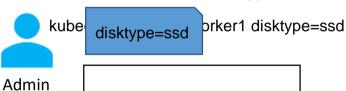
- > The simplest recommended form of node selection constraint.
- nodeSelector is a field of PodSpec
- It specifies a map of key-value pairs
- For the pod to be eligible to run on a node, the node must have each of the indicated key-value pairs as labels

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

Node Selector







Worker 1

```
kind: Deployment
                 metadata:
                   name: nginx-deployment
                   labels:
                     app: nginx
                 spec:
Developers
                    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
      Worker 2
```

apiVersion: apps/v1

Advanced Scheduling



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

Node Affinity

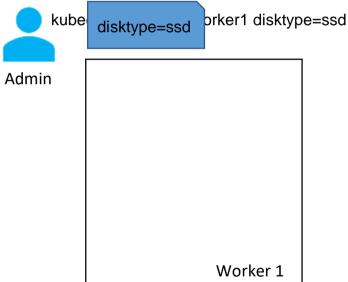


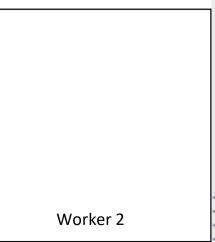
- Similar to nodeSelector
- Allows to constrain based on labels on the node
- > Types of node affinity:
 - preferredDuringSchedulingIgnoredDuringExecution
 - > requiredDuringSchedulingIgnoredDuringExecution

Node Affinity - Reqd-In







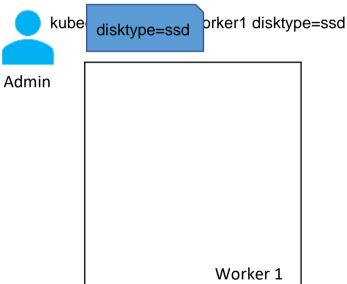


```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 labels:
   app: nginx
spec:
 replicas: 2
  selector:
   matchlabels:
      app: nginx
 template:
   metadata:
      labels:
       app: nginx
   spec:
      containers:
      - name: nginx
        image: nginx:1.12
       ports:
        - containerPort: 88
      affinity:
       nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: disktype
                operator: In
                values:
                - ssd
```

Node Affinity - Pref-In







Developers Worker 2



Node Affinity/Anti-Affinity



```
apiVersion: apps/vl
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
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 labels:
   app: nginx
 replicas: 2
 selector:
   matchLabels:
     app: nginx
 template:
   metadata:
     labels:
       app: nginx
    spect:
     containers:
     - name: nginx
       image: nginx:1.12
       ports:
       - containerPort: 88
     affinity:
       nodeAffinity:
         requiredDuringSchedulingIgnoredDuringExecution:
           nodeSelectorTerms:
           - matchExpressions:
             - key: disktype
            operator: Notin
               values:
               - ssd
```

Node Affinity - Reqd-NotIn



Worker 1





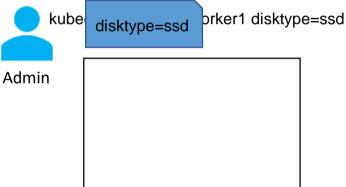
```
Developers
      Worker 2
```

```
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







Worker 1 Worker 2

```
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
```







Questions

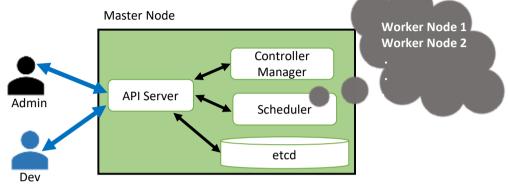




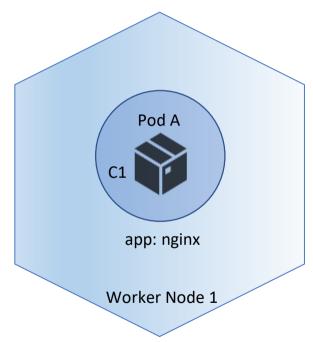
Maintenance & Troubleshooting

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



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



```
root@kubeadm-master:/home/ubuntu# kubect1 logs kube-apiserver-kubeadm-master -n kube-system
Flag -- insecure-port has been deprecated. This flag will be removed in a future version.
                            1 server.go:618] external host was not specified, using 18.8.8.4
18718 14:27:83.294371
18718 14:27:83.294756
                            1 server.go:148] Version: v1.18.6
                            1 plugins go:158] Loaded 12 mutating admission controller(s) successfully in the following order: NamespaceLifecycle,LimitHanger,ServiceAccount,N
18718 14:27:83.814846
odeRestriction. TaintNodesByCondition. Priority. DefaultTolerationSeconds. DefaultStorageClass. StorageObjectInUseProtection. RuntimeClass. DefaultIngressClass. MutatingAdmissionWeb
18718 14:27:83.814382
                            1 plugins.go:161] Loaded 18 validating admission controller(s) successfully in the following order: LimitWanger, ServiceAccount, Priority, Persisten
tVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ValidatingAdmissionWebhook, ResourceQuota.
                            1 plugins.go:158) Loaded 12 mutating admission controller(s) successfully in the following order: NamespaceLifecycle, LimitRanger, ServiceAccount, N
18718 14:27:83.822264
odeRestriction, TaintNodesByCondition, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, RuntimeClass, DefaultIngressClass, MutatingAdmissionWeb
hook.
18718 14:27:83.838255
                            1 plugins.go:161] Loaded 18 validating admission controller(s) successfully in the following order: LimitRanger.ServiceAccount.Priority.Persisten
tVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ValidatingAdmissionWebbook, ResourceQuota.
18718 14:27:83.833987
                            1 client.go:3611 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



```
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
10718 14:33:36.410772
                            1 node.go:136] Successfully retrieved node IP: 10.0.0.5
                            1 server others.go:186] Using iptables Proxier.
10718 14:33:36.410937
                            1 server others.go:4361 detect-local-mode set to ClusterCIDR, but no cluster CIDR defined
W0718 14:33:36.410965
                            1 server others.go:4471 detect-local-mode: ClusterCIDR . defaulting to no-op detect-local
10718 14:33:36.410972
                            1 server.go:583] Version: v1.18.6
10718 14:33:36.411420
                            1 conntrack.go:1001 Set sysctl 'net/netfilter/nf conntrack max' to 131072
10718 14:33:36.411959
                            1 conntrack.go:52] Setting of conntrack max to 131072
10718 14:33:36.411995
10718 14:33:36.412168
                            1 conntrack.go:100] Set sysctl 'net/netfilter/nf conntrack tcp timeout established' to 86400
                            1 conntrack.go:100] Set sysctl 'net/netfilter/nf conntrack tcp timeout close wait' to 3600
10718 14:33:36.412247
                            1 config.go:315] Starting service config controller
10718 14:33:36.414454
10718 14:33:36.414484
                            1 shared informer.go:223] Waiting for caches to sync for service config
                            1 config.go:133] Starting endpoints config controller
10718 14:33:36.414883
```

```
root@kubeadn-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu# kubectl logs kube-scheduler-kubeadm-master -n kube-system
10718 14:27:03.854038
                            1 registry.go:150) Registering EvenPodsSpread predicate and priority function
                           1 registry.go:150] Registering EvenPodsSpread predicate and priority function
18718 14:27:83.854115
18718 14:27:84.946783
                           1 serving.go:313] Generated self-signed cert in-memory
                           1 authentication.go:349] Unable to get configmap/extension-apiserver-authentication in kube-system. Usually fixed by 'kubectl create rolebinding
W8718 14:27:89,252268
 -n kube-system ROLEBINDING NAME --role=extension-apiserver-authentication-reader --serviceaccount=YOUR NS:YOUR SA'
                           1 authentication.go;297] Error looking up in-cluster authentication configuration; configmaps "extension-apiserver-authentication" is forbidden:
W8718 14:27:89.252316
User "system:kube-scheduler" cannot get resource "configmaps" in API group "" in the namespace "kube-system"
                           1 authentication.go:298] Continuing without authentication configuration. This may treat all requests as anonymous.
W8718 14:27:89.252328
                           1 authentication.go:299] To require authentication configuration lookup to succeed, set --authentication-tolerate-lookup-failure=false
W0718 14:27:09.252336
                           1 registry.go:150] Registering EvenPodsSpread predicate and priority function
10718 14:27:09.276928
18718 14:27:89.276957
                           1 registry.go:1501 Registering EvenPodsSpread predicate and priority function
18718 14:27:89.281887
                           1 secure serving.go:1781 Serving securely on 127.8.8.1:18259
                           1 configmap_cafile_content.go:282] Starting client-ca::kube-system::extension-apiserver-authentication::client-ca-file
18718 14:27:89.281359
                           1 shared informer.go:223] Waiting for caches to sync for client-ca::kube-system::extension-apiserver-authentication::client-ca-file
18718 14:27:89.281377
18718 14:27:09.281542
                           1 tlsconfig.go:2401 Starting DynamicServingCertificateController
```

kube-system Process Logs



```
TOOT@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-add r:127.0.0.1:6784 ipalloc-init:consensus=2 ipalloc-range:10.32.0.0/12 metrics-addr:0.0.0.8:6782 name:76:ec:70:b2:aa:58 nickname:kubeadm-master no-dns:true port:6783]

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 10.0.6]

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 14:37:38.993932 [allocator 76:ec:70:b2:aa:58] No valid persisted data

INFO: 2020/07/18 14:37:38.993934 [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: []
```





Network issue

- Service issues
- Service <-> Pod mapping issue
- Port mapping issue

> Application issues

- > Image pull error
- App error



Application Logs



```
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





Kubernetes Security

3 Step Security Check



- 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

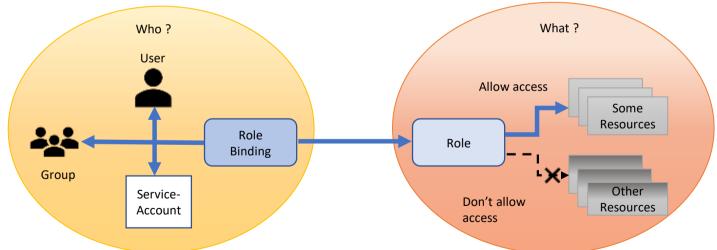


- Determine or create namespace
- Create certificate credentials for user
- > Set the credentials for the user to the namespace using a context
- Create a role for the expected task set
- Bind the user to the role
- Verify the user has limited access

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



- 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



- > 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
- A policy specifies if groups of pods are allowed to communicate with each other and other network endpoints
- Pods become isolated by attaching to a NetworkPolicy
- Once NetworkPolicy is attached to a pod, that pod will reject any connections that are not allowed by any NetworkPolicy

Network Security

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: hello-allow-from-foo
spec:
  policyTypes:
  - Ingress
  podSelector:
    matchLabels:
      app: hello
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: foo
```



```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
 name: foo-allow-to-hello
spec:
 policyTypes:
 - Egress
  podSelector:
   matchLabels:
      app: foo
  egress:
 - to:
    - podSelector:
        matchLabels:
          app: hello
```

Security Context



- 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 <u>user ID (UID)</u> 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



```
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



```
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







Questions





Cluster Resource Limit

Resource Quota



- 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

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

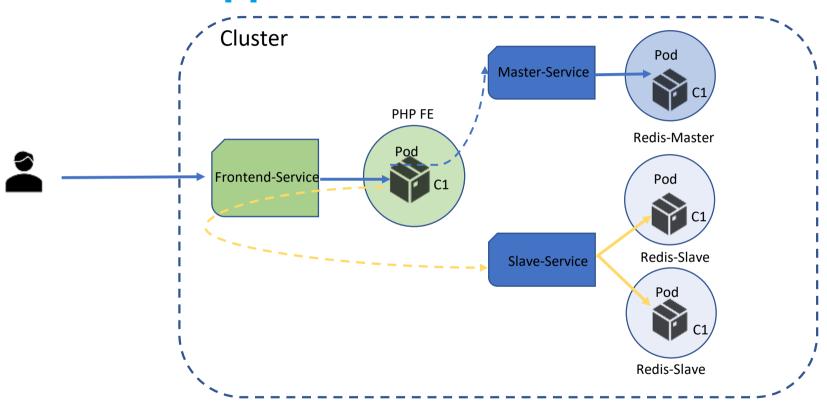




Working with Application Stack

Guestbook Application









Deploy Stateful Application

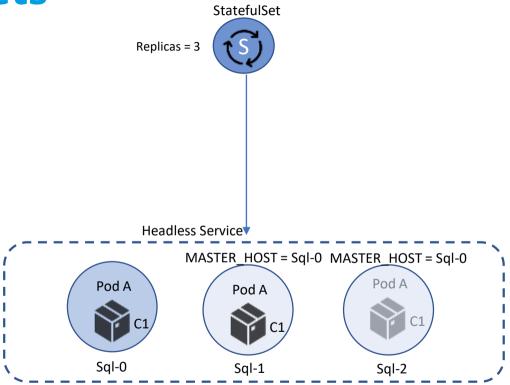
Database App as Deployment



- 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



- 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



- 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



- 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





Deploy Daemon Process

DaemonSet



- 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





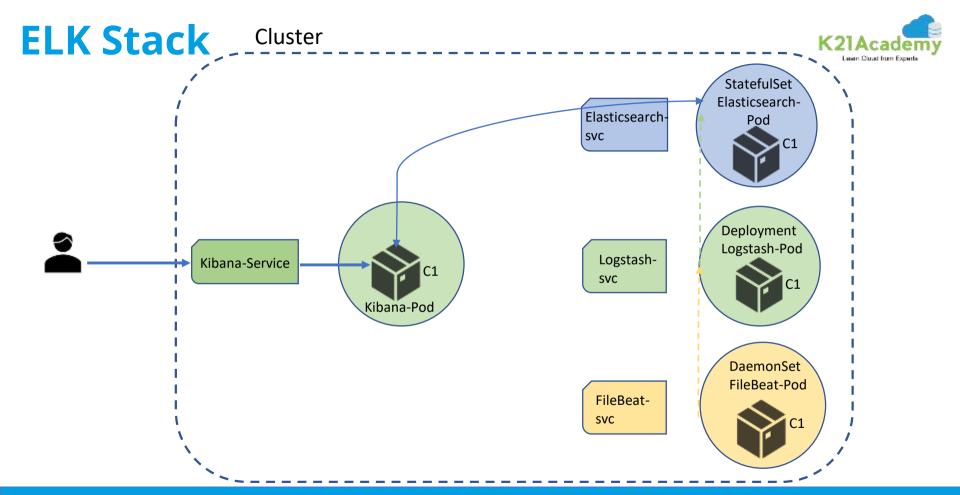


Questions





Working with Logging 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, highavailability 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



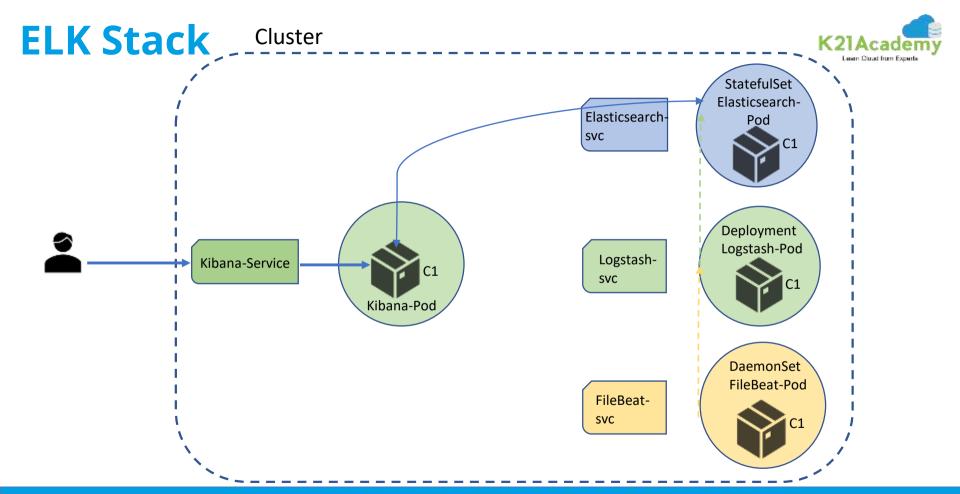
Filebeat-DaemonSet

- > 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









Questions

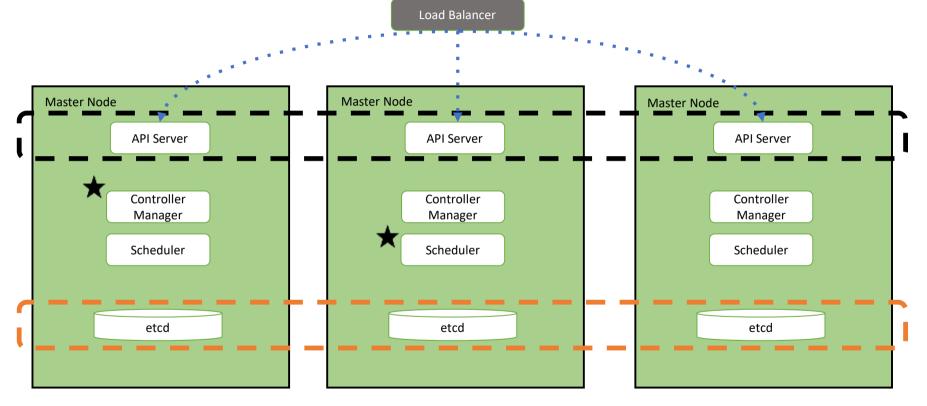




Highly Available Cluster & Upgrade

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

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



- 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 ETCOCTL_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



- 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



- > 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





Managed Kubernetes on Cloud

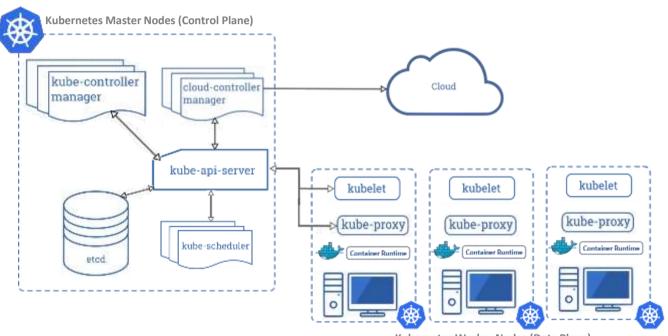
Kubernetes Adoption in Cloud



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

Kubernetes Architecture





Kubernetes Worker Nodes (Data Plane)





Advanced Scheduling

Adv Scheduling - Taint & Tolerations



- 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
- One or more taints are applied to a node; this marks that the node should not accept any pods that do not tolerate the taints

\$
\$ \$ aubect| taint node aks-apentpool-AR017546-wess800001 disktype-magnetic:NoSchedule
node/aks-apentpool-40017546-wess0000001 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



> 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

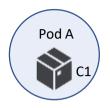
> Operator:

- **Equal:** The key/value/effect all three parameters must match. This is the default
- **Exists:** The key/effect both parameters must match. You must leave a blank value parameter, which matches any

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

Taint- NoSchedule







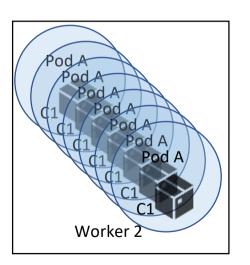


kubectl taint node worker1 disktype=magnetic:NoSchedule

Worker 1







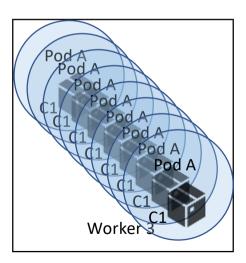


apiVersion: v1 kind: Pod metadata:

name: tt-pod1

spec:

containers: - name: nginx image: nginx



Taint & Toleration - NoSchedule





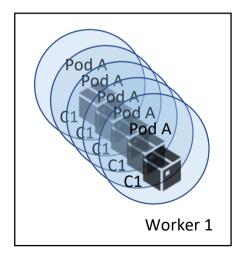


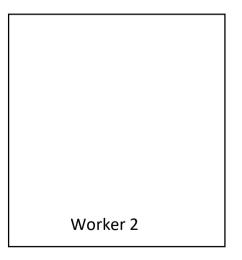




kubectl taint node worker1 disktype=magnetic:NoSchedule







```
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



> 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

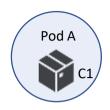
> Operator:

- **Equal:** The key/value/effect all three parameters must match. This is the default
- **Exists:** The key/effect both parameters must match. You must leave a blank value parameter, which matches any

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

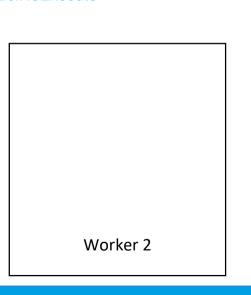
Taint- NoExecute







kubectl taint node worker1 disktype=magnetic:NoExecute Admin Worker 1





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

- name: nginx image: nginx

Worker 3

Taint Effect and Operator



> 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:** The key/effect both parameters must match. You must leave a blank value parameter, which matches any

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







Questions





Advanced Networking

Kubernetes Service - LoadBalancer

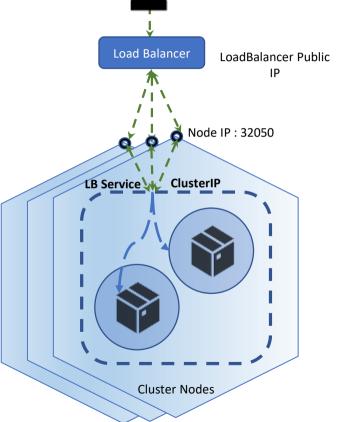


> Exposes the Pod to outside the cluster

Assigns a Private IP Address in Cluster IP range

Assigns a port on the worker nodes to expose it outside cluster

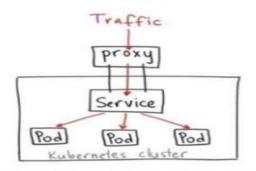
Create a Load Balancer in Cloud

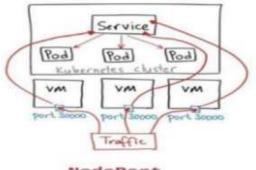


K8s Service Types

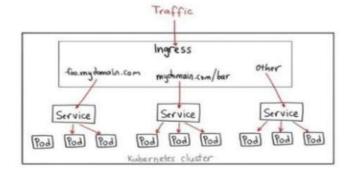




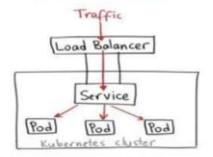




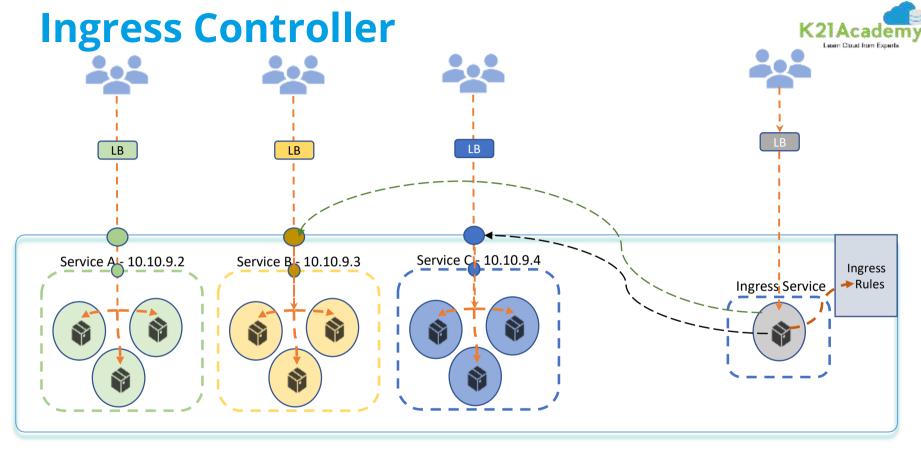
NodePort



LoadBalancer



Ingress



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





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





Dynamic Storage Provisioning

Yaml Definitions



```
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
```

```
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]}
```

```
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



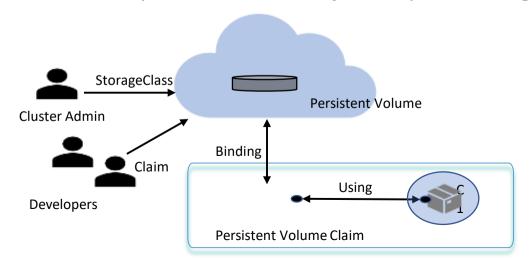
- A StorageClass provides a way for administrators to describe the "classes" of storage they offer
- Different classes might map to quality-of-service levels, or to backup policies
- Each StorageClass has a provisioner that determines what volume plugin is used for provisioning PVs

```
manta@Azure:-/Kubernetes$ kubectl describe sc managed-premium
Name:
IsDefaultClass:
                  kubectl.kubernetes.io/last-applied-configuration={"allowVolumeExpansion":true, apiV
Annotations:
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
                        cachingmode=ReadOnly, kind=Managed, storageaccounttype=Premium LRS
Parameters:
AllowVolumeExpansion:
                        True
MountOptions:
                        <none>
ReclaimPolicy:
                        Delete
VolumeBindingMode:
                        Immediate
Events:
                        <none>
manta@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



Introduction to Kubernetes

- Kubernetes Architecture
- Kubernetes Object
- Kubernetes Pod
- Kubernetes Networking

Kubernetes Deep Dive

- Kubernetes Service
- Highly Available and Scalable Application
- Autoscaling with HPA
- Persistent Storage
- Advanced Scheduling

Summary: Module



- Maintenance & Troubleshooting
- Kubernetes Security
 - RBAC Role Based Access Control
 - Role and RoleBinding
 - Network Security
- Working with Application Stack
- Deploy Stateful Application
- Kubernetes Cluster HA
 - Backing Up and Restoring Cluster
 - Master HA in K8s Cluster

Summary: Module



- Deploy Daemon Process
- Working with Logging Stack
- Kubernetes Next Level
 - ➤ Adv Scheduling Taint & Tolerations
 - Probing Parameters
 - Kubernetes Service LoadBalancer
 - Managed Kubernetes on Cloud
 - Advanced Networking
 - ➤ 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/