**Content**

[1. Core concepts 2](#_Toc80615427)

[1. Kubernetes definition files 3](#_Toc80615428)

[2. Kubectl command structure 4](#_Toc80615429)

[3. Imperative vs Declarative 5](#_Toc80615430)

[2. Scheduling 6](#_Toc80615431)

[3. Exam tricks 7](#_Toc80615432)

[1. Bash 7](#_Toc80615433)

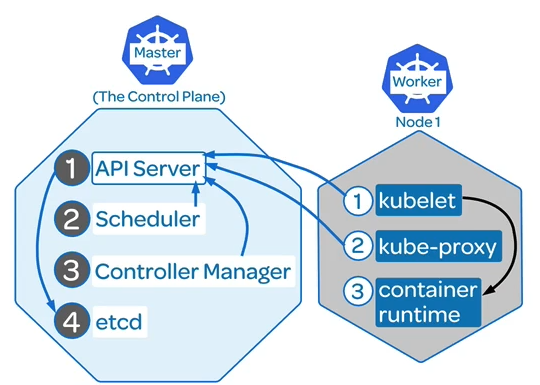
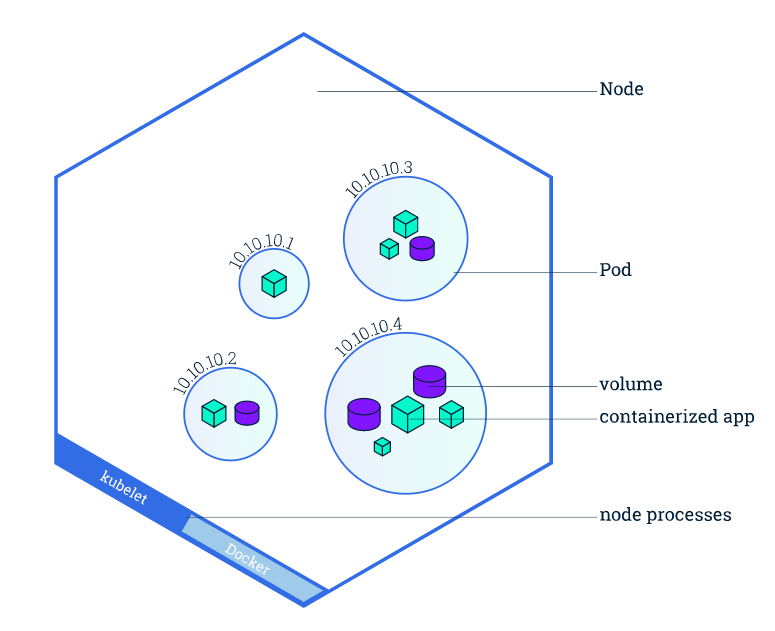
[2. Kubernetes 7](#_Toc80615434)

[4. Necessary linux knowledge 8](#_Toc80615435)

1. Core concepts

* **Kubeadm** is a tool built to provide kubeadm init and kubeadm join as best-practice "fast paths" for creating Kubernetes clusters.
* **minikube** is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.
* **etdc** – key value database
* **kube-api server** – kubectl utility is reaching to this server
* **controller** – a proces, which continuously monitors the state of various componens within the system. Works towards brining the whole system towards the desired functioning state. Intelligence is implemented through the various controllers.
  + **node controller**
  + **replication controller**
  + **deployment c**.
  + **job c.**
  + ....
* **kube controller manager** – the controllers are packaget into this one. When you install this, the different controllers get installed as well.
* **kube-scheduler** – responsible for scheduling pods on nodes. Only responsible for deciding which pod goes on which node. It does not place the pod on the nodes. Thats the job of the kubelet. Only decides, which pod goes where.
  + filter nodes
  + rank nodes
  + chooses the best place
* **kubelet** – captain on the ship, who creates the pod the ship.
  + register node
  + create PODS
  + monitor NODE & PODS, report
* **kube proxy** – within a kubernetes cluster, every pod can reach every other pod. This is accomplished by deploying a pod networking solution to the cluster. There are many solutions.
  + services in kubernetes (they are in memory, not a pod. They are accessible)
* **control plane** - The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.
* **Pods** & **Nodes**

A Pod always runs on a Node. A Node is a worker machine in Kubernetes and may be either a virtual or a physical machine, depending on the cluster. Each Node is managed by the control plane. Kubernetes does not deploy containers directly into worker nodes. They are encapsulated in kuberentes objects, called pods. Pod 1:1 containers. Pods however can have multi containers, it only makes sense if they are not the same kind.



* 1. Kubernetes definition files

pod-definition.yml

|  |  |
| --- | --- |
| Kind | Version |
| POD | V1 |
| Service | V1 |
| ReplicaSet | Apps/v1 |
| Deployment | Apps/v1 |

# the version of the kubernetes api we are using

apiVersion:v1

kind:POD

metadata:

name: myapp-pod

labels: # whatever youy wish, like tagging.

app: myapp

type: front-end

spec:

containers:

-name: nginx-container

image: nginx

kubectl create –f pod-definition.yml

kubectl get pods

kubectl describe pod myapp-pod

* 1. Kubectl command structure

kubectl [comand] [resource type] [name] [flags]

Commands:

* create
* delete
* describe
* get
* ...

Resource type

* namespaces (ns)

Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces.

* nodes (no)

Kubernetes runs your workload by placing containers into Pods to run on Nodes.

* pods (po)

Pods are the smallest deployable units, a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers.

* services (svc)

responsible for communicaiton thus loose coupling

* + NodePort
  + ClusterIP
  + LoadBalancer
* replicationcontrollers (rc)

A ReplicationController ensures that a specified number of pod replicas are running at any one time.

* replicasets (rs)

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time.

* deployments

A Deployment provides declarative updates for Pods and ReplicaSets.

* 1. Imperative vs Declarative

Impretive:

* kubectl run - - image=nginx nginx
* create
* expose
* edit
* scale
* set
* replace
* delete
* ...

Declarative

* kubectl apply –f file.yaml

1. Scheduling

The Kubernetes scheduler is a control plane process which assigns Pods to Nodes.

* Taints
* Tolerations
* Node Selectors
* Node Affinity
* Labels and Selectors
* Manual Scheduling
* Daemon Sets (e.g. monitoring solution, agents). Deploys automatically to all pods. (Kube proxy, wave net)

1. Exam tricks
   1. Bash

* vi editor

kubectl config view --minify | grep namespace

* 1. Kubernetes

kubectl create deployment --image=nginx nginx **--dry-run=client -o yaml > generate.yaml**

alias k=kubectl

kubectl run redis –image=redis:alpine –labels=”tier=db”

kubectl expose pod redis –name redis-service –port 6

# Create a pod called httpd using the image httpd:alpine in the default namespa# Next, create a service of type ClusterIP by the same name (httpd).

# The target port for the service should be 80.

kubectl run httpd --image=httpd:alpine --port=80 --expose

1. Necessary linux knowledge

* systemd
* journald
* systemctl
* rsyslog
* Interrupts (software, hardware ?!) and how to handle interrupts
* Stress
* Linux user management (for interviews)