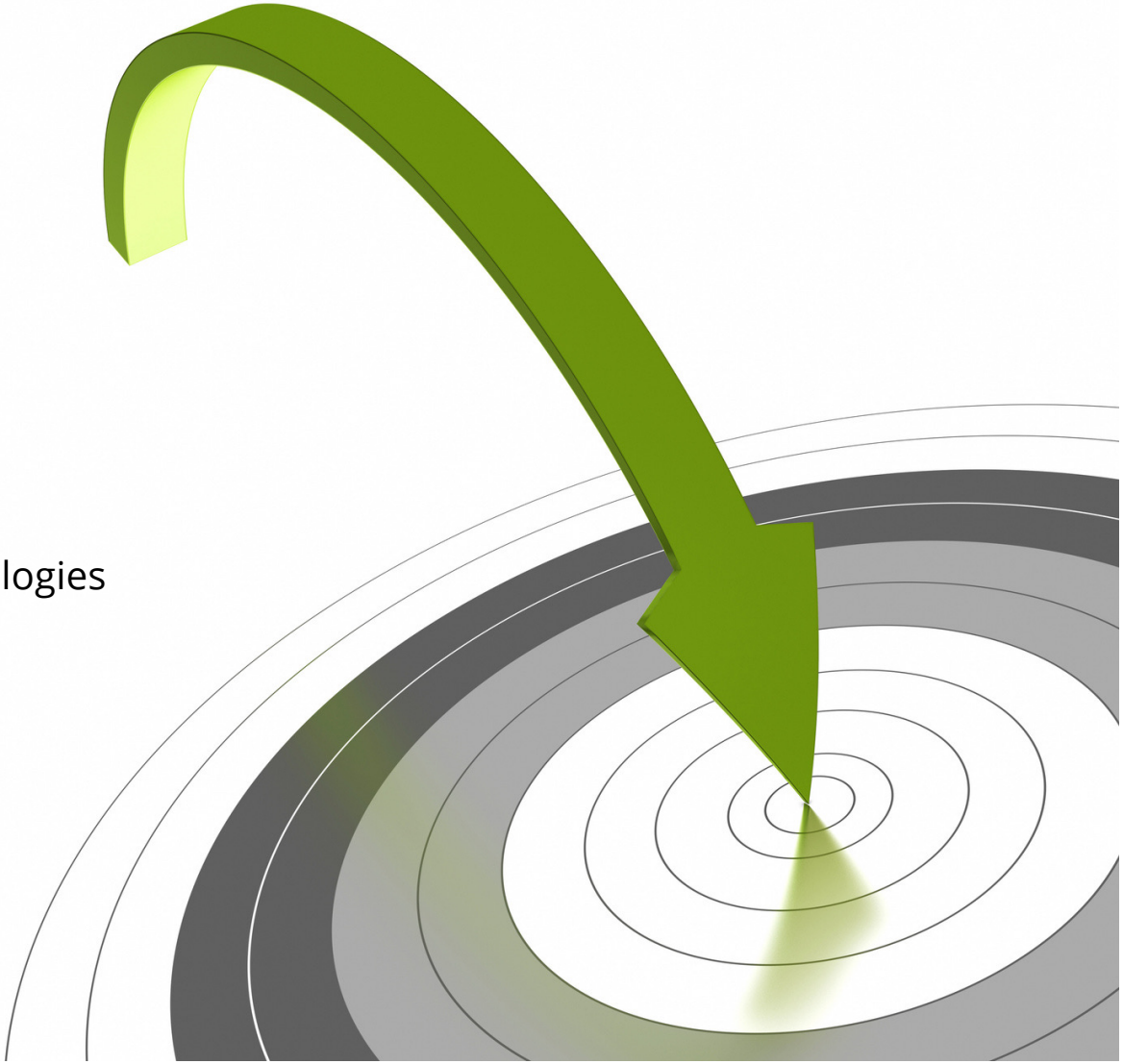
Kubernetes

ENABLING DISCOVERY | POWERING EDUCATION | SHAPING WORKFORCES

Objectives

By the end of this module, you will be able to:

- Become familiar with Kubernetes
- Discuss Kubernetes architecture
- Install Kubernetes using Kubeadm
- Setup Minikube
- Become familiar with objects and terminologies

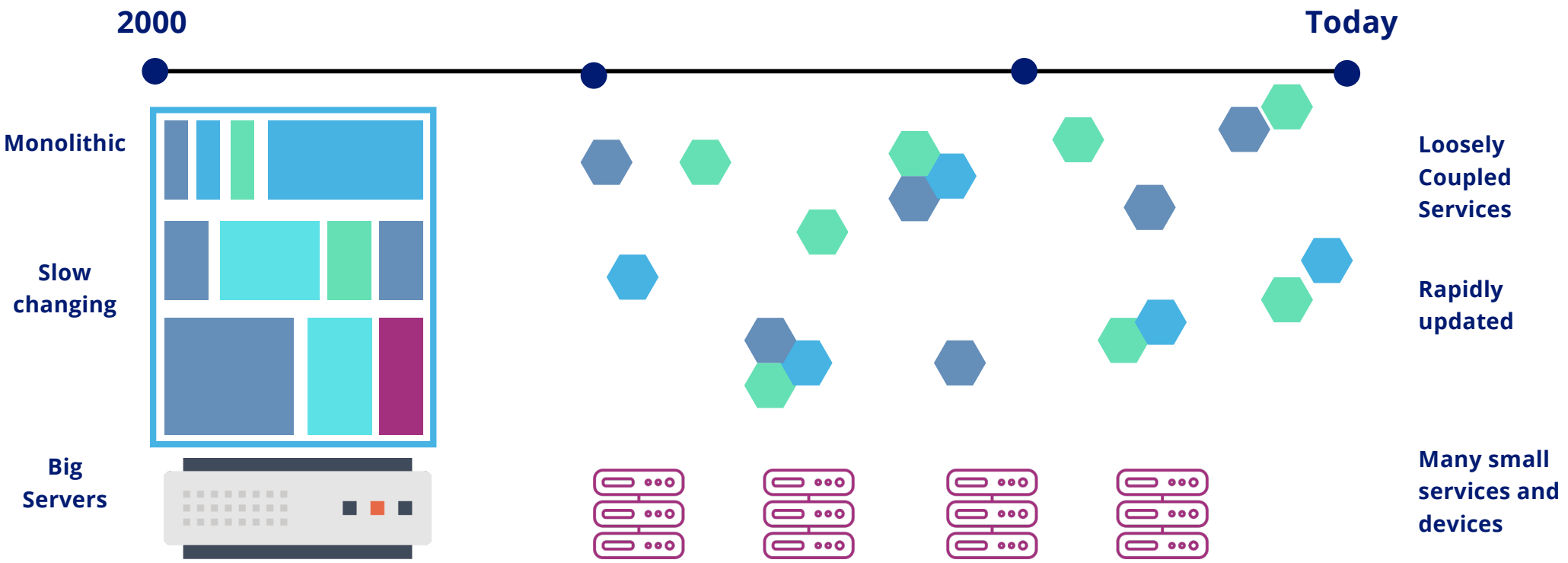


Introduction to Kubernetes

What We Will Cover

- Monolithic vs. Microservices
- What is a container orchestration
- Need for Kubernetes
- Kubernetes alternatives

Monolithic vs. Microservices



Monolithic vs. Microservices

Monolithic	Microservices
Architecture is based on a single logical programming unit. Everything is located within one “Application Block”	Architecture consists of a series of small services that work independently and communicate with each other.
Needs full re-compile and re-test for minor changes.	Each service can be changed independently.
Can lead to Single point of failure	Makes the application stateless and highly available.
Scaling of application became difficult.	Highly scalable.
Developed on a single programming language	Each service can be developed in a different programming language. Integration happens via APIs.

Container Orchestration

Container orchestration is a tool that provides a framework for managing containers and maintains the lifecycle of containers to manage and automate various tasks.

Orchestration Tasks:

- Deployment of apps
- Resource allocation
- Configuration and scheduling
- Load balancing and routing
- Auto scaling and healing
- Health check and monitoring

Kubernetes Alternatives

Kubernetes Architecture

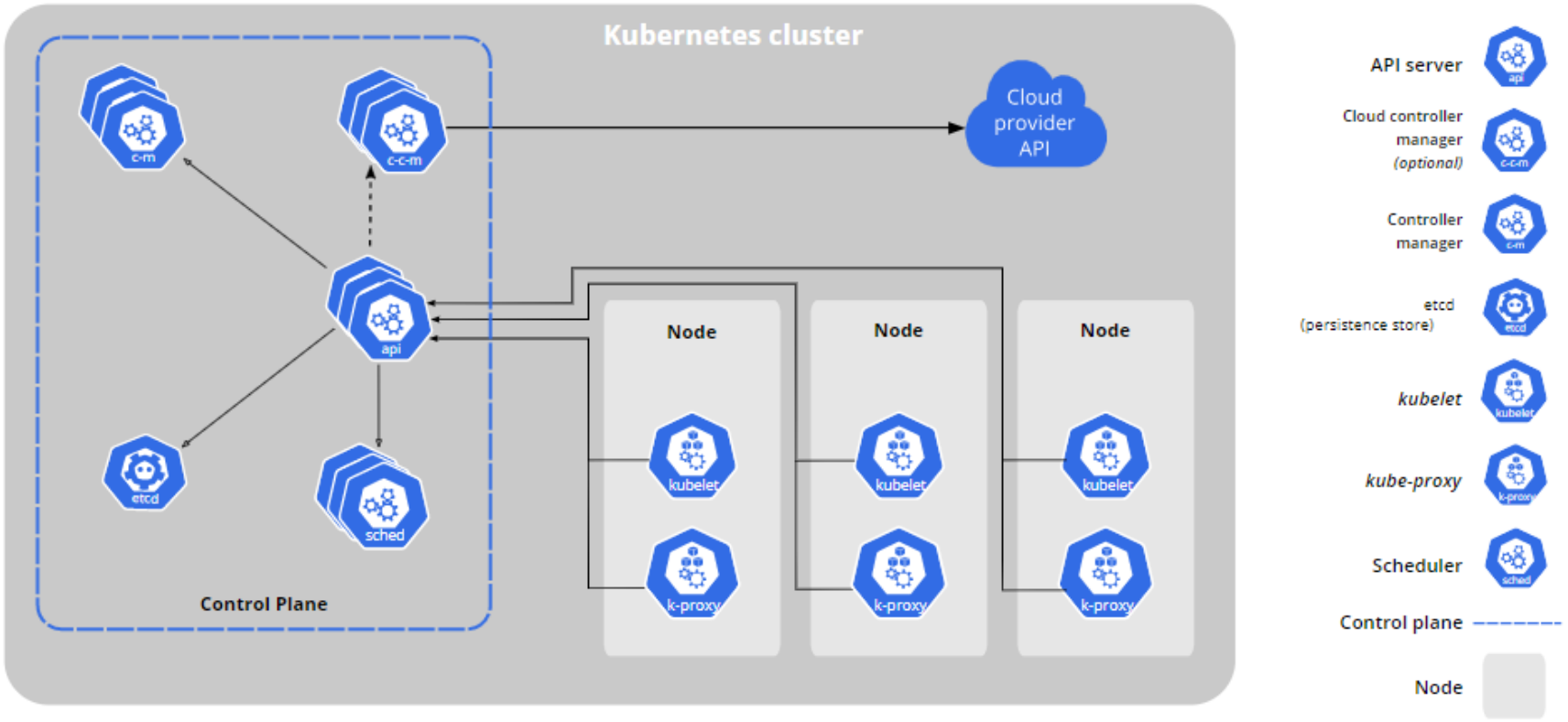
What We Will Cover

- What is Kubernetes
- Kubernetes cluster insights and components
- Control plane components
- Data plane components

What is Kubernetes

- **Kubernetes** is an open-source platform to run containerized workloads.
- **K8's** an abbreviation refers to the eight letters between "K" and "s."
- **K8's** was open-sourced by google with a decade of experience running production workloads.
- **Kubernetes** can efficiently run and manages micro-services that are distributed and loosely coupled.

Kubernetes Architecture



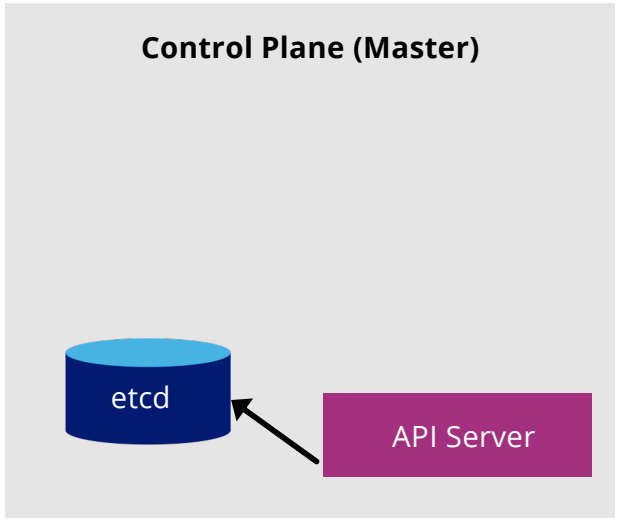
Source: https://kubernetes.io/docs/concepts/overview/components/

Kubernetes Control Plane

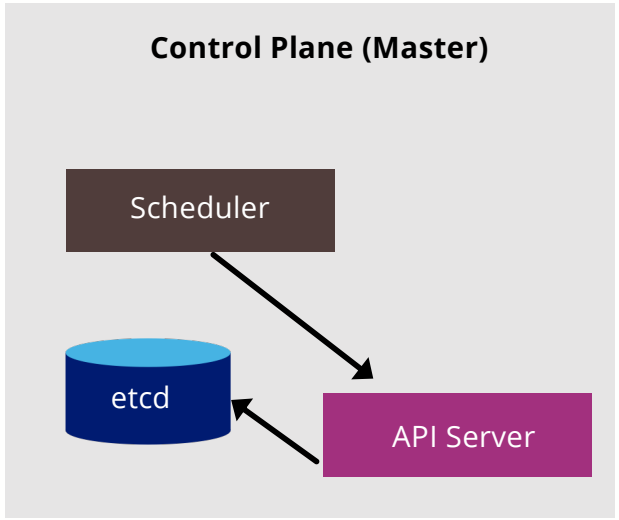
- **Kube-Apiserver** – It's the gateway to K8's, does API Management and Discovery.
- **Etcd** – Key-Value store of Kubernetes for all Cluster Data.
- **Kube-Scheduler** – Scans the API Server to select a node for the pod, considering various scheduling decisions and resource requirements.
- **Kube-Controller-Manager** – The control Loop which enables the Current State to Desired State configuration in the cluster.

Kubernetes Components

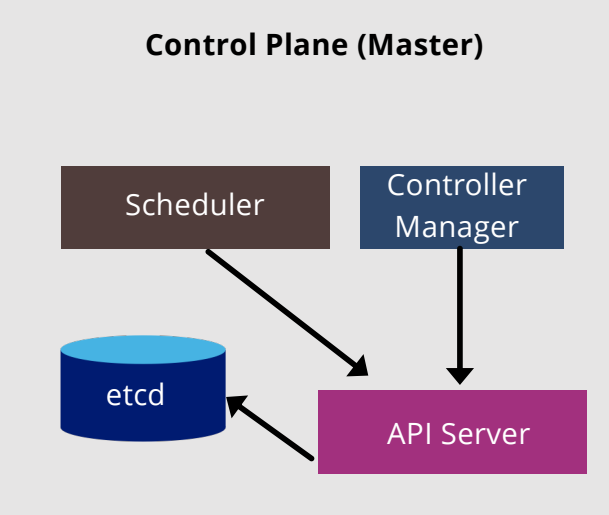
Kubernetes Components



Kubernetes Components



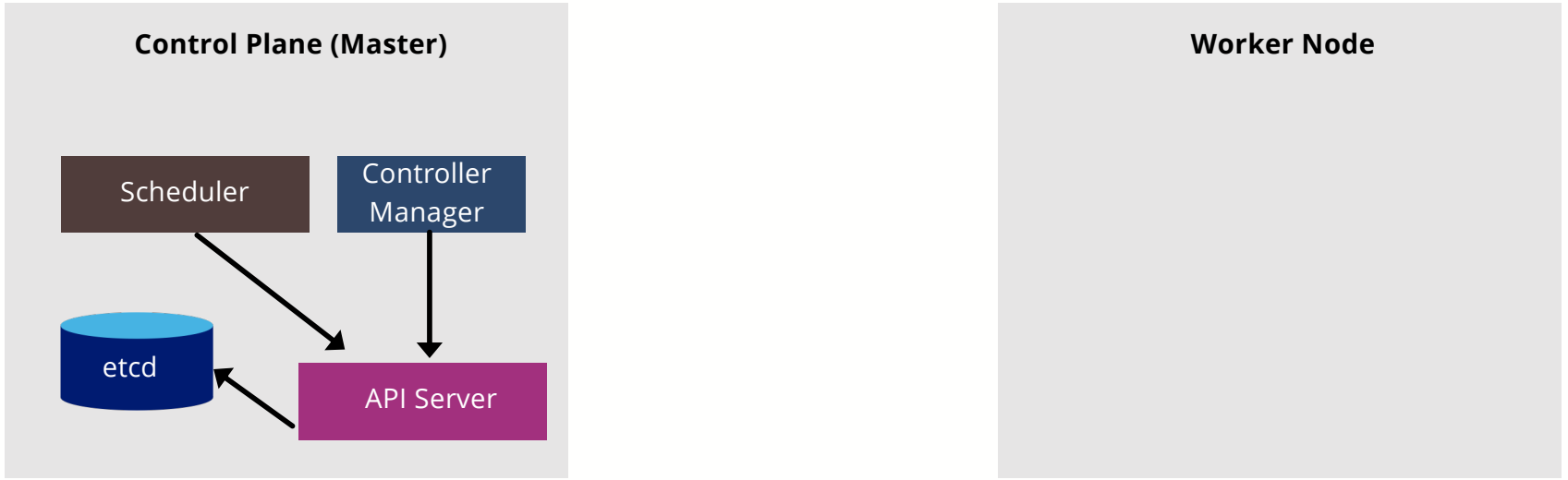
Kubernetes Components



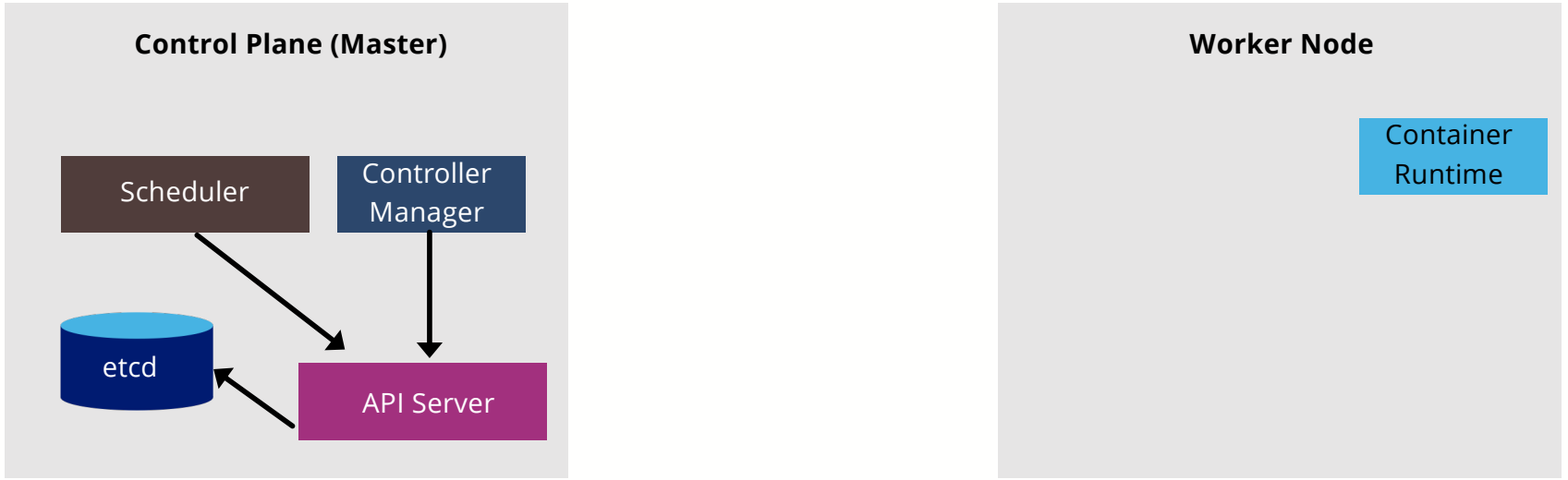
Kubernetes Data Plane (Node) Components

- **Kubelet** – Agent running on all nodes to ensure the Pods Running state.
- **Kube-Proxy** – Maintains Network rules and communication to the pods and is responsible for implementing K8's Services.
- **Container runtime** – Implementation of K8's CRI in each node to run Pods. Several Container runtimes are:
  - Containerd
  - CRI-O
  - Docker

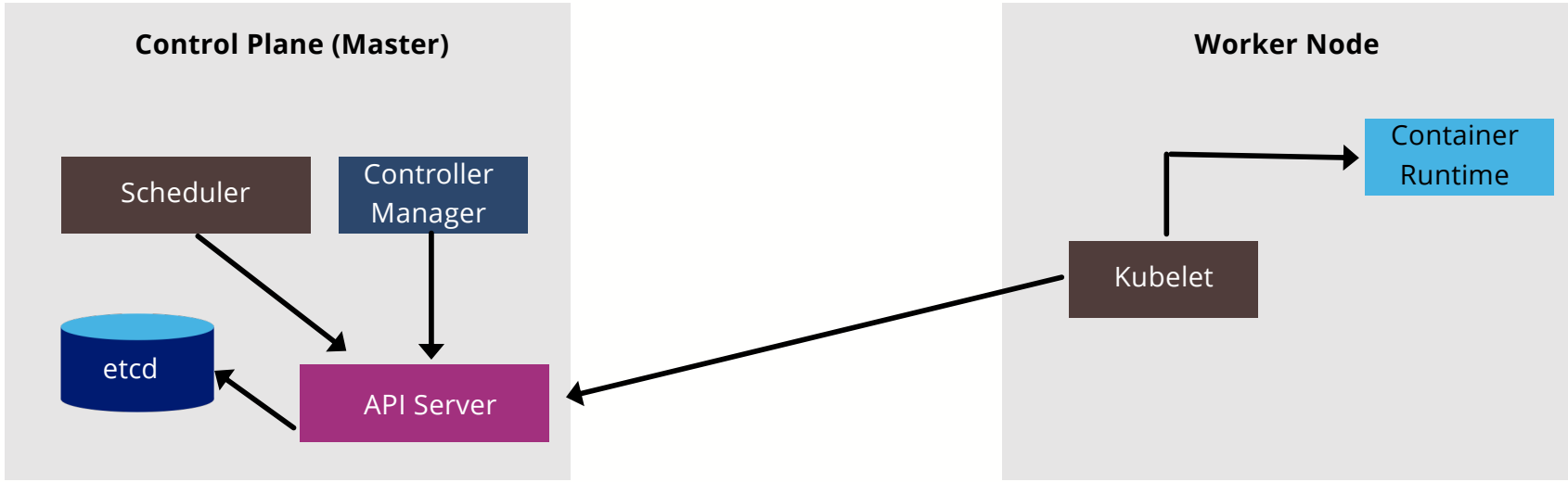
Kubernetes Components



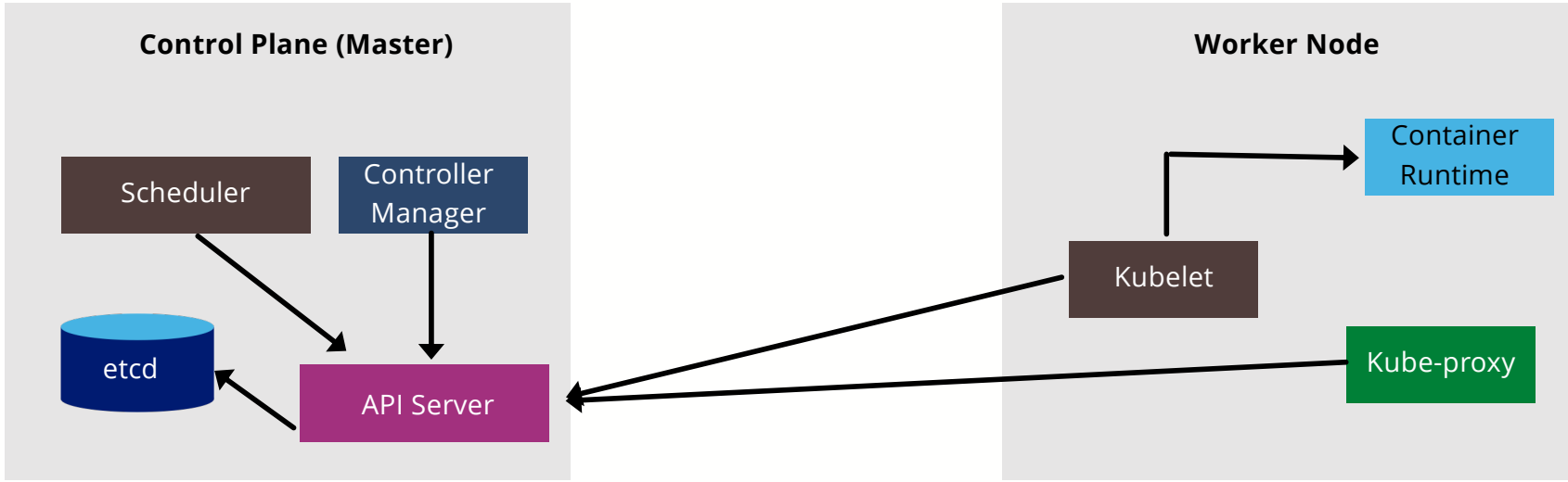
Kubernetes Components



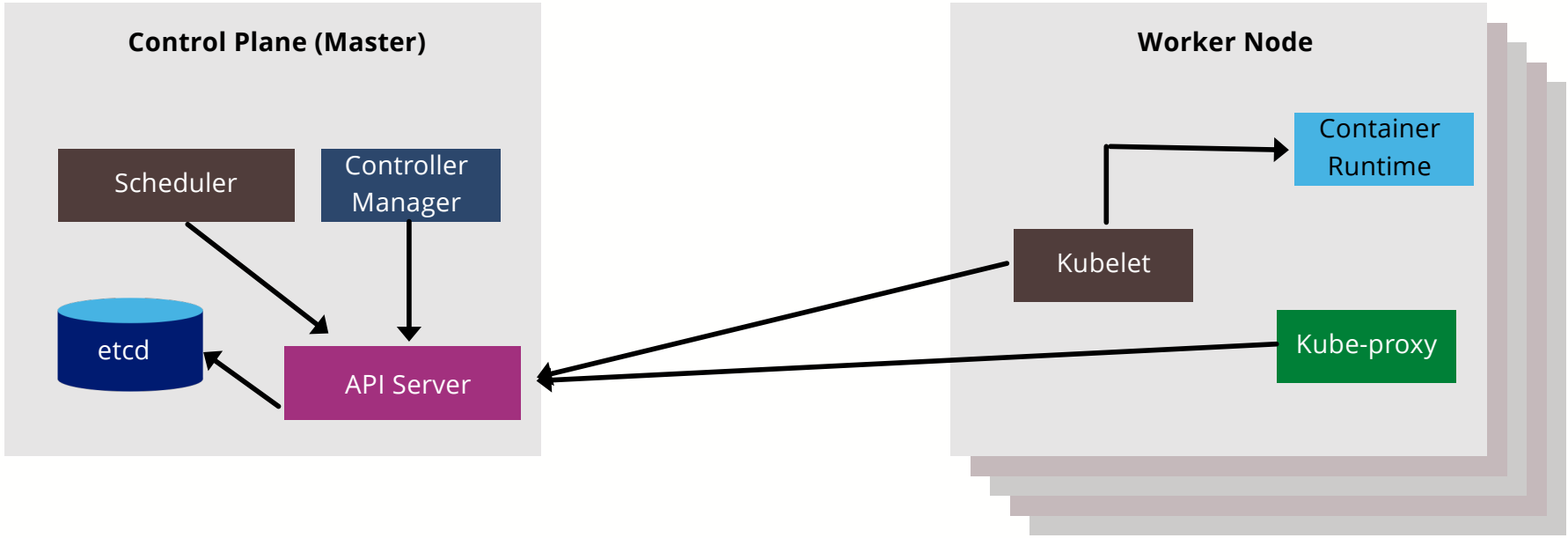
Kubernetes Components



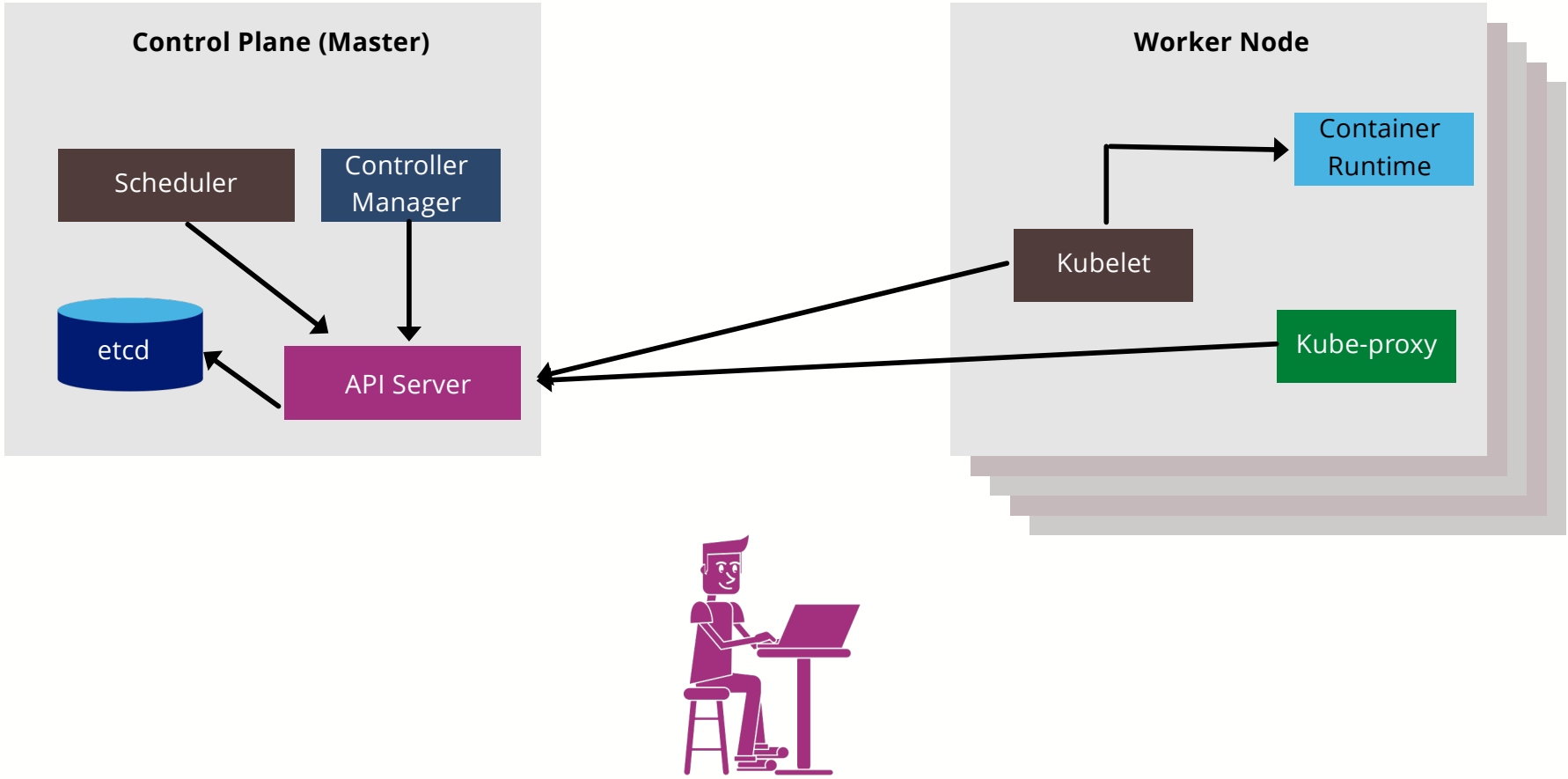
Kubernetes Components



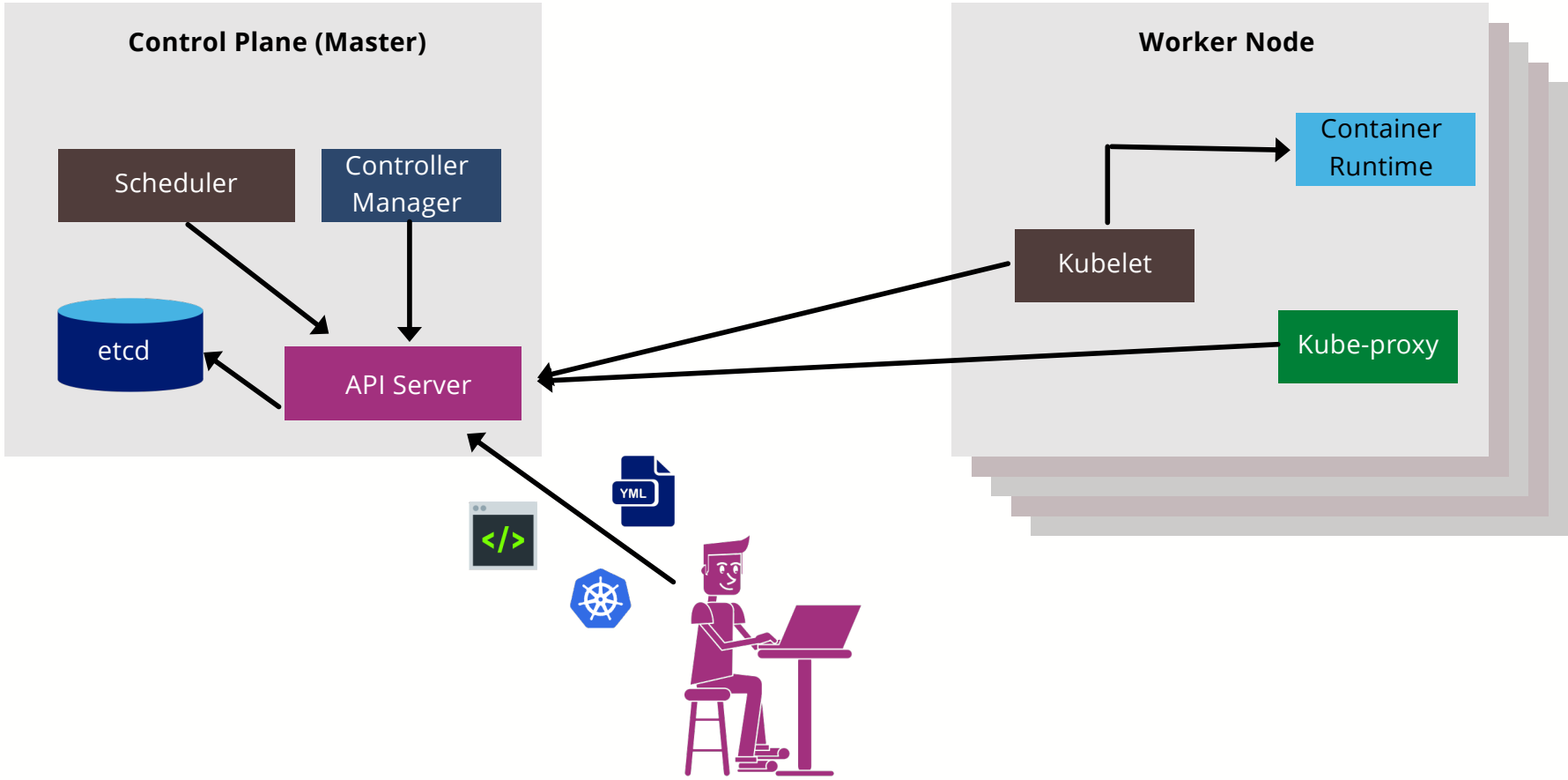
Kubernetes Components



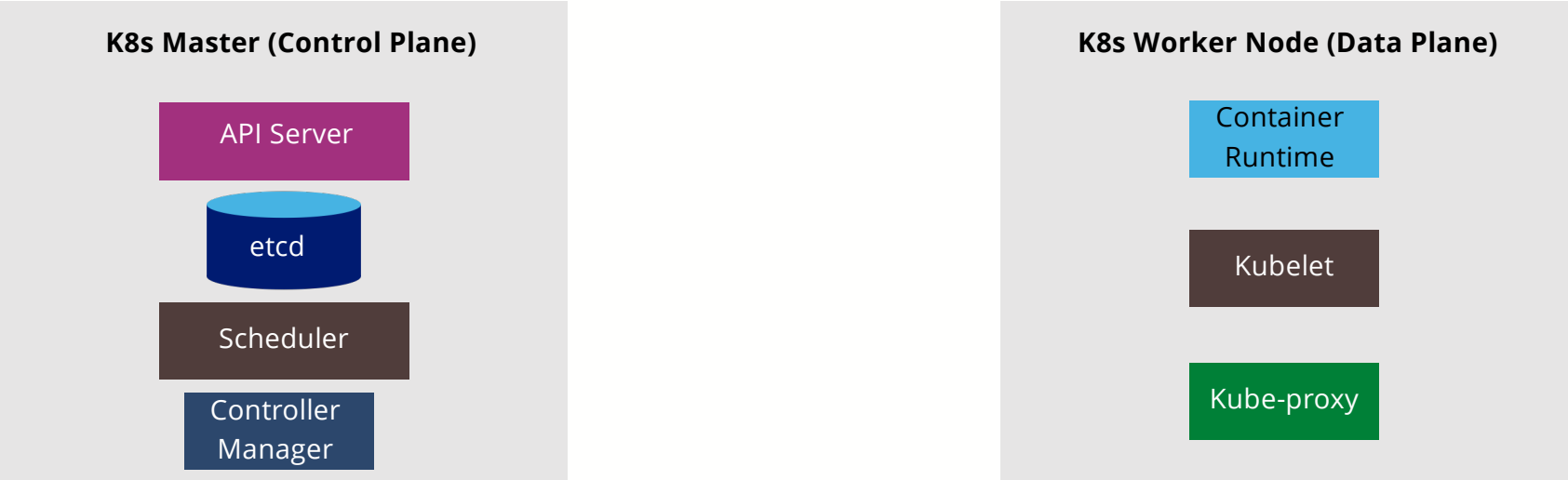
Kubernetes Components



Kubernetes Components



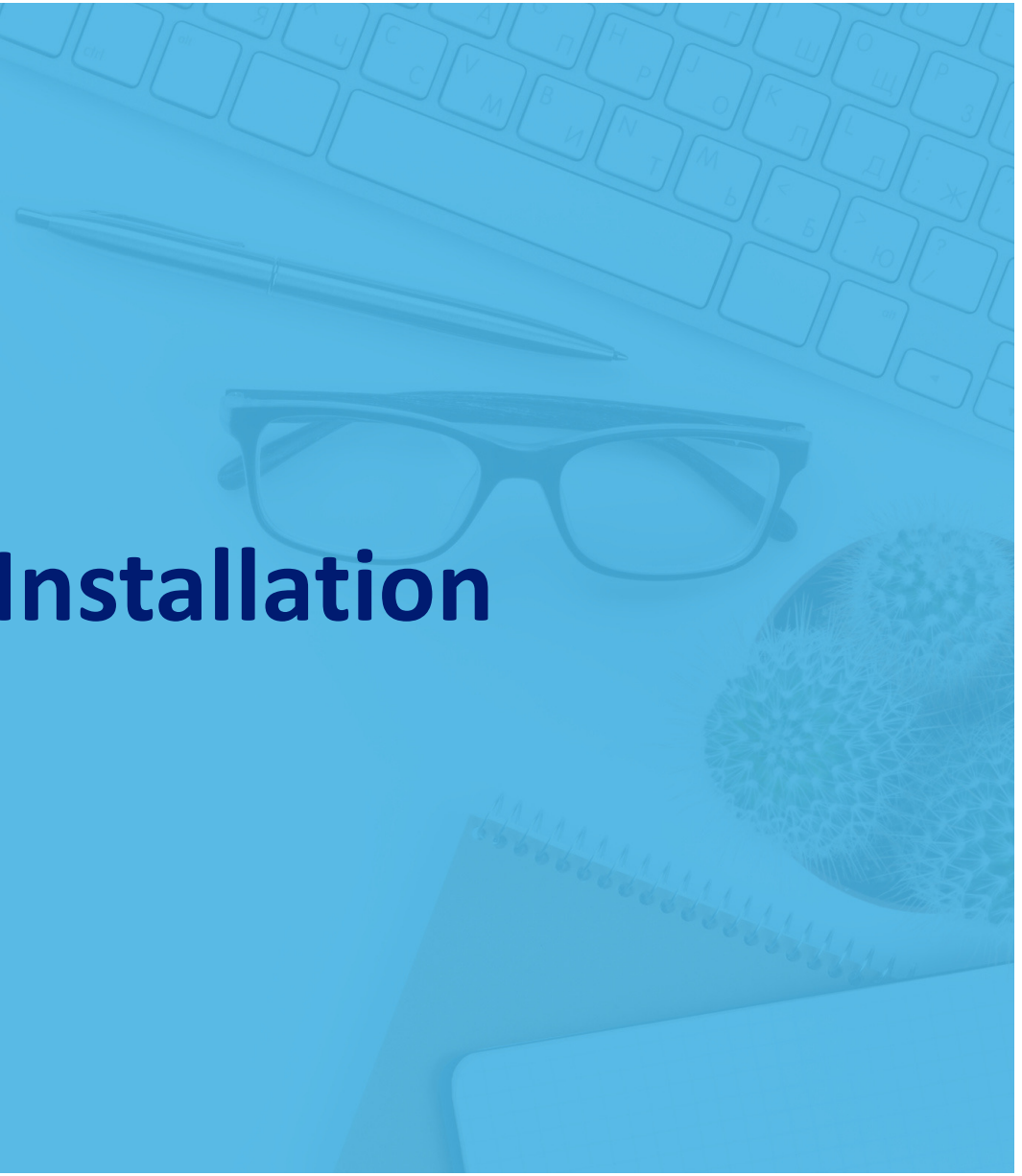
What We Learned



Users can communicate to Kubernetes using:

- Kubectl (command line)
- Kube API (programmatically)
- K8s Dashboard

Kubernetes Installation



What We Will Cover

- Pre-requisites
- Master node installation
- Worker node configuration
- Cluster networking

Kubernetes Installation – Prerequisites

- Container runtime like Docker, Containerd, or CRI-O to be pre-installed and running.
- Swap should be turned off.
- Master node's minimum memory requirement is 2GB and needs 1.5 cores of CPU.
- Worker node's minimum memory requirement is 2GB and needs 0.7 cores of CPU.
- Choose one major Linux distribution like Ubuntu 16.04, Centos/RHEL 7, Debian 9, etc.

Kubernetes Installation – Master & Worker Nodes Component Install

Turn off Swap and add Docker Repository

```
##Turn the Swapoff & Permanently Disable Linux swap space

sudo swapoff –a #Hash out swap space in file /etc/fstab with a system reboot.

sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release

##Add Docker's official GPG key

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg

##Add Docker's stable repository

echo \ "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

Note: Run the above commands on both Manager and Worker Nodes.

Kubernetes Installation – Master & Worker Nodes Component Install

Docker Installation and hold off the package

```
##Choose a suitable version of docker that has Kubernetes support.
sudo apt-get update
apt-cache madison docker-ce
apt-cache madison docker-ce-cli
##Install Docker replacing the version from next step
sudo apt-get install docker-ce=<VERSION_STRING> docker-ce-cli=<VERSION_STRING>
containerd.io
##Hold the docker packages to prevent from upgrade
sudo apt-mark hold docker-ce docker-ce-cli
```

Note: Run the above commands on both Manager and Worker Nodes.

Kubernetes Installation – Master & Worker Nodes Component Install

Configure IP Tables

```
##Configure Iptables to see bridge traffic
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

Note: Run the above commands on both Manager and Worker Nodes.

Kubernetes Installation – Master & Worker Nodes Component Install

```
##Download the Google Cloud public signing key

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg

##Add Kubernetes repository

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list

##Update and choose a package version

sudo apt-get update

apt-cache madison kubeadm
```

Note: Run the above commands on both Manager and Worker Nodes.

Kubernetes Installation – Master & Worker Nodes Component Install

```
##Install Docker replacing the version from next step
sudo apt-get install kubeadm=<VERSION_STRING> kubectl=<VERSION_STRING>
kubelet=<VERSION_STRING>

##Hold the kubernetes packages to prevent from upgrade
sudo apt-mark hold kubeadm Kubectl kubelet
```

Note: Run the above commands on both Manager and Worker Nodes.

Kubernetes Installation – Master & Worker Nodes Component Install

```
##Initialize the master node.
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
##Reset Kubeadm and re-run it if the previous command fails
sudo kubeadm reset
##Root user command for Kubeconfig Setup
export KUBECONFIG=/etc/kubernetes/admin.conf
##Non-Root user commands for Kubeconfig Setup
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Note: Run the above commands on both Manager and Worker Nodes.

Kubernetes Installation – Master & Worker Nodes Component Install

```
##Choose Pod Network Add-on, Flannel and Calico being widely used Networks

##Deploying Flannel using Kubectl

kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
flannel.yml


##Deploying Calico using Kubectl

kubectl create -f https://docs.projectcalico.org/manifests/tigera-operator.yaml


##Get the token to join Worker nodes to Master

kubeadm token list
```

Note: Run the above commands on both Manager and Worker Nodes.

Kubernetes Installation – Master & Worker Nodes Component Install

```
##Join the Worker Node

##Run the below command as root user in worker node, this command is also an output of Kubeadm init

kubeadm join --token <token> <control-plane-host>:<control-plane-port> --discovery-token-ca-cert-hash sha256:<hash>


## Run this Command in Master Node

## List the nodes in your cluster and look for the status ready.

Kubectl get nodes  ## Master node command
```

Note: Run the above commands on both Manager and Worker Nodes.

Minikube Setup

WILEY

Minikube Installation

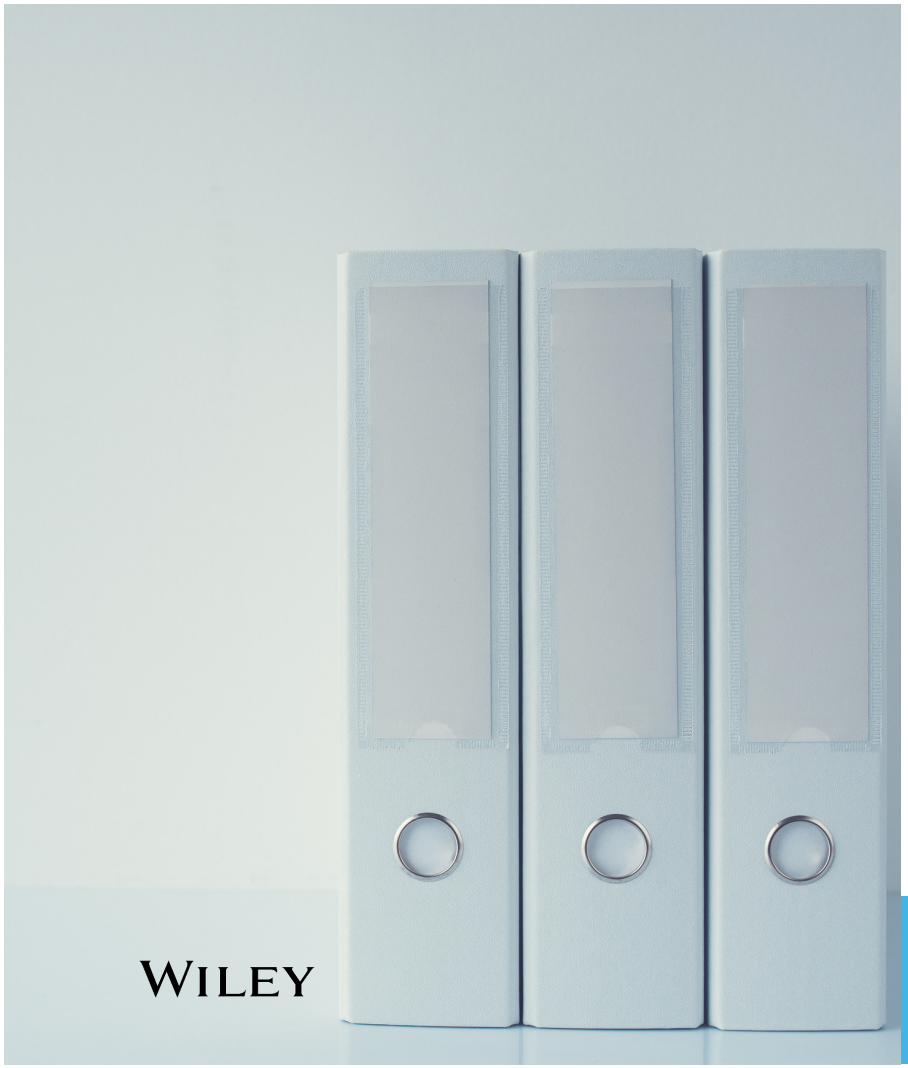
```
##Download the Minikube executable from the below link
https://minikube.sigs.k8s.io/docs/start/

##Start the Minikube Cluster on workstation
minikube start

##Kubectl commands to verify cluster inf, status and pods
kubectl cluster-info
kubectl get nodes
Kuebctl get pods -A
```

Kubernetes Objects and Terminology

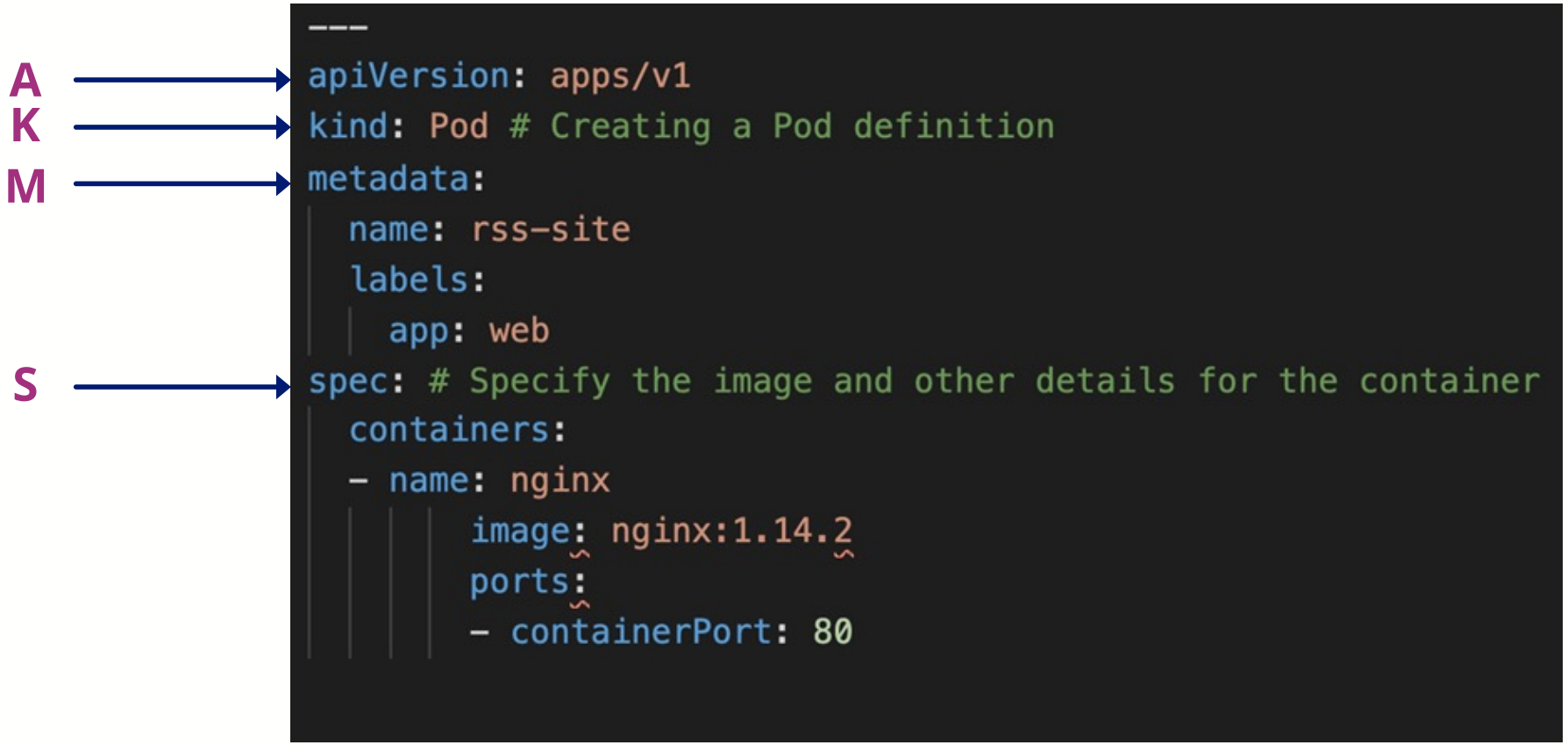
WILEY



Kubernetes Objects and Terminology

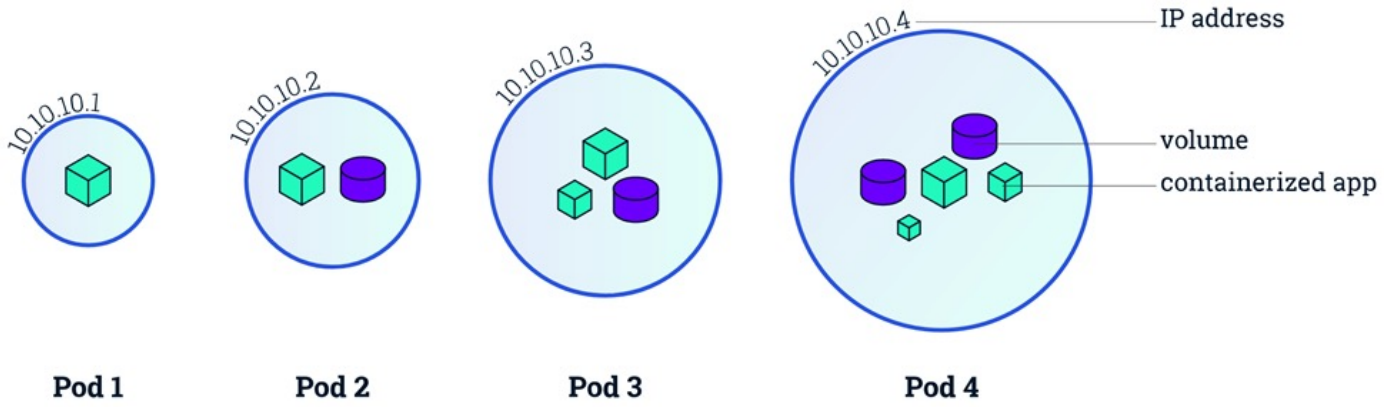
- Pods
- Nodes
- Replication controller
- Replica sets
- Deployments
- Volumes
- Services

Kubernetes YAML file Structure



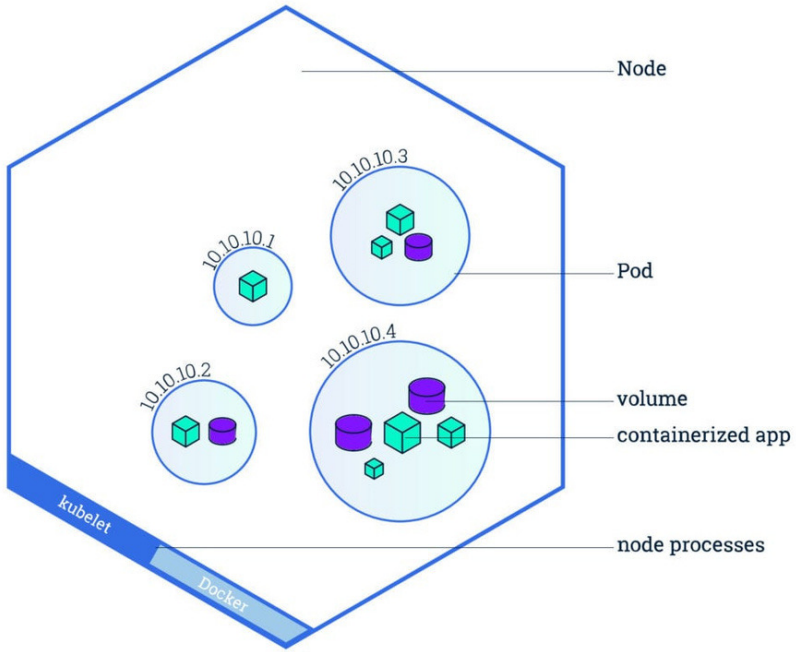
Pods

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.



https://kubernetes.io/docs/

Nodes



Source:dzone.com/articles/getting-started-with-kubernetes-clusters

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.

Replication Controllers

The replication Controller is responsible for managing the pod lifecycle. In addition, it is responsible for ensuring that the specified number of pod replicas are running at any point in time.

https://kubernetes.io/docs/

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

ReplicaSets

Replica Set ensures how many replicas of the pod should be running. It is generally considered as a replacement for a replication controller.

Both serve very similar purposes except that Replication Controller only supports equality-based selectors while ReplicaSet supports set-based selectors as well.

https://kubernetes.io/docs/

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

Kubernetes Deployment

A deployment in Kubernetes is a way to declare the desired state for your application or workload.

Deployments are basically a wrapper around ReplicaSet and provide additional functionality for a rollback in case of failures.

https://kubernetes.io/docs/

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Kubernetes Services (Networking Model)

Kubernetes Pods need Services to communicate with other pods and being accessible from within and outside the cluster network. Kubernetes networking model provides you with 3 major types of Services:

- ClusterIP – Exposes the service on a Cluster-internal IP
- NodePort – Exposes the service on each Node's IP at a static port. Accessible via ***<NodeIP>:<NodePort>***
- LoadBalancer – Exposes the service externally using a cloud provider's Load Balancer

Kubernetes Volumes (Making data Persistent)

One of the problems working with containers is that the data is ephemeral by default. So, it would be best if you found a way to persist it, and Kubernetes helps you solve this problem via "Volumes":

Example Volumes:

- awsElasticBlockStore
- azureDisk
- cephfs
- configMap
- cinder
- gcePersistentDisk

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-ebs
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-ebs
      name: test-volume
  volumes:
  - name: test-volume
    # This AWS EBS volume must already exist.
    awsElasticBlockStore:
      volumeID: "<volume id>"
      fsType: ext4
```

Demonstration
Kubernetes

WILEY

Additional Resources

https://kubernetes.io/docs/tutorials/
https://labs.play-with-k8s.com/
https://github.com/kelseyhightower/kubernetes-the-hard-way
https://www.eksworkshop.com/
https://devsecops.aksworkshop.io/

SUMMARY

- In this module, you learned:
- Introduction to Kubernetes
- Kubernetes architecture
- Installation
- Minikube setup
- Objects and terminologies