# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ✓ Installation, Configuration & Validation

- ○ Design a Kubernetes Cluster
- ○ Choose Kubernetes Infrastructure Config
- ○ Choose a Network Solution
- ○ HA Kubernetes Cluster

- ○ Provision Infrastructure
- ○ Secure Cluster Communication
- ○ Kubernetes Release Binaries
- ○ Install Kubernetes Master Nodes
- ○ Install Kubernetes Worker Nodes

- ○ TLS Bootstrapping a Node
- ○ Node end-to-end tests
- ○ Run & Analyze end-to-end tes

- ● Troubleshooting

KODEKLOUD

# DESIGN A KUBERNETES CLUSTER

# Objectives

- Node Considerations
- Resource Requirements
- Network Considerations

# Ask

- Purpose
  - Education
  - Development & Testing
  - Hosting Production Applications
- Cloud or OnPrem?
- Workloads
  - How many?
  - What kind?
    - Web
    - Big Data/Analytics
  - Application Resource Requirements
    - CPU Intensive
    - Memory Intensive
  - Traffic
    - Heavy traffic
    - Burst Traffic

KODEKLOUD

# Purpose

- Education

  - Minikube
  - Single node cluster with kubeadm/GCP/AWS

- Development & Testing

  - Multi-node cluster with a Single Master and Multiple workers
  - Setup using kubeadm tool or quick provision on GCP or AWS or AKS

- Hosting Production Applications

# Hosting Production Applications

- High Availability  Multi node cluster with multiple master nodes
- Kubeadm or GCP or Kops on AWS or other supported platforms
- Upto 5000 nodes
- Upto 150,000 PODs in the cluster
- Upto 300,000 Total Containers
- Upto 100 PODs per Node

| Nodes | GCP | | AWS | |
|-------|-----|------|-----|------|
| 1-5 | N1-standard-1 | 1 vCPU 3.75 GB | M3.medium | 1 vCPU 3.75 GB |
| 6-10 | N1-standard-2 | 2 vCPU 7.5 GB | M3.large | 2 vCPU 7.5 GB |
| 11-100 | N1-standard-4 | 4 vCPU 15 GB | M3.xlarge | 4 vCPU 15 GB |
| 101-250 | N1-standard-8 | 8 vCPU 30 GB | M3.2xlarge | 8 vCPU 30 GB |
| 251-500 | N1-standard-16 | 16 vCPU 60 GB | C4.4xlarge | 16 vCPU 30 GB |
| > 500 | N1-standard-32 | 32 vCPU 120 GB | C4.8xlarge | 36 vCPU 60 GB |

# Cloud or OnPrem?

- Use Kubeadm for on-prem
- GKE for GCP
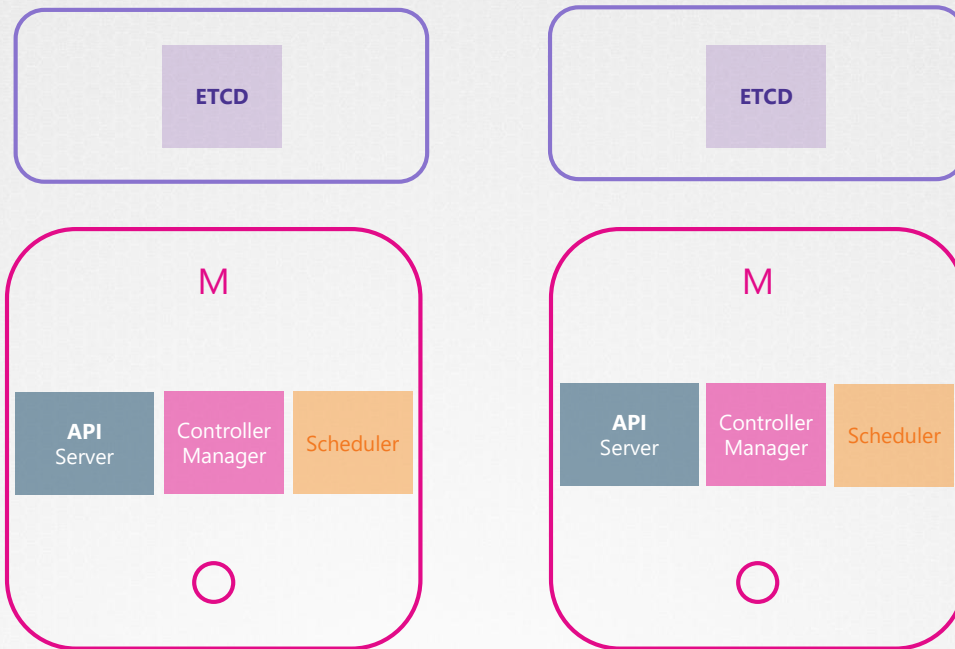- Kops for AWS
- Azure Kubernetes Service(AKS) for Azure

# Storage

- High Performance – SSD Backed Storage
- Multiple Concurrent connections – Network based storage
- Persistent shared volumes for shared access across multiple PODs
- Label nodes with specific disk types
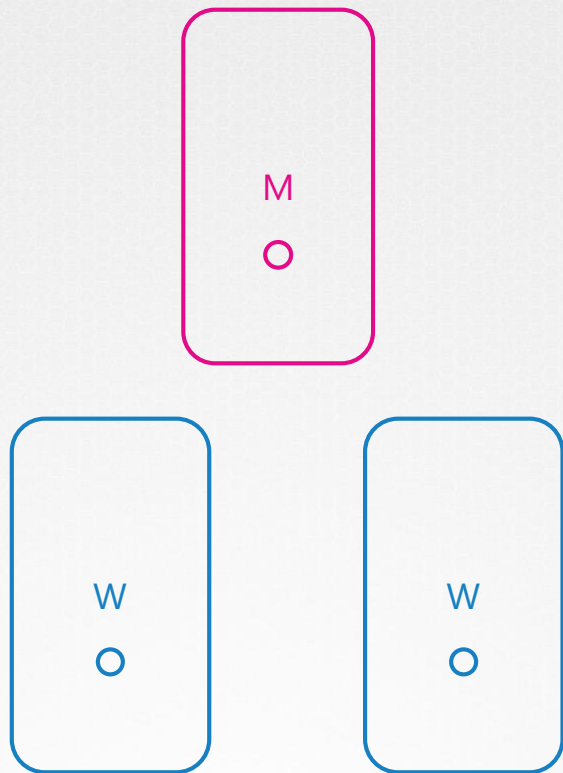- Use Node Selectors to assign applications to nodes with specific disk types

# Nodes

- Virtual or Physical Machines
- Minimum of 4 Node Cluster (Size based on workload)
- Master vs Worker Nodes
- Linux X86_64 Architecture

- Master nodes can host workloads
- Best practice is to not host workloads on Master nodes

# Master Nodes

# Our Design

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- Choose Kubernetes Infrastructure Config
- Choose a Network Solution
- HA Kubernetes Cluster

- Provision Infrastructure
- Secure Cluster Communication
- Kubernetes Release Binaries
- Install Kubernetes Master Nodes
- Install Kubernetes Worker Nodes

- TLS Bootstrapping a Node
- Node end-to-end tests
- Run & Analyze end-to-end tes

- Troubleshooting

KODEKLOUD

# Choosing Kubernetes Infrastructure
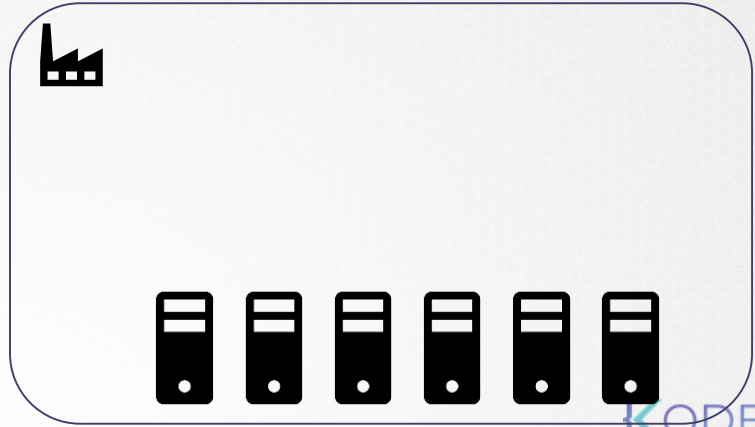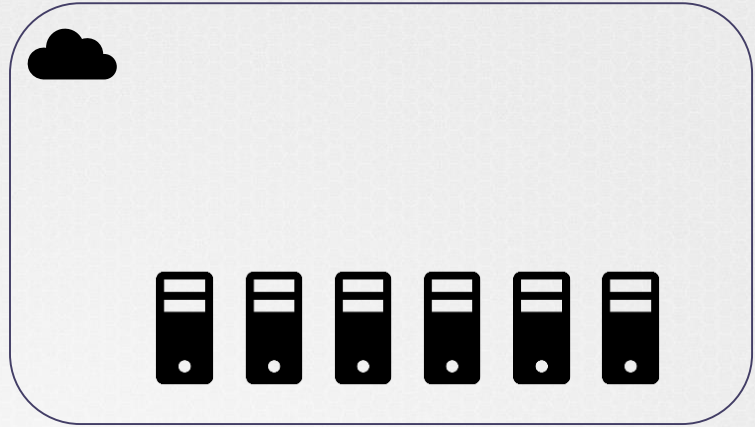
Minikube

KUBEADM

Deploys VMs

Requires VMs to be ready

Singe Node Cluster

Singe/Multi Node Cluster
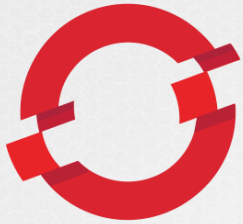
## Turnkey Solutions

- You Provision VMs
- You Configure VMs
- You Use Scripts to Deploy Cluster
- You Maintain VMs yourself
- Eg: Kubernetes on AWS using KOPS

## Hosted Solutions

(Managed Solutions)

- Kubernetes-As-A-Service
- Provider provisions VMs
- Provider installs Kubernetes
- Provider maintains VMs
- Eg: Google Container Engine (GKE)

# Turnkey Solutions

**OpenShift**

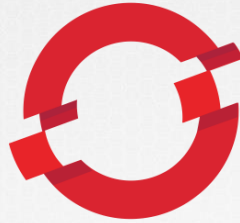**Cloud Foundry Container Runtime**

**VMware Cloud PKS**

**Vagrant**

# Hosted Solutions

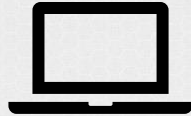Google Container
Engine **(GKE)**

OpenShift
Online

Azure Kubernetes
Service

Amazon Elastic
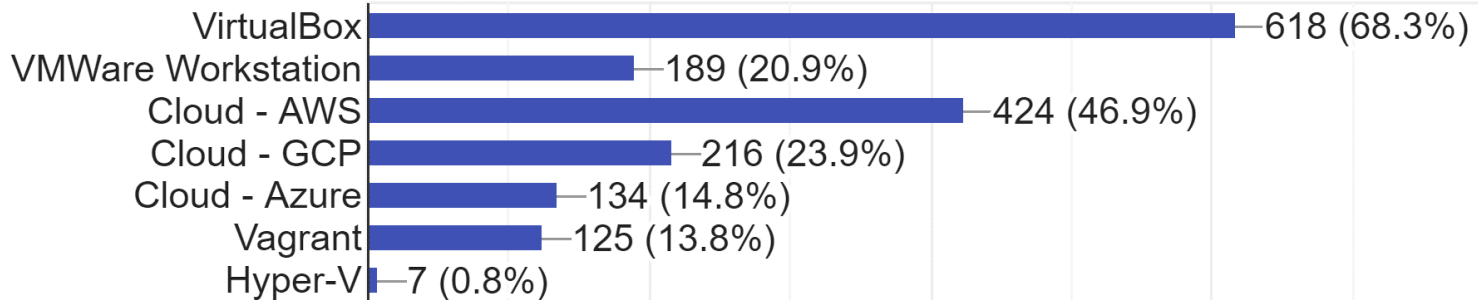Container Service
for Kubernetes
(EKS)

# Our Choice

## Preferred Virtualization technology for Labs?

905 responses

| Technology | Responses |
|------------|-----------|
| VirtualBox | 618 (68.3%) |
| VMWare Workstation | 189 (20.9%) |
| Cloud - AWS | 424 (46.9%) |
| Cloud - GCP | 216 (23.9%) |
| Cloud - Azure | 134 (14.8%) |
| Vagrant | 125 (13.8%) |
| Hyper-V | 7 (0.8%) |

KODEKLOUD

# Our Design

M ○

W ○    W ○
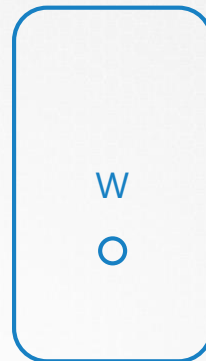
KODE KLOUD

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ● Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Config
- Choose a Network Solution
- ○ HA Kubernetes Cluster
- ● Troubleshooting

- ○ Provision Infrastructure
- ○ Secure Cluster Communication
- ○ Kubernetes Release Binaries
- ○ Install Kubernetes Master Nodes
- ○ Install Kubernetes Worker Nodes

- ○ TLS Bootstrapping a Node
- ○ Node end-to-end tests
- ○ Run & Analyze end-to-end tes

KODEKLOUD

# Chose a Networking Solution

CISCO  weaveworks  flannel  cilium  vmware NSX

GCE  nuagenetworks

KODEKLOUD

# Our Design

POD CIDR: 10.32.0.0/12
Service CIDR: 10.96.0.0/24

M

W

W
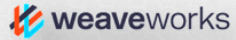
# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ● Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Config
- ✓ Choose a Network Solution
- HA Kubernetes Cluster
- Troubleshooting

- ○ Provision Infrastructure
- ○ Secure Cluster Communication
- ○ Kubernetes Release Binaries
- ○ Install Kubernetes Master Nodes
- ○ Install Kubernetes Worker Nodes

- ○ TLS Bootstrapping a Node
- ○ Node end-to-end tests
- ○ Run & Analyze end-to-end tes

KODEKLOUD

# HA Kubernetes Cluster

KODEKLOUD

Kube-controller-manager
Endpoint

🔒 master1

Active

Standby

Controller Manager

Scheduler

M

API Server

ETCD

Controller Manager

Scheduler

M

API Server

ETCD
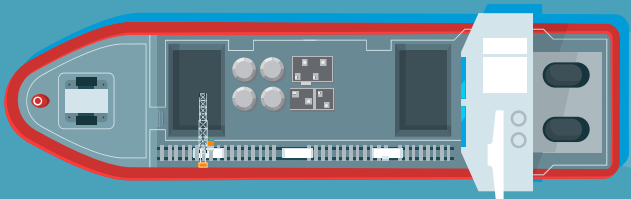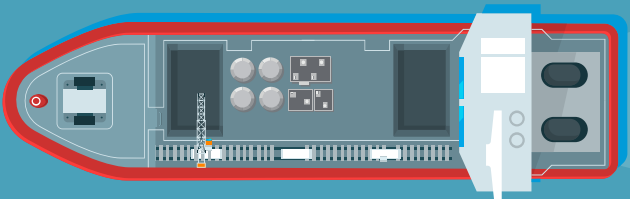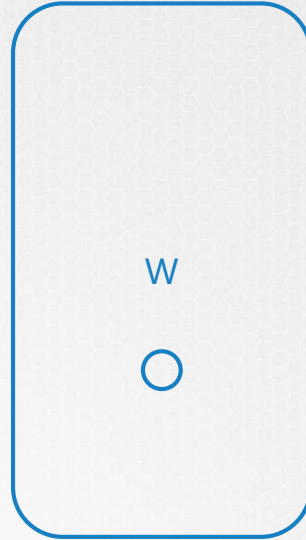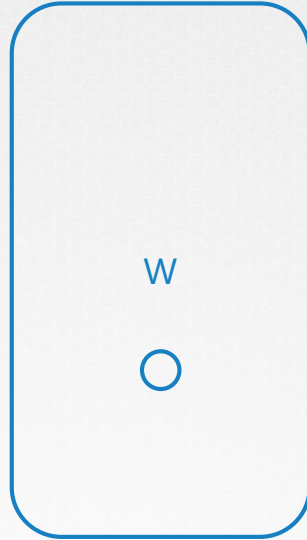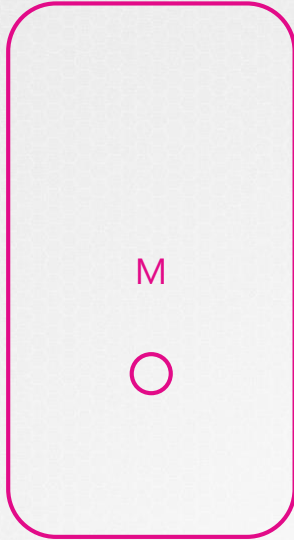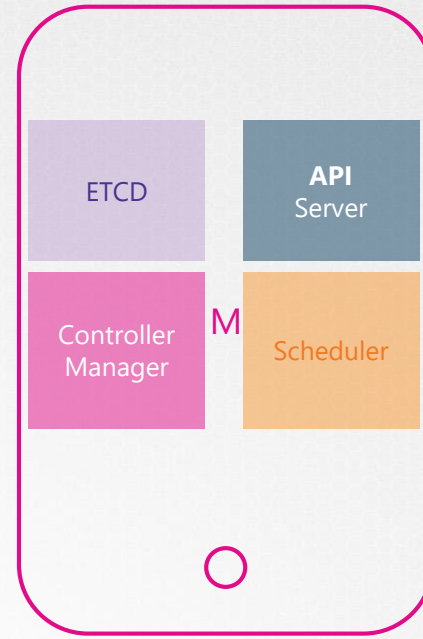
```
kube-controller-manager --leader-elect true  [other options]
                --leader-elect-lease-duration 15s
                --leader-elect-renew-deadline 10s
                --leader-elect-retry-period 2s
```

# Stacked Topology
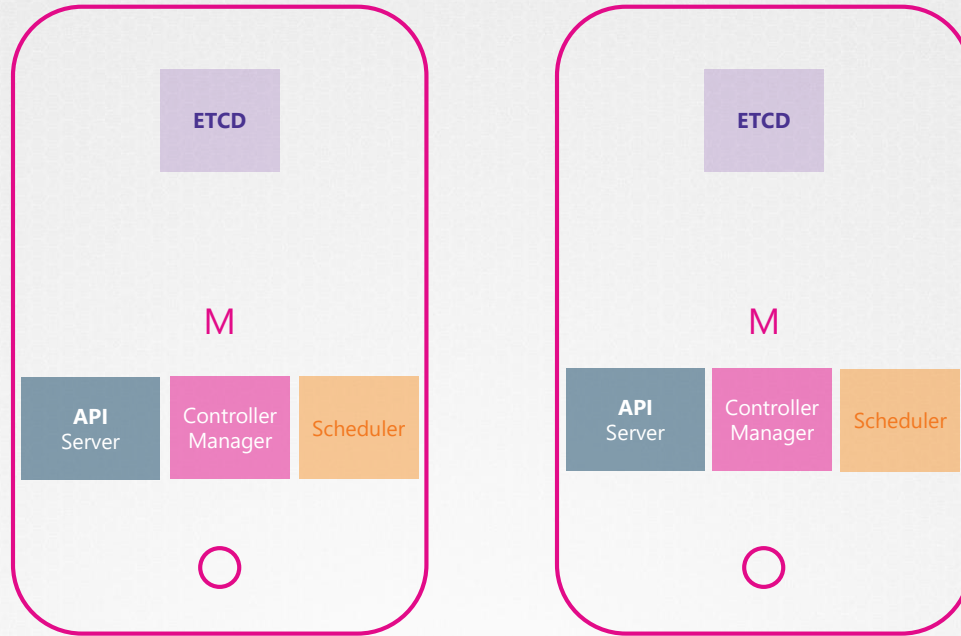


- ✓ Easier to setup
- ✓ Easier to manage
- ✓ Fewer Servers
- ❖ Risk during failures

# External ETCD Topology

ETCD

ETCD

M

M

| API Server | Controller Manager | Scheduler |
|---|---|---|

| API Server | Controller Manager | Scheduler |
|---|---|---|
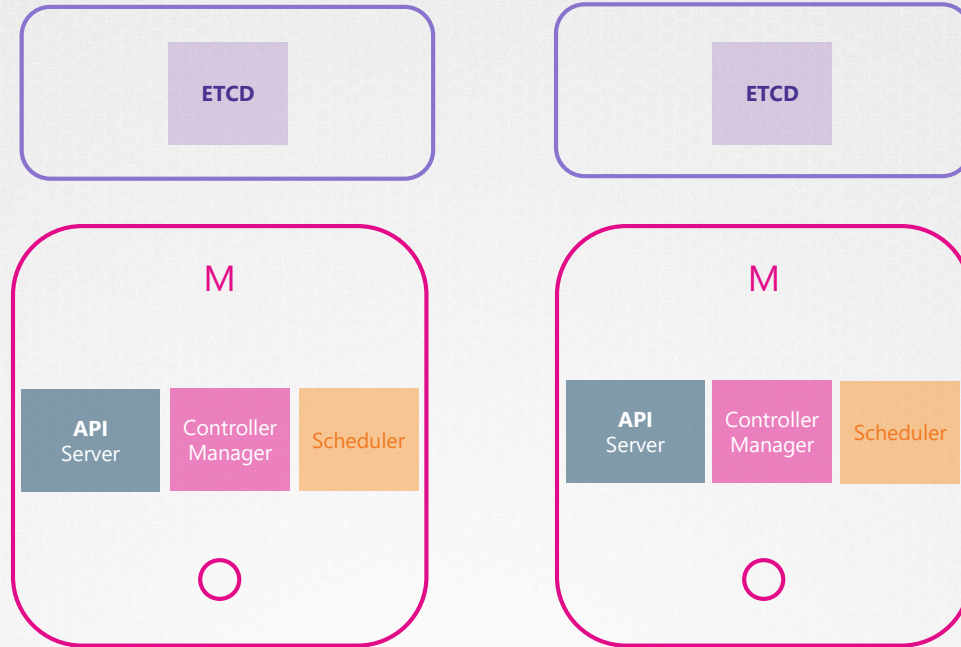
✓ Less Risky
❖ Harder to Setup
❖ More Servers

```
cat /etc/systemd/system/kube-apiserver.service
```

```
[Service]
ExecStart=/usr/local/bin/kube-apiserver \\
   --advertise-address=${INTERNAL_IP} \\
   --allow-privileged=true \\
   --apiserver-count=3 \\
   --etcd-cafile=/var/lib/kubernetes/ca.pem \\
   --etcd-certfile=/var/lib/kubernetes/kubernetes.pem \\
   --etcd-keyfile=/var/lib/kubernetes/kubernetes-key.pem \\
   --etcd-servers=https://10.240.0.10:2379,https://10.240.0.11:2379
```
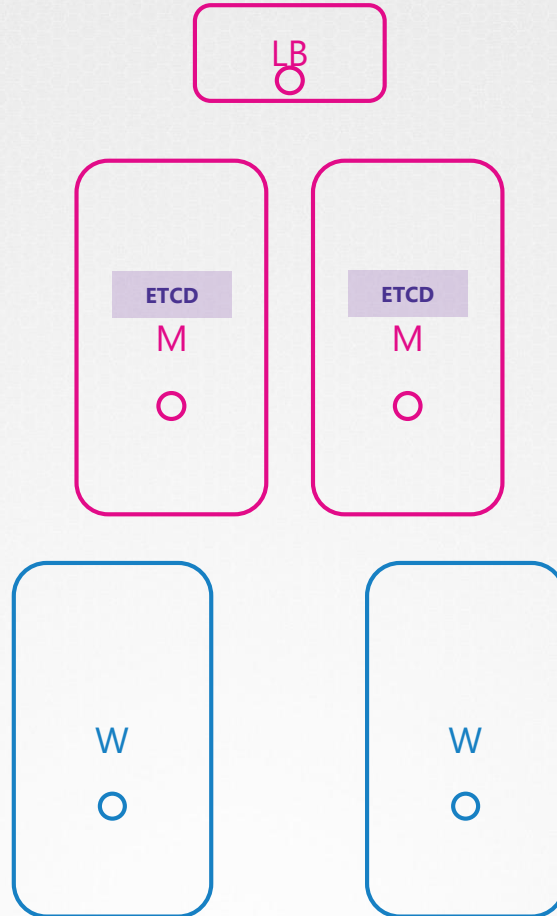


ETCD

ETCD

M

M

API
Server

Controller
Manager

Scheduler

API
Server

Controller
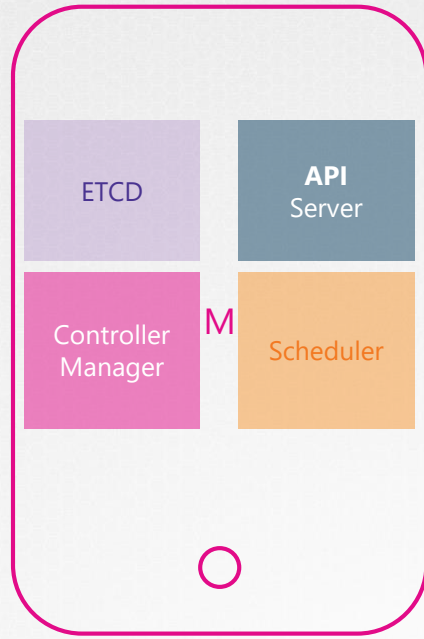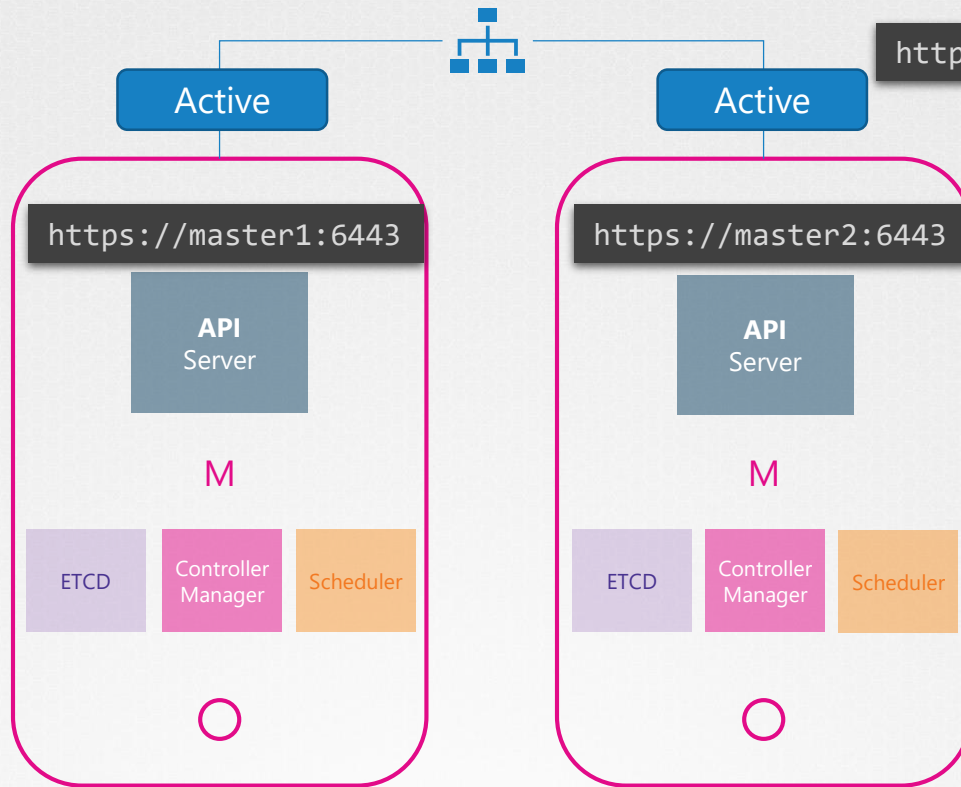Manager

Scheduler

# Our Design

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ✓ Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Config
- ✓ Choose a Network Solution
- HA Kubernetes Cluster
- Troubleshooting

- ○ Provision Infrastructure
- ○ Secure Cluster Communication
- ○ Kubernetes Release Binaries
- ○ Install Kubernetes Master Nodes
- ○ Install Kubernetes Worker Nodes

- ○ TLS Bootstrapping a Node
- ○ Node end-to-end tests
- ○ Run & Analyze end-to-end tes

KODEKLOUD

# ETCD

In HA

# Objectives

- What is ETCD?
- What is a Key-Value Store?
- How to get started quickly?
- How to operate ETCD?
- What is a distributed system?
- How ETCD Operates
- RAFT Protocol
- Best practices on number of nodes

**ETCD** is a distributed reliable key-value store that is Simple, Secure & Fast

# key-value store

Tabular/Relational Databases

| Name | Age | Location | Salary | Grade |
|------|-----|----------|--------|-------|
| John Doe | 45 | New York | 5000 | |
| Dave Smith | 34 | New York | 4000 | |
| Aryan Kumar | 10 | New York | | A |
| Lauren Rob | 13 | Bangalore | | C |
| Lily Oliver | 15 | Bangalore | | B |

KODEKLOUD

# key-value store

| Key | Value |
| --- | --- |
| Name | John Doe |
| Age | 45 |
| Location | New York |
| Salary | 5000 |

| Key | Value |
| --- | --- |
| Name | Dave Smith |
| Age | 34 |
| Location | New York |
| Salary | 4000 |

| Key | Value |
| --- | --- |
| Name | Aryan Kumar |
| Age | 10 |
| Location | New York |
| Grade | A |

| Key | Value |
| --- | --- |
| Name | Lauren Rob |
| Age | 13 |
| Location | Bangalore |
| Grade | C |

| Key | Value |
| --- | --- |
| Name | Lily Oliver |
| Age | 15 |
| Location | Bangalore |
| Grade | B |

# key-value store

```
{
  "name": "John Doe",
  "age": 45,
  "location": "New York",
  "salary": 5000
}
```

```
{
  "name": "Dave Smith",
  "age": 34,
  "location": "New York",
  "salary": 4000,
  "organization": "ACME"
}
```

```
{
  "name": "Aryan Kumar",
  "age": 10,
  "location": "New York",
  "Grade": "A"
}
```

```
{
  "name": "Lily Oliver",
  "age": 15,
  "location": "Bangalore",
  "Grade": "B"
}
```

```
{
  "name": "Lauren Rob",
  "age": 13,
  "location": "Bangalore",
  "Grade": "C"
}
```

KODEKLOUD

**ETCD** is a distributed reliable key-value store that is Simple, Secure & Fast

# distributed

# Consistent
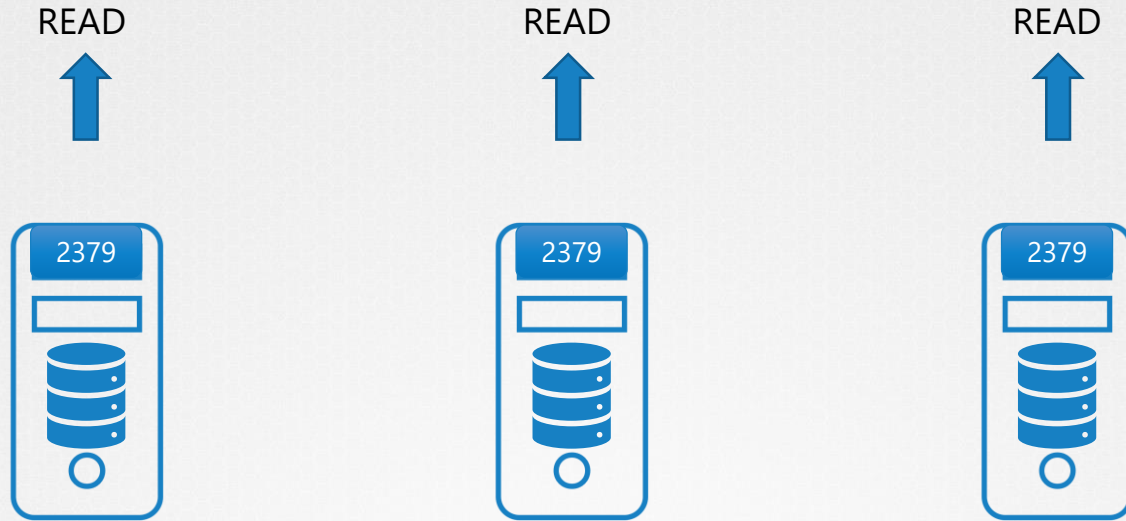
# Leader Election - RAFT
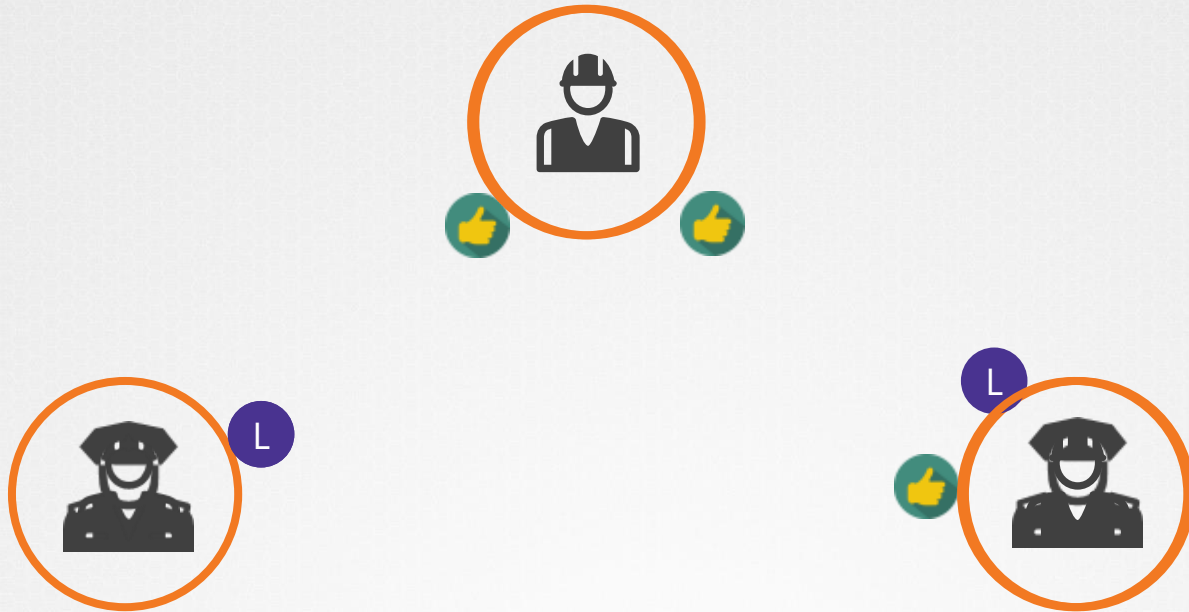
| Instances | Quorum | Fault Tolerance |
|-----------|--------|-----------------|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 3 | 2 |
| 6 | 4 | 2 |
| 7 | 4 | 3 |

Majority = N/2 + 1

Quorum of 2 = 2/2 + 1 = 2

Quorum of 3 = 3/2 + 1 = 2.5 ~= 2

Quorum of 5 = 5/2 + 1 = 3.5 ~= 3

# Odd or even?

| Managers | Majority | Fault Tolerance |
|----------|----------|-----------------|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 3 | 2 |
| 6 | 4 | 2 |
| 7 | 4 | 3 |



Quorum = 4

Quorum - 4
Quorum - 4

Quorum - 4
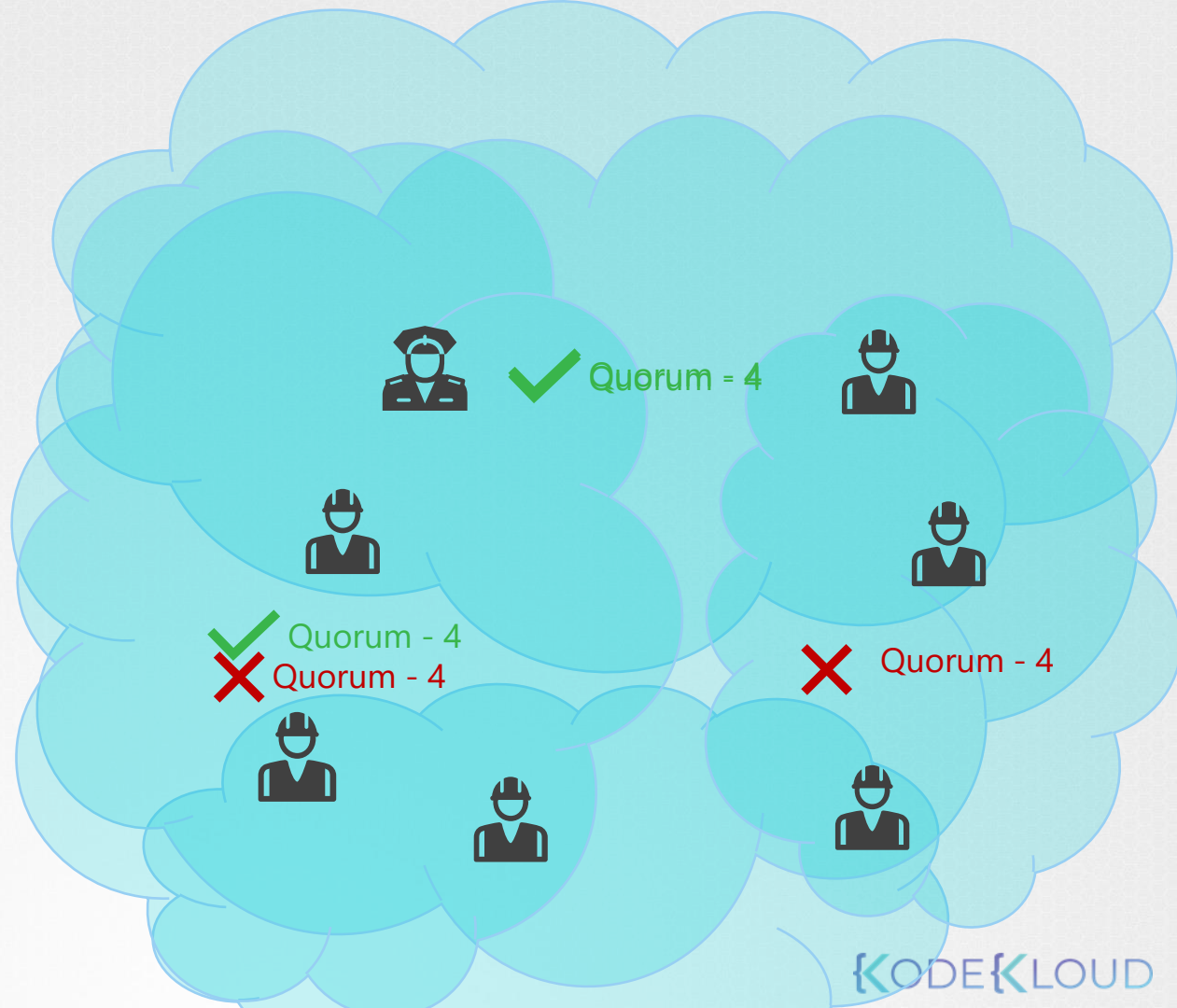
# Getting Started

```
wget -q --https-only \
    "https://github.com/coreos/etcd/releases/download/v3.3.9/etcd-v3.3.9-linux-amd64.tar.gz"
```

```
tar -xvf etcd-v3.3.9-linux-amd64.tar.gz
```

```
mv etcd-v3.3.9-linux-amd64/etcd* /usr/local/bin/
```

```
mkdir -p /etc/etcd /var/lib/etcd
```

```
cp ca.pem kubernetes-key.pem kubernetes.pem /etc/etcd/
```

```
ExecStart=/usr/local/bin/etcd \\
  --name ${ETCD_NAME} \\
  --cert-file=/etc/etcd/kubernetes.pem \\
  --key-file=/etc/etcd/kubernetes-key.pem \\
  --peer-cert-file=/etc/etcd/kubernetes.pem \\
  --peer-key-file=/etc/etcd/kubernetes-key.pem \\
  --trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-client-cert-auth \\
  --client-cert-auth \\
  --initial-advertise-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-client-urls https://${INTERNAL_IP}:2379,https://127.0.0.1:2379 \\
  --advertise-client-urls https://${INTERNAL_IP}:2379 \\
  --initial-cluster-token etcd-cluster-0 \\
  --initial-cluster peer-1=https://${PEER1_IP}:2380,peer-2=https://${PEER2_IP}:2380 \\
  --initial-cluster-state new \\
  --data-dir=/var/lib/etcd
```

KODEKLOUD

# ETCDCTL

```
export ETCDCTL_API=3
```

```
etcdctl put name john
```

```
etcdctl get name
```
```
name
john
```

```
etcdctl get / --prefix --keys-only
```
```
name
```

# Number of Nodes

| Instances | Quorum | Fault Tolerance |
|-----------|--------|-----------------|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 3 | 2 |
| 6 | 4 | 2 |
| 7 | 4 | 3 |

# Our Design

# DEMO
Pre-Requisites

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ● Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Config
- ✓ Choose a Network Solution
- ✓ HA Kubernetes Cluster

- ○ Provision Infrastructure
- ○ Secure Cluster Communication
- ○ Kubernetes Release Binaries
- ○ Install Kubernetes Master Nodes
- ○ Install Kubernetes Worker Nodes

- ○ TLS Bootstrapping a Node
- ○ Node end-to-end tests
- ○ Run & Analyze end-to-end tes

- Troubleshooting

KODEKLOUD

# Provision Infrastructure

# Our Design

VAGRANT

VAGRANT

```
▶   vagrant up
```

- Deploys 5 VMs - 2 Master, 2 Worker and 1 Loadbalancer with the name 'kubernetes-ha-* '
- Set's IP addresses in the range 192.168.5
- Add's a DNS entry to each of the nodes to access internet
- Install's Docker on the nodes

# DEMO
## Provision Infrastructure

# DEMO
## Install Client Tools

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- 🟢 Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Config
- ✓ Choose a Network Solution
- ✓ HA Kubernetes Cluster

- ✓ Provision Infrastructure
- Secure Cluster Communication
- ◯ Kubernetes Release Binaries
- ◯ Install Kubernetes Master Nodes
- ◯ Install Kubernetes Worker Nodes

- ◯ TLS Bootstrapping a Node
- ◯ Node end-to-end tests
- ◯ Run & Analyze end-to-end tes

- Troubleshooting

# DEMO
## Secure Cluster Communication

# DEMO
## Kube Config Files
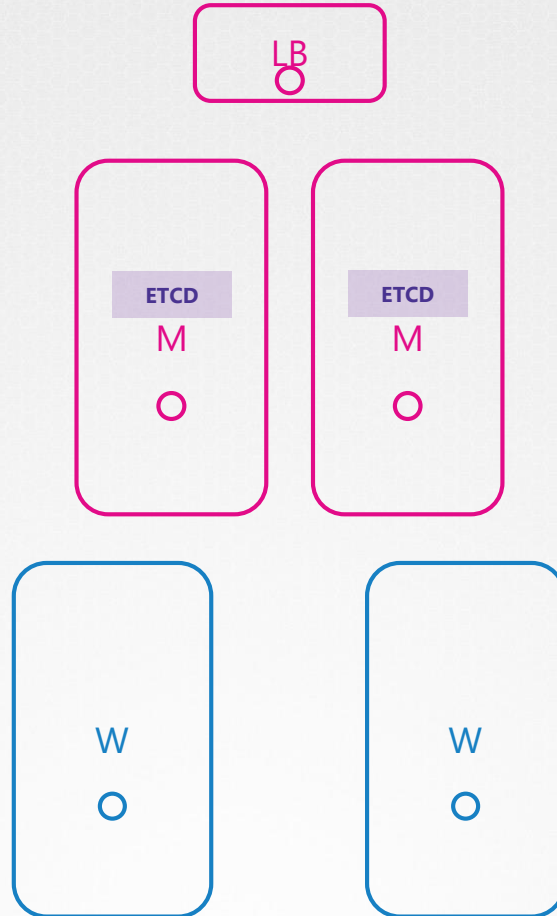
# DEMO
## Data Encryption

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ● Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Confi...
- ✓ Choose a Network Solution
- ✓ HA Kubernetes Cluster

- ✓ Provision Infrastructure
- ✓ Secure Cluster Communication
- ○ Kubernetes Release Binaries
- ○ Install Kubernetes Master Nodes
- ○ Install Kubernetes Worker Nodes

- ○ TLS Bootstrapping a Node
- ○ Node end-to-end tests
- ○ Run & Analyze end-to-end tes...

- ● Troubleshooting

# Kubernetes Release Binaries

<> **Code**   ⊙ Issues **2,151**   ⑁ Pull requests **1,007**   ▥ Projects **11**   ⅲ Insights

## Production-Grade Container Scheduling and Management   https://kubernetes.io

kubernetes   go   cncf   containers

⊙ **76,018** commits     ⑁ **40** branches     ⬯ **489** releases     ⚏ **2,048** contributors     ⚖ Apache-2.0

Branch: master ▾     New pull request          Create new file   Upload files   Find File   Clone or download ▾

🤖 **k8s-ci-robot** Merge pull request #75308 from msau42/csi-beta-e2e   ⋯          Latest commit 517922f 3 hours ago

| | | |
|---|---|---|
| 📁 .github | delete all duplicate empty blanks | 19 days ago |
| 📁 Godeps | Update to go 1.12 version of golang.org/x/net | a day ago |
| 📁 api | update openapi-spec | a day ago |
| 📁 build | Merge pull request #74799 from lubinsz/pr_bazel | 9 hours ago |

# v1.13.3

▣ **k8s-release-robot** released this on Feb 2 · **64 commits** to release-1.13 since this release

See **kubernetes-announce@** and **CHANGELOG-1.13.md** for details.

SHA512 for `kubernetes.tar.gz` :

```
151af896b72c7fd09c05da1a7685e8b2f167c717adbe5776f80a264171e5f3359a948af93642856e0bfbabb49d3bf9c274085ea
cf6109c4b972ba5bc9d24b8a7
```

Additional binary downloads are linked in the **CHANGELOG-1.13.md**.

─────────────────────────────────────────────

▼ **Assets** 3

| 📦 **kubernetes.tar.gz** | 1.87 MB |
|---|---|
| 🗎 **Source code** (zip) | |
| 🗎 **Source code** (tar.gz) | |

```
▶  wget https://github.com/kubernetes/kubernetes/releases/download/v1.13.3/kubernetes.tar.gz

kubernetes.tar.gz


▶  tar -xzvf kubernetes.tar.gz

kubernetes


▶ cd kubernetes; ls

client   cluster   docs   hack   LICENSES   platforms   README.md
server   version


▶  cluster/get-kube-binaries.sh

client/kubernetes-client-linux-amd64.tar.gz
server/kubernetes-server-linux-amd64.tar.gz

Extracting /root/kubernetes/client/kubernetes-client-linux-amd64.tar.gz into /root/kubernetes/platforms/linux/amd64
Add '/root/kubernetes/client/bin' to your PATH to use newly-installed binaries.


▶  cd server; tar -zxvf kubernetes-server-linux-amd64.tar.gz

Kubernetes/server
```

```
cluster/get-kube-binaries.sh

client/kubernetes-client-linux-amd64.tar.gz
server/kubernetes-server-linux-amd64.tar.gz

Extracting /root/kubernetes/client/kubernetes-client-linux-amd64.tar.gz into /root/kubernetes/platforms/linux/amd64
Add '/root/kubernetes/client/bin' to your PATH to use newly-installed binaries.
```

```
cd server; tar -zxvf kubernetes-server-linux-amd64.tar.gz

Kubernetes/server
```

```
ls kubernetes/server/bin

apiextensions-apiserver        kubeadm                      kube-proxy.docker_tag           mounter
cloud-controller-manager       kube-apiserver               kube-controller-manager.tar     kube-proxy.tar
kubectl                        kube-scheduler               cloud-controller-manager.tar    kube-apiserver.tar
kubelet                        kube-scheduler.docker_tag
hyperkube                      kube-controller-manager      kube-proxy                      kube-scheduler.tar
```

❈ **kubernetes**                                                                                           Documen

# Setup

HOME   **SETUP**   CONCEPTS   TASKS   TUTORIALS   REFERENCE   CONTRIBUTE

**Setup**

Picking the Right Solution

▼ Downloading Kubernetes

    Building a release

    v1.13 Release Notes

▶ Bootstrapping Clusters with kubeadm

▶ Turnkey Cloud Solutions

▶ Custom Cloud Solutions

▶ On-Premises VMs

Kubernetes Version and Version Skew
Support Policy

Building Large Clusters

Running in Multiple Zones

CRI installation

Installing Kubernetes with Digital Rebar
Provision (DRP) via KRIB

PKI Certificates and Requirements

Running Kubernetes Locally via Minikube

Validate Node Setup

# v1.13 Release Notes

**v1.13.0**
- **Downloads for v1.13.0**
  - **Client Binaries**
  - **Server Binaries**
  - **Node Binaries**
- **Kubernetes 1.13 Release Notes**
- **Security Content**
- **Urgent Upgrade Notes**
  - **(No, really, you MUST do this before you upgrade)**
- **Known Issues**
- **Deprecations**
- **Major Themes**
  - **SIG API Machinery**
  - **SIG Auth**
  - **SIG AWS**
  - **SIG Azure**
  - **SIG Big Data**
  - **SIG CLI**
  - **SIG Cloud Provider**
  - **SIG Cluster Lifecycle**
  - **SIG IBM Cloud**

**K**LOUD

# External Dependencies

- Default etcd server is unchanged at v3.2.24 since Kubernetes 1.12. (#68318)

- The list of validated docker versions remain unchanged at 1.11.1, 1.12.1, 1.13.1, 17.03, 17.06, 17.09, 18.06 since Kubernetes 1.12. (#68495)

- The default Go version was updated to 1.11.2. (#70665)

- The minimum supported Go version was updated to 1.11.2 (#69386)

- CNI is unchanged at v0.6.0 since Kubernetes 1.10 (#51250)

- CSI is updated to 1.0.0. Pre-1.0.0 API support is now deprecated. (#71020])

- The dashboard add-on has been updated to v1.10.0. (#68450)

- Heapster remains at v1.6.0-beta, but is now retired in Kubernetes 1.13 (#67074)

- Cluster Autoscaler has been upgraded to v1.13.0 (#71513)

- kube-dns is unchanged at v1.14.13 since Kubernetes 1.12 (#68900)

- Influxdb is unchanged at v1.3.3 since Kubernetes 1.10 (#53319)

- Grafana is unchanged at v4.4.3 since Kubernetes 1.10 (#53319)

- Kibana has been upgraded to v6.3.2. (#67582)

- CAdvisor has been updated to v0.32.0 (#70964)

- fluentd-gcp-scaler has been updated to v0.5.0 (#68837)

- Fluentd in fluentd-elasticsearch is unchanged at v1.2.4 since Kubernetes 1.11 (#67434)

# DEMO
## Download Release Binaries

# Course Objectives

- Core Concepts
- Scheduling
- Logging Monitoring
- Application Lifecycle Management
- Cluster Maintenance
- Security
- Storage
- Networking
- Installation, Configuration & Validation

- Design a Kubernetes Cluster
- Choose Kubernetes Infrastructure Config
- Choose a Network Solution
- HA Kubernetes Cluster
- Troubleshooting

- Provision Infrastructure
- Secure Cluster Communication
- Kubernetes Release Binaries
- Install Kubernetes Master Nodes
- Install Kubernetes Worker Nodes

- TLS Bootstrapping a Node
- Node end-to-end tests
- Run & Analyze end-to-end tes

KODEKLOUD

# Install Master

# Our Design

HAPRoxy

LB

| | |
|---|---|
| **API** Server | **API** Server |
| **ETCD** | **ETCD** |
| M | M |
| Controller Manager | Controller Manager |
| Scheduler | Scheduler |

☐ Deploy ETCD Cluster
☐ Deploy Control Plane Components
☐ Network Loadbalancer

W

W

weaveworks

# DEMO
Install ETCD Cluster

# DEMO
## Install Control-plane Components

# DEMO
## Install Load Balancer

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ● Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Config
- ✓ Choose a Network Solution
- ✓ HA Kubernetes Cluster
- ● Troubleshooting

- ✓ Provision Infrastructure
- ✓ Secure Cluster Communication
- ✓ Kubernetes Release Binaries
- ✓ Install Kubernetes Master Nodes
- Install Kubernetes Worker Nodes

- ○ TLS Bootstrapping a Node
- ○ Node end-to-end tests
- ○ Run & Analyze end-to-end tes

# Install Worker Nodes

# Our Design

HAPRoxy

LB

| API Server | | API Server |
| ETCD | | ETCD |
| M | | M |
| Controller Manager | | Controller Manager |
| Scheduler | | Scheduler |

✓ Deploy ETCD Cluster
✓ Deploy Control Plane Components
✓ Network Loadbalancer

| kubelet | | kubelet |
| Kube-proxy | | Kube-proxy |
| W | | W |

□ Generate CERTs for Worker-1
□ Configure Kubelet for Worker-1
□ Renew Certificates
□ Configure kube-proxy

**TLS Bootstrap:**
□ Worker-2 to create and configure certificates by itself
□ Configure Kubelet for Worker-2
□ Worker-2 to renew certificates by itself
□ Configure kube-proxy

weaveworks

KODEKLOUD

# DEMO
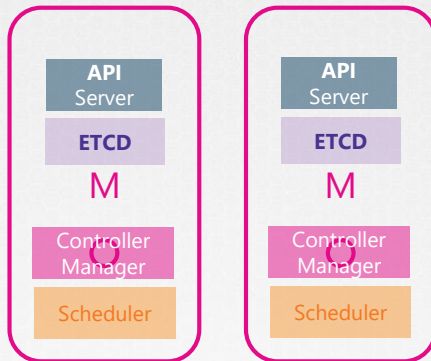Install Worker-1

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- Installation, Configuration & Validation
  - ✓ Design a Kubernetes Cluster
  - ✓ Choose Kubernetes Infrastructure Config
  - ✓ Choose a Network Solution
  - ✓ HA Kubernetes Cluster

  - ✓ Provision Infrastructure
  - ✓ Secure Cluster Communication
  - ✓ Kubernetes Release Binaries
  - ✓ Install Kubernetes Master Nodes
  - ✓ Install Kubernetes Worker Nodes

  - ✓ TLS Bootstrapping a Node
  - ○ Node end-to-end tests
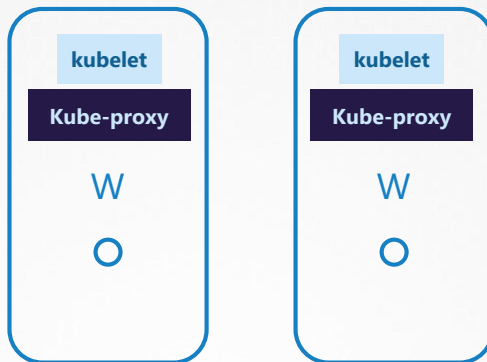  - ○ Run & Analyze end-to-end tes

- Troubleshooting

# TLS Bootstrap Kubelet

# Our Design



HAPRoxy

LB

API Server

ETCD

M

Controller Manager

Scheduler

API Server

ETCD

M

Controller Manager

Scheduler

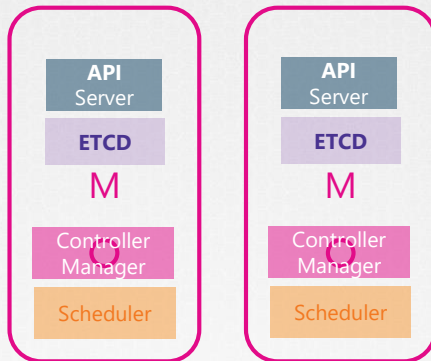✓ Deploy ETCD Cluster
✓ Deploy Control Plane Components
✓ Network Loadbalancer

kubelet

Kube-proxy

W

W

✓ Generate CERTs for Worker-1
✓ Configure Kubelet for Worker-1
✓ Renew Certificates
✓ Configure kube-proxy

**TLS Bootstrap:**
❑ Worker-2 to create and configure certificates by itself
❑ Configure Kubelet for Worker-2
❑ Worker-2 to renew certificates by itself
❑ Configure kube-proxy

weaveworks

Server Certs

M

W

Client Certs
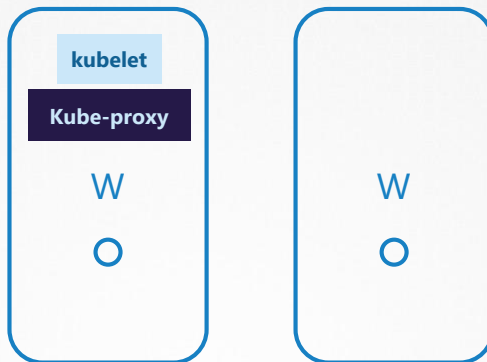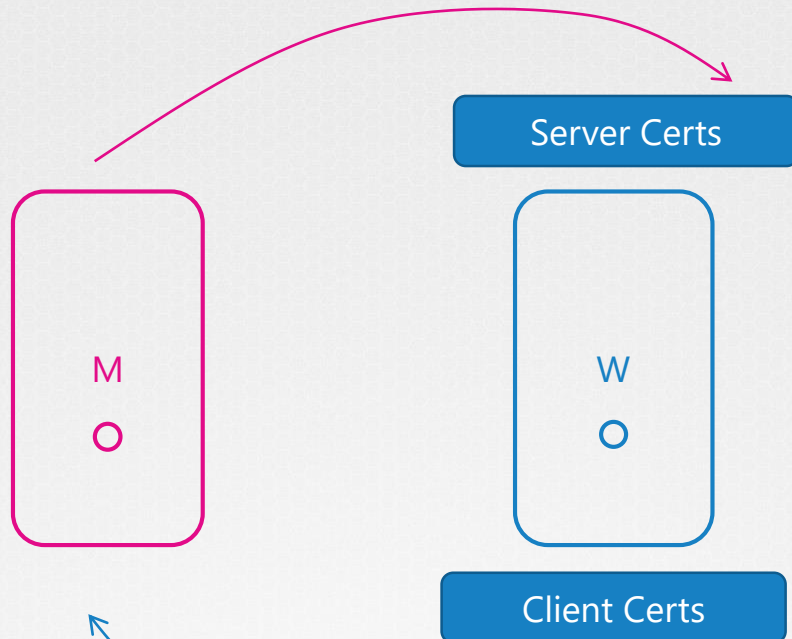
**kubelet.service**

```
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --image-pull-progress-deadline=2m \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --tls-cert-file=/var/lib/kubelet/worker-1.crt \\
  --tls-private-key-file=/var/lib/kubelet/worker-1.key \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2
```

```
kubectl get csr
```
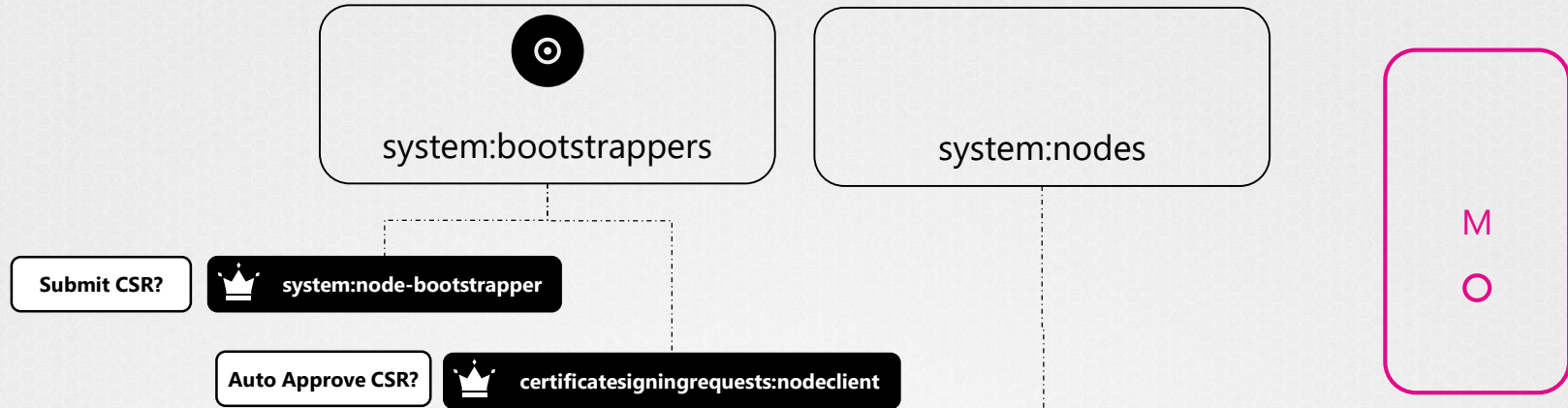
| NAME | AGE | REQUESTOR | CONDITION |
|------|-----|-----------|-----------|
| node-csr-VnzfkE6WdOMOna_S7jIuMTtQzu1-utgAA5gbk3dooUY | 13m | system:bootstrap:07401b | Approved,Issued |

1. Create Bootstrap Token and associate it to group **system:bootstrappers**
2. Assign Role – **system:node-bootstrapper** to group **system:bootstrappers**
3. Assign Role - **system:certificates.k8s.io:certificatesigningrequests:nodeclient** to group **system:bootstrappers**
4. Assign Role - **system:certificates.k8s.io:certificatesigningrequests:selfnodeclient** to group **system:nodes**

## bootstrap-kubeconfig

```yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /var/lib/kubernetes/ca.crt
    server: https://192.168.5.30:6443
  name: bootstrap
contexts:
- context:
    cluster: bootstrap
    user: kubelet-bootstrap
  name: bootstrap
current-context: bootstrap
kind: Config
preferences: {}
users:
- name: kubelet-bootstrap
  user:
    token: 07401b.f395accd246ae52d
```

W
○

## kubelet.service

```
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --image-pull-progress-deadline=2m \\
  --bootstrap-kubeconfig="/var/lib/kubelet/bootstrap-kubeconfig" \\
  --tls-cert-file=/var/lib/kubelet/worker-1.crt \\
  --tls-private-key-file=/var/lib/kubelet/worker-1.key \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2
```

## bootstrap-kubeconfig

```yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /var/lib/kubernetes/ca.crt
    server: https://192.168.5.30:6443
  name: bootstrap
contexts:
- context:
    cluster: bootstrap
    user: kubelet-bootstrap
  name: bootstrap
current-context: bootstrap
kind: Config
preferences: {}
users:
- name: kubelet-bootstrap
  user:
    token: 07401b.f395accd246ae52d
```

W

○

## kubelet.service

```
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --image-pull-progress-deadline=2m \\
  --bootstrap-kubeconfig="/var/lib/kubelet/bootstrap-kubeconfig" \\
  --rotate-certificates=true \\
  --tls-cert-file=/var/lib/kubelet/worker-1.crt \\
  --tls-private-key-file=/var/lib/kubelet/worker-1.key \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2
```

## bootstrap-kubeconfig

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /var/lib/kubernetes/ca.crt
    server: https://192.168.5.30:6443
  name: bootstrap
contexts:
- context:
    cluster: bootstrap
    user: kubelet-bootstrap
  name: bootstrap
current-context: bootstrap
kind: Config
preferences: {}
users:
- name: kubelet-bootstrap
  user:
    token: 07401b.f395accd246ae52d
```

W

O

## Client Certs

## kubelet.service

```
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --image-pull-progress-deadline=2m \\
  --bootstrap-kubeconfig="/var/lib/kubelet/bootstrap-kubeconfig" \\
  --rotate-certificates=true \\

  --rotate-server-certificates=true \\

  --network-plugin=cni \\
  --register-node=true \\
  --v=2
```

**Server Certs**

**Client Certs**

CSR Approval
**Manual**

CSR Approval
**Automatic**

```
kubectl get csr
```

```
NAME                                                AGE   REQUESTOR                   CONDITION
csr-x254z                                           13m   system:node:worker-2        Pending
node-csr-VnzfkE6WdOMOna_S7jIuMTtQzu1-utgAA5gbk3dooUY 13m   system:bootstrap:07401b     Approved,Issued
```

```
kubectl certificate approve csr-x254z
```

```
csr-x254z approved!
```

# DEMO
## TLS Bootstrap Kubelet

# DEMO
## Configure KubeConfig File

# DEMO
Provision Networking

# DEMO

## KubeApi Server to Kubelet Connectivity

# DEMO
**Deploy DNS - CoreDNS**

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ✓ Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Config
- ✓ Choose a Network Solution
- ✓ HA Kubernetes Cluster
- Troubleshooting

- ✓ Provision Infrastructure
- ✓ Secure Cluster Communication
- ✓ Kubernetes Release Binaries
- ✓ Install Kubernetes Master Nodes
- ○ Install Kubernetes Worker Nodes

- ✓ TLS Bootstrapping a Node
- Node end-to-end tests
- ○ Run & Analyze end-to-end tests

KODEKLOUD

# Node end-to-end Tests

# Test - Manual

```
kubectl get nodes
NAME        STATUS    ROLES     AGE    VERSION
worker-1    Ready     <none>    8d     v1.13.0
worker-2    Ready     <none>    8d     v1.13.0
```

```
kubectl get pods
NAMESPACE      NAME                         READY    STATUS     RESTARTS    AGE
kube-system    coredns-69cbb76ff8-9p45d     1/1      Running    1           8d
kube-system    coredns-69cbb76ff8-rmhzt     1/1      Running    0           8d
kube-system    weave-net-58j2j              2/2      Running    2           8d
kube-system    weave-net-rr5dk              2/2      Running    2           8d
```

# Test - Manual

```
▶  kubectl get pods -n kube-system

NAME                             READY   STATUS    RESTARTS   AGE
coredns-78fcdf6894-5dntv         1/1     Running   0          1h
coredns-78fcdf6894-knpzl         1/1     Running   0          1h
etcd-master                      1/1     Running   0          1h
kube-apiserver-master            1/1     Running   0          1h
kube-controller-manager-master   1/1     Running   0          1h
kube-proxy-fvbpj                 1/1     Running   0          1h
kube-proxy-v5r2t                 1/1     Running   0          1h
kube-scheduler-master            1/1     Running   0          1h
weave-net-7kd52                  2/2     Running   1          1h
weave-net-jtl5m                  2/2     Running   1          1h
```

# Test - Manual

```
service kube-apiserver status
```

```
● kube-apiserver.service - Kubernetes API Server
   Loaded: loaded (/etc/systemd/system/kube-apiserver.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-03-20 07:57:25 UTC; 1 weeks 1 days ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 15767 (kube-apiserver)
    Tasks: 13 (limit: 2362)
```

```
service kube-controller-manager status
```

```
● kube-controller-manager.service - Kubernetes Controller Manager
   Loaded: loaded (/etc/systemd/system/kube-controller-manager.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-03-20 07:57:25 UTC; 1 weeks 1 days ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 15771 (kube-controller)
    Tasks: 10 (limit: 2362)
```

```
service kube-scheduler status
```

```
● kube-scheduler.service - Kubernetes Scheduler
   Loaded: loaded (/etc/systemd/system/kube-scheduler.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2019-03-29 01:45:32 UTC; 11min ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 28390 (kube-scheduler)
    Tasks: 10 (limit: 2362)
```

KODEKLOUD

# Test - Manual

```
▶ service kubelet status

● kubelet.service - Kubernetes Kubelet
   Loaded: loaded (/etc/systemd/system/kubelet.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-03-20 14:22:06 UTC; 1 weeks 1 days ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 1281 (kubelet)
    Tasks: 24 (limit: 1152)
```

```
▶ service kube-proxy status

● kube-proxy.service - Kubernetes Kube Proxy
   Loaded: loaded (/etc/systemd/system/kube-proxy.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-03-20 14:21:54 UTC; 1 weeks 1 days ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 794 (kube-proxy)
    Tasks: 7 (limit: 1152)
```

# Test - Manual

```
kubectl run nginx
```
```
deployment.apps/nginx created
```

```
kubectl get pods
```
```
NAME                    READY   STATUS    RESTARTS   AGE
nginx-7cdbd8cdc9-g5q8d  1/1     Running   0          19s
```

```
kubectl scale --replicas=3 deploy/nginx
```
```
deployment.extensions/nginx scaled
```

```
kubectl get pods
```
```
NAME                    READY   STATUS    RESTARTS   AGE    IP          NODE      NOMINATED NODE   READINESS GATES
nginx-7cdbd8cdc9-djj6x  1/1     Running   0          74s    10.40.0.5   worker-2  <none>           <none>
nginx-7cdbd8cdc9-g5q8d  1/1     Running   0          3m29s  10.32.0.5   worker-1  <none>           <none>
nginx-7cdbd8cdc9-rsskt  1/1     Running   0          74s    10.32.0.6   worker-1  <none>           <none>
```

# Test - Manual

```
kubectl expose deployment nginx --port=80 --type=NodePort
```
```
service/nginx exposed
```

```
kubectl get service
```
```
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1     <none>         443/TCP        8d
nginx        NodePort    10.96.0.88    <none>         80:31850/TCP   3s
```

```
curl http://worker-1:31850
```
```
...
<h1>Welcome to nginx!</h1>
...
```

# kubetest

# kubetest – Tests

e2e: ~1000

| sig-api-machinery | sig-apps | sig-auth | sig-cli |
|---|---|---|---|

| sig-network | sig-scheduling | sig-storage |
|---|---|---|

# kubetest - Tests

| sig-api-machinery | sig-apps | sig-auth | sig-cli |
|---|---|---|---|

| sig-network | sig-scheduling | sig-storage |
|---|---|---|

- ✓ Networking should function for intra-pod communication (http)
- ✓ Services should serve a basic endpoint from pods
- ✓ Service endpoints latency should not be very high
- ✓ DNS should provide DNS for services

- ✓ Secrets should be consumable in multiple volumes in a pod
- ✓ Secrets should be consumable from pods in volume with mappings
- ✓ ConfigMap should be consumable from pods in volume

KODEKLOUD

# kubetest - Tests

✓ Networking should function for intra-pod communication (http)

1. **Prepare:** Creates a namespace for this test
2. Creates Test Pod in this namespace Waits for the PODs to come up
3. **Test:** Executes curl on one pod to reach the IP of another over HTTP
4. Record the result
5. **Cleanup:** Delete the namespace

```
[sig-network] Networking Granular Checks: Pods
  should function for intra-pod communication: http [NodeConformance] [Confo
STEP: Building a namespace api object
STEP: Performing setup for networking test in namespace e2e-tests-pod-networ
Mar 14 11:35:19.315: INFO: Waiting up to 10m0s for all (but 0) nodes to be s
STEP: Creating test pods
Mar 14 11:35:39.522: INFO: ExecWithOptions {Command:[/bin/sh -c curl -g -q -
'http://10.32.0.8:8080/dial?request=hostName&protocol=http&host=10.32.0.7&po
drstd PodName:host-test-container-pod ContainerName:hostexec Stdin:<nil> Cap
PreserveWhitespace:false}
Mar 14 11:35:39.522: INFO: >>> kubeConfig: /root/.kube/config
Mar 14 11:35:39.656: INFO: Waiting for endpoints: map[]
Mar 14 11:35:39.660: INFO: ExecWithOptions {Command:[/bin/sh -c curl -g -q -
'http://10.32.0.8:8080/dial?request=hostName&protocol=http&host=10.38.0.12&p
drstd PodName:host-test-container-pod ContainerName:hostexec Stdin:<nil> Cap
PreserveWhitespace:false}
Mar 14 11:35:39.660: INFO: >>> kubeConfig: /root/.kube/config
STEP: Destroying namespace "e2e-tests-pod-network-test-drstd" for this suite
```

# Kubernetes Test-Infra

Build Deploy Test Cleanup

My-kubeadm.sh
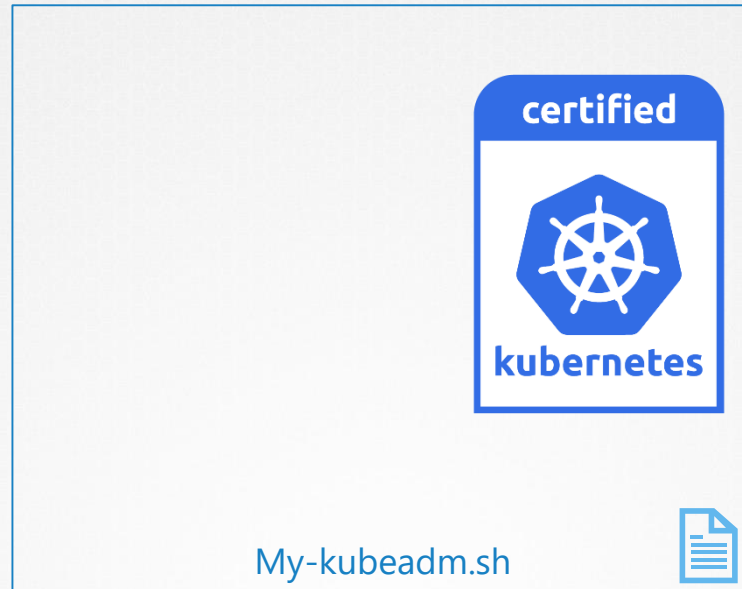
# Kubetest - Tests

e2e: ~1000

conformance: ~160

# Kubetest - Tests

e2e: ~1000

conformance: ~160

# Kubetest - Time

Full e2e = ~1000 Tests / 12 Hours

Conformance = 164 Tests / 1.5 hours

# Course Objectives

- ✓ Core Concepts
- ✓ Scheduling
- ✓ Logging Monitoring
- ✓ Application Lifecycle Management
- ✓ Cluster Maintenance
- ✓ Security
- ✓ Storage
- ✓ Networking
- ✓ Installation, Configuration & Validation

- ✓ Design a Kubernetes Cluster
- ✓ Choose Kubernetes Infrastructure Config
- ✓ Choose a Network Solution
- ✓ HA Kubernetes Cluster
- ● Troubleshooting

- ✓ Provision Infrastructure
- ✓ Secure Cluster Communication
- ✓ Kubernetes Release Binaries
- ✓ Install Kubernetes Master Nodes
- ○ Install Kubernetes Worker Nodes

- ✓ TLS Bootstrapping a Node
- ✓ Node end-to-end tests
- Run & Analyze end-to-end tes

KODEKLOUD

# Run & Analyze E2E Tests

# kubetest- Run

```
go get -u k8s.io/test-infra/kubetest
```

```
kubetest --extract=v1.11.3
kubernetes
```

Note: Version must match the kubernetes
server version

```
cd kubernetes
```

```
export KUBE_MASTER_IP="192.168.26.10:6443"
```

```
export KUBE_MASTER=kube-master
```

```
kubetest --test --provider=skeleton > testout.txt
```

```
kubetest --test --provider=skeleton --test_args="--ginkgo.focus=Secrets" > testout.txt
```

```
kubetest --test --provider=skeleton --test_args="--ginkgo.focus=\[Conformance\]" > testout.txt
```

KODEKLOUD

# kubetest- Run

```
▶ kubetest --test --provider=skeleton --test_args="--ginkgo.focus=\[Conformance\]" > testout.txt
```

```
▶  cat testout.txt
```

```
Client Version: version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.8",
GitCommit:"4e209c9383fa00631d124c8adcc011d617339b3c", GitTreeState:"clean", BuildDate:"2019-02-28T18:49:34Z",
GoVersion:"go1.10.8", C
ompiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.8",
GitCommit:"4e209c9383fa00631d124c8adcc011d617339b3c", GitTreeState:"clean", BuildDate:"2019-02-28T18:40:05Z",
GoVersion:"go1.10.8", C
ompiler:"gc", Platform:"linux/amd64"}
Setting up for KUBERNETES_PROVIDER="skeleton".
Mar 14 11:16:12.419: INFO: Overriding default scale value of zero to 1
Mar 14 11:16:12.419: INFO: Overriding default milliseconds value of zero to 5000
I0314 11:16:12.674596   20093 e2e.go:333] Starting e2e run "933b1eae-464a-11e9-81ea-02f0aa2d49f4" on Ginkgo node 1
Running Suite: Kubernetes e2e suite
===================================
Random Seed: 1552562172 - Will randomize all specs
Will run 167 of 1008 specs

Mar 14 11:16:12.731: INFO: >>> kubeConfig: /root/.kube/config
Mar 14 11:16:12.745: INFO: Waiting up to 30m0s for all (but 0) nodes to be schedulable
Mar 14 11:16:12.770: INFO: Waiting up to 10m0s for all pods (need at least 0) in namespace 'kube-system' to be running
and ready
Mar 14 11:16:12.831: INFO: 12 / 13 pods in namespace 'kube-system' are running and ready (0 seconds elapsed)
Mar 14 11:16:12.831: INFO: expected 2 pod replicas in namespace 'kube-system', 2 are Running and Ready
```

```
• [SLOW TEST:8.486 seconds]
[sig-storage] EmptyDir volumes
/workspace/anago-v1.11.8-
beta.0.41+4e209c9383fa00/src/k8s.io/kubernetes/_output/dockerized/go/src/k8s.io/kubernetes/test/e2e/common/empty_dir.g
o:40
  should support (root,0777,tmpfs) [NodeConformance] [Conformance]
  /workspace/anago-v1.11.8-
beta.0.41+4e209c9383fa00/src/k8s.io/kubernetes/_output/dockerized/go/src/k8s.io/kubernetes/test/e2e/framework/framewor
k.go:684
------------------------------
SSMar 14 13:01:15.397: INFO: Running AfterSuite actions on all node
Mar 14 13:01:15.397: INFO: Running AfterSuite actions on node 1


Summarizing 2 Failures:

[Fail] [sig-network] DNS [It] should provide DNS for services  [Conformance]
/workspace/anago-v1.11.8-
beta.0.41+4e209c9383fa00/src/k8s.io/kubernetes/_output/dockerized/go/src/k8s.io/kubernetes/test/e2e/network/dns_common
.go:497

[Fail] [sig-network] DNS [It] should provide DNS for the cluster  [Conformance]
/workspace/anago-v1.11.8-
beta.0.41+4e209c9383fa00/src/k8s.io/kubernetes/_output/dockerized/go/src/k8s.io/kubernetes/test/e2e/network/dns_common
.go:497

Ran 166 of 1008 Specs in 6302.670 seconds
FAIL! -- 164 Passed | 2 Failed | 0 Pending | 842 Skipped --- FAIL: TestE2E (6302.72s)
FAIL

Ginkgo ran 1 suite in 1h45m3.31433997s
Test Suite Failed
```

# DEMO
Run Smoke Test

# DEMO
## Run End-to-End Tests