Kubernetes Bible

2021-06-03

Table of Contents

[Learning Path 3](#_Toc73802081)

[Architecture 4](#_Toc73802082)

[API Server (Master Node: pod) 4](#_Toc73802083)

[Etcd (Master Node: pod) 4](#_Toc73802084)

[Controller Manager (Master Node: pod) 4](#_Toc73802085)

[Scheduler (Master Node: pod) 4](#_Toc73802086)

[coredns (Master Node: pod) 4](#_Toc73802087)

[Kubelet (Worker Node: native process) 5](#_Toc73802088)

[Container Runtime (Worker Node: native process) 5](#_Toc73802089)

[Kube-proxy (Worker Node: pod) 5](#_Toc73802090)

[calico-kube-controller (Worker Node: pod) 5](#_Toc73802091)

[calico-node (All Nodes: pod) 5](#_Toc73802092)

[Container Runtime Interface (CRI) 5](#_Toc73802093)

[Container Network Interface (CNI) 5](#_Toc73802094)

[Container Storage Interface (CSI) 5](#_Toc73802095)

[Kubernetes Objects 6](#_Toc73802096)

[Computing 6](#_Toc73802097)

[Pod 6](#_Toc73802098)

[Deployment 6](#_Toc73802099)

[DeamonSet 6](#_Toc73802100)

[Network 6](#_Toc73802101)

[ClusterIP 6](#_Toc73802102)

[Nodeport 6](#_Toc73802103)

[Ingress 6](#_Toc73802104)

[LoadBalancer 6](#_Toc73802105)

[Storage 6](#_Toc73802106)

[Persistent Volume 6](#_Toc73802107)

[Persistent Volume Claim 6](#_Toc73802108)

[StatefulSet 6](#_Toc73802109)

[Security 6](#_Toc73802110)

[Service Account 6](#_Toc73802111)

[Scalability 6](#_Toc73802112)

[Kubernetes CICD 7](#_Toc73802113)

[Build 7](#_Toc73802114)

[Deploy 7](#_Toc73802115)

[Helm 7](#_Toc73802116)

[Monitoring 7](#_Toc73802117)

[Logging 7](#_Toc73802118)

[Kubeadm 8](#_Toc73802119)

[Setup 8](#_Toc73802120)

[Deploy 8](#_Toc73802121)

[Monitor 8](#_Toc73802122)

[Logging 8](#_Toc73802123)

[Scalability 8](#_Toc73802124)

[Rancher 9](#_Toc73802125)

[Setup 9](#_Toc73802126)

[Deploy 9](#_Toc73802127)

[Monitor 9](#_Toc73802128)

[Logging 9](#_Toc73802129)

[Scalability 9](#_Toc73802130)

[AWS EKS 10](#_Toc73802131)

[Setup 10](#_Toc73802132)

[Deploy 10](#_Toc73802133)

[Monitor 10](#_Toc73802134)

[Logging 10](#_Toc73802135)

[Scalability 10](#_Toc73802136)

[Azure AKS 11](#_Toc73802137)

[Setup 11](#_Toc73802138)

[Deploy 11](#_Toc73802139)

[Monitor 11](#_Toc73802140)

[Logging 11](#_Toc73802141)

[Scalability 11](#_Toc73802142)

[Comparison 12](#_Toc73802143)

# Learning Path

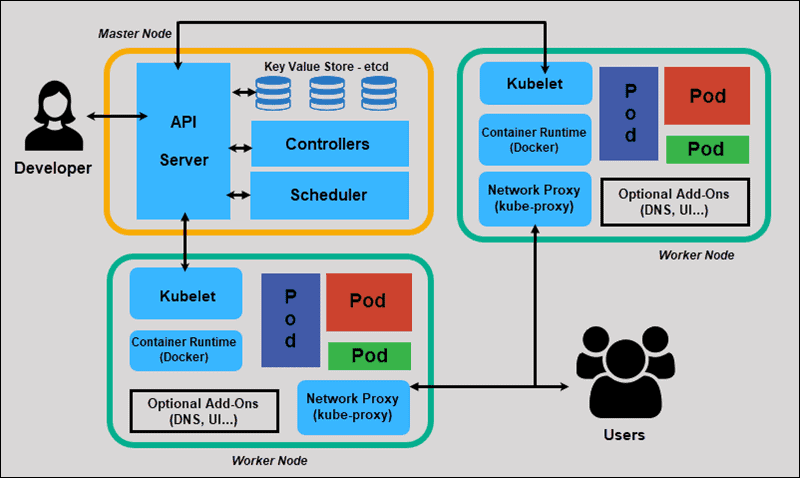
Udemy

Youtube (justmeandopensource)

Udemy

Kubernetes document (<https://kubernetes.io/docs>)

# Architecture



## API Server (Master Node: pod)

* Front-end of the control plane and the only component in the control plane that we interact with directly.
* Internal system components, as well as external user components, all communicate via the same API.

## Etcd (Master Node: pod)

* Stores the entire configuration and state of the cluster. The Master node queries etcd to retrieve parameters for the state of the nodes, pods, and containers.

## Controller Manager (Master Node: pod)

* Manage different controllers.
* Obtains the desired state from the API Server. It checks the current state of the nodes it is tasked to control, and determines if there are any differences, and resolves them, if any.

## Scheduler (Master Node: pod)

* Watches for new requests coming from the API Server and assigns them to healthy nodes. It ranks the quality of the nodes and deploys pods to the best-suited node. If there are no suitable nodes, the pods are put in a pending state until such a node appears.

## coredns (Master Node: pod)

* DNS.

## Kubelet (Worker Node: native process)

* Handles all communication between the master and the node. It receives commands from the master in the form of a **Manifest** which defines the workload and the operating parameters.
* Executes any configured liveness probes and readiness checks periodically.
* Interfaces with the **Container Runtime** that is responsible for creating, starting, and monitoring pods.
* Has an internal HTTP server exposing a read-only view at port 10255. Endpoints: /healthz /pods /spec

## Container Runtime (Worker Node: native process)

* Manage container.

## Kube-proxy (Worker Node: pod)

* Acts like a network proxy and load balancer.
* Translates service object into rules in the local **iptables** on the worker node.

## calico-kube-controller (Worker Node: pod)

* Container networking.

## calico-node (All Nodes: pod)

* Container networking.

## Container Runtime Interface (CRI)

Kubernetes uses CRI to interface with your chosen container runtime.

<https://kubernetes.io/blog/2016/12/container-runtime-interface-cri-in-kubernetes/>

* Docker (default)

## Container Network Interface (CNI)

## Container Storage Interface (CSI)

# Kubernetes Objects

## Computing

### Pod

### Deployment

### DeamonSet

## Network

These services are virtual IP addresses.   
--pod-network-cidr  
--service-cidr (default ‘10.96.0.0/12’)

### ClusterIP

### Nodeport

### Ingress

<https://medium.com/flant-com/comparing-ingress-controllers-for-kubernetes-9b397483b46b>

Kubernetes Ingress is recommended.

How does it work???

### LoadBalancer

Used in cloud environment, the cloud provider will create Load Balancer for this service.

## Storage

### Persistent Volume

### Persistent Volume Claim

### StatefulSet

## Security

### Service Account

## Scalability

Kubernetes has three auto-scaling abilities for deployments: horizontal pod autoscaler (HPA), vertical pod autoscaler (VPA) and cluster autoscaling.

**Horizontal pod autoscaler** automatically scales the quantity of pods in deployment, a replication controller, replica set or stateful set based on perceived CPU utilization.

**Vertical pod autoscaler** recommends suitable values to be set for CPU and memory requests and limits, and it can automatically update the values.

**Cluster Autoscaler** expands and shrinks the size of the pool of worker nodes. It adjusts the size of a Kubernetes cluster depending on the current utilization.

# Kubernetes CICD

## Build

Build docker image with tag from build pipeline JOB\_ID, and push to docker repository.

## Deploy

Deployer use update Helm Chart??? and deploy.

### Helm

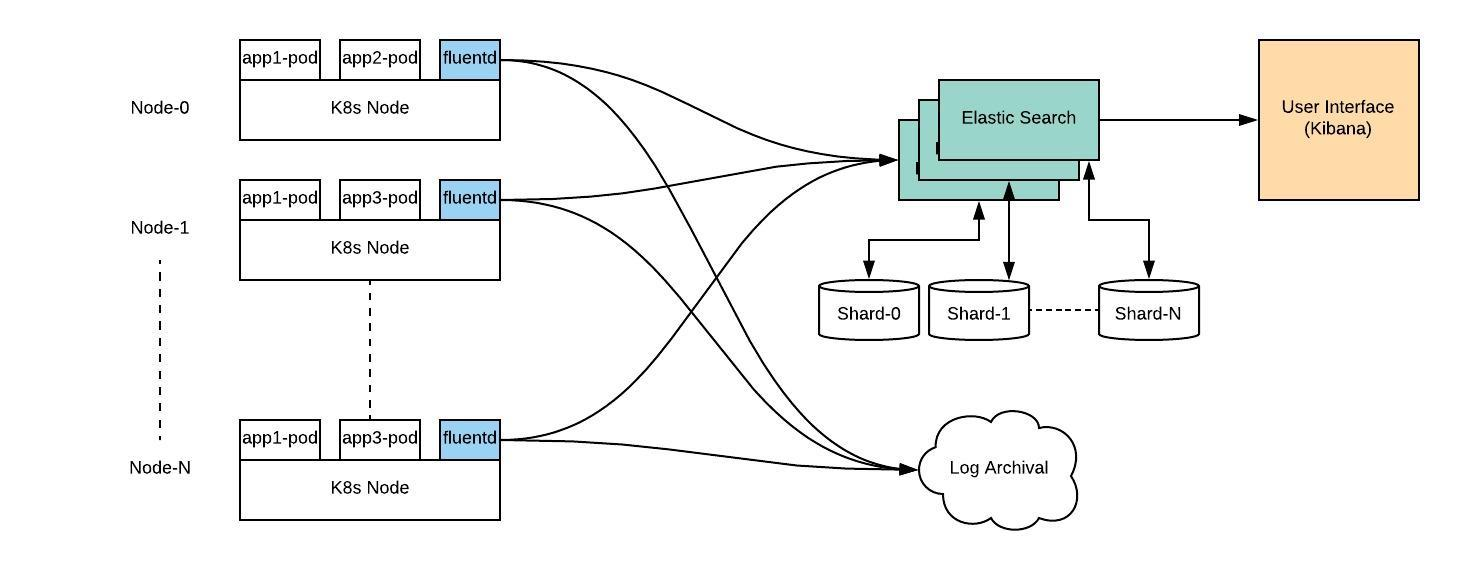
Helm is owned by Infra as the app developers will not know the CPU and memory usage. The Helm chart need to be stored in Helm Chart Repository.

## Monitoring

Prometheus/Grafana

## Logging

EFK



Loki/Grafana

# Kubeadm

## Setup

**Setup Cluster (1 Master Node, 2 Worker Nodes)**CentOS 8.3 <https://github.com/justmeandopensource/kubernetes/blob/master/docs/install-cluster-centos-7.md>

Extra steps

* Remember masternode must be 2CPU (VirtualBox adjust Execution Cap to 50%)
* Ignore the version in the instruction when install docker/kubeadm/kubectl/kubelet
* kubeadm init --ignore-preflight-errors=NumCPU --apiserver-advertise-address=192.168.56.105 --pod-network-cidr=10.244.0.0/16  
  (use **Flannel,** triedCalico but not success maybe due to it only support CentOS 7)
* Install network before join  
  kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
* kubectl get cs  
  Encounter: “controller-manager Unhealthy Get "http://127.0.0.1:10252/healthz": dial tcp 127.0.0.1:10252: connect: connection refused”  
  sudo nano /etc/kubernetes/manifests/kube-controller-manager.yaml  
  sudo nano /etc/kubernetes/manifests/kube-scheduler.yaml  
  Comment out - --port=0

**Uninstall**

kubeadm reset  
sudo rm /etc/cni/net.d

**Setup haproxy and Kubernetes Ingress Controller (on Master Node)**<https://jhooq.com/ingress-controller-nginx/>

<https://www.digitalocean.com/community/tutorials/how-to-configure-haproxy-logging-with-rsyslog-on-centos-8-quickstart>

Use Kubernetes ingress NGINX, reason is Rancher default use it. Try Nginx-ingress but not success.

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.46.0/deploy/static/provider/baremetal/deploy.yaml>

Need to define nodeport to specific port like 30080/30443, so that haproxy loadbalance can be configured.

**Error**: pod in status of “CrashLoopBackOff”

* Diagnosis: run below command to view error message  
  kubectl describe -n nginx-ingress pod <*pod\_name*>
* Find the pod running on which node, and go to the node run “docker ps -a” to get failed container
* Docker logs <*container\_id*>

**Error**: Flannel - CoreDns pods are stuck with status ContainerCreating

* solution was to delete /etc/cni/\* (in my case, it is due to I installed Calico before)

**Error**: Post "https://ingress-nginx-controller-admission.ingress-nginx.svc:443/networking/v1beta1/ingresses?timeout=10s": context deadline exceeded

* kubectl delete -A ValidatingWebhookConfiguration ingress-nginx-admission

## Deploy

<https://www.magalix.com/blog/deploying-an-application-on-kubernetes-from-a-to-z>

## Monitor

## Logging

## Scalability

# Rancher

## Setup

Rancher default:  
- Use Canal as CNI provider.  
- Install Ingress-Nginx as ingress controller. (Question: how to do load balancing???)

## Deploy

## Monitor

## Logging

## Scalability

# AWS EKS

## Setup

## Deploy

## Monitor

## Logging

## Scalability

# Azure AKS

## Setup

## Deploy

## Monitor

## Logging

## Scalability

# Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |