

Kubernetes Vagrant CI / CD GitHub Project Repo: kubernetes Vagrant

Sample of Work of fully automate build, deploy, validate containers locally. Deploy on a k8s cluster and validations.

September 30, 2021

# Outline I

Contents

#### Introduction

Container Runtime Interface(s) Kubernetes

CI / CD

Project smooth CI / CD How to accomplish requirements The correct tool of choice

## Implementation

CI / CD Flow Build / Deploy / Test CI / CD Flow Kubernetes

# Outline II

Contents

#### Architecture

Minimal Kubernetes Components
My view of Kubernetes Minimal Core Components

*Demo* Demo CI / CE

Summary

Key points repetition

**Bibliography** 

# High level description

- VM Vs Container.
- ► What is a Pod (pea pod)
- Container Runtime Interface(s) (CRI).

- Docker Vs Podman
- ► What is actually k8s?

### Virtual Machine Vs Container

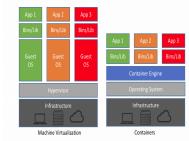


Figure 1: k8s Overview

ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliography

Kubernetes

# High level description

- VM Vs Container.
- ► What is a Pod (pea pod)?
- Container Runtime Interface(s) (CRI).

- Docker Vs Podman
- ► What is actually k8s?

#### Container inside Pod

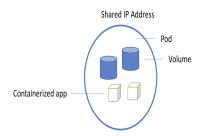


Figure 1: k8s Overview

Kubernetes

# High level description

- VM Vs Container.
- ► What is a Pod (pea pod)?
- Container Runtime Interface(s) (CRI).
  - Docker
  - Podman
  - ► CRI\_O
- Docker Vs Podman.
- ► What is actually k8s?

### Container Runtime Interface

Application Definition / Development

**Cloud Native Services** 

Cloud Native Runtime

Cloud Native OS / Provisioning

Infrastructure (Bare Metal/Cloud)

Figure 1: k8s Overview

ontents Introduction  $CI\ /\ CD$  Implementation Architecture Demo Summary Bibliographs

Kubernetes

# High level description

- VM Vs Container.
- ► What is a Pod (pea pod)?
- Container Runtime Interface(s) (CRI).
  - Docker
  - Podman
  - ► CRI-O
- Docker Vs Podman.
- ▶ What is actually k8s?

Most known (insecure) socket



Figure 1: k8s Overview

ontents Introduction CI/CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

Kubernetes

# High level description

- VM Vs Container.
- ► What is a Pod (pea pod)?
- Container Runtime Interface(s) (CRI).
  - Docker
  - Podman
  - CRI-O
- Docker Vs Podman.
- ► What is actually k8s?

### Most unknown (secure) socket

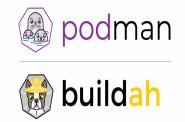


Figure 1: k8s Overview

ontents Introduction CI/CD Implementation Architecture Demo Summary Bibliograph,  $\bigcirc$   $\bigcirc$   $\bigcirc$   $\bigcirc$   $\bigcirc$   $\bigcirc$   $\bigcirc$ 

Kubernetes

# High level description

- VM Vs Container.
- ► What is a Pod (pea pod)?
- Container Runtime Interface(s) (CRI).
  - Docker
  - Podman
  - ► CRI-O
- Docker Vs Podman.
- ► What is actually k8s?

## Lightest fastest socket



CRI-O: OCI-based Kubernetes Runtime

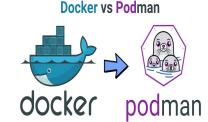
Figure 1: k8s Overview

ontents Introduction  $CI\ /\ CD$  Implementation Architecture Demo Summary Bibliographs

Kubernetes

# High level description

- VM Vs Container.
- ► What is a Pod (pea pod)?
- Container Runtime Interface(s) (CRI).
  - Docker
  - Podman
  - ► CRI-O
- Docker Vs Podman.
- What is actually k8s?



Read about it

Figure 1: k8s Overview

ontents Introduction CI/CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

Kubernetes

# High level description

- VM Vs Container.
- ► What is a Pod (pea pod)?
- Container Runtime Interface(s) (CRI).
  - Docker
  - Podman
  - ► CRI-O
- Docker Vs Podman.
- ► What is actually k8s?

## Is a puzzle of elements



Figure 1: k8s Overview

Contents Introduction CI / CD Implementation Architecture Demo Summary Bibliography O O O

Project smooth CI / CD

# Problems / Desires / Solutions on CI / CD

### Assessment Requirements

- CI / CD (Locally / Remotely).
- Everything As a Code
- ► Validation Locally!!!
- ► OS / Infra. dependencies

### Justification of Requirements

- ► Works on my PC, why not on Cloud?
- ► Human error (manual).
- ► Test (automatically).
- ► GitHub Actions, Jenkins

Project smooth CI / CD

# Problems / Desires / Solutions on CI / CD

### Assessment Requirements

- CI / CD (Locally / Remotely).
- ► Everything As a Code
- ► Validation Locally!!!
- ► OS / Infra. dependencies

### Justification of Requirements

- ► Works on my PC, why not on Cloud?
- ► Human error (manual).
- ► Test (automatically).
- ► GitHub Actions, Jenkins

Contents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

Project smooth CI / CD

# Problems / Desires / Solutions on CI / CD

### Assessment Requirements

- CI / CD (Locally / Remotely).
- Everything As a Code
- ► Validation Locally!!!
- ▶ OS / Infra. dependencies

### Justification of Requirements

- Works on my PC, why not on Cloud?
- ► Human error (manual).
- Test (automatically).
- ► GitHub Actions, Jenkins

4/14

Contents Introduction CI / CD Implementation Architecture Demo Summary Bibliography O O O

Project smooth CI / CD

# Problems / Desires / Solutions on CI / CD

### Assessment Requirements

- CI / CD (Locally / Remotely).
- Everything As a Code
- ► Validation Locally!!!
- OS / Infra. dependencies.

### Justification of Requirements

- Works on my PC, why not on Cloud?
- ► Human error (manual).
- Test (automatically).
- ► GitHub Actions, Jenkins

 $^{\prime}$  ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliograph:  $\circ$   $\circ$   $\circ$   $\circ$ 

# Problems / Desires / Solutions on CI / CD

### Assessment Requirements

- CI / CD (Locally / Remotely).
- Everything As a Code
- Validation Locally!!!
- OS / Infra. dependencies.

## Justification of Requirements

- ► Works on my PC, why not on Cloud?
  - Human error (manual).
  - Test (automatically).
- GitHub Actions, Jenkins

- Solution has to be reproducible locally. Exactly as cloud.
- ► Minimal human interaction. Auto error handling (Rollback)
- ► Powerful PCs! (8 CPUs / 35 GB RAM). Browsing (tabs)?
- ▶ No Vendor binding (Azzure DevOps, Jenkins, Bamboo etc)

 $^{\prime}$  ontents Introduction  $^{\prime}$   $^{\prime}$ 

# Problems / Desires / Solutions on CI / CD

### Assessment Requirements

- CI / CD (Locally / Remotely).
- Everything As a Code
- Validation Locally!!!
- OS / Infra. dependencies.

## Justification of Requirements

- Works on my PC, why not on Cloud?
- ► Human error (manual).
- ► Test (automatically).
- ► GitHub Actions, Jenkins

- Solution has to be reproducible locally. Exactly as cloud.
- Minimal human interaction. Auto error handling (Rollback)!
- ▶ Powerful PCs! (8 CPUs / 35 GB RAM). Browsing (tabs)?
- ▶ No Vendor binding (Azzure DevOps, Jenkins, Bamboo etc)

ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliograph  $\circ$   $\circ$   $\circ$   $\circ$ 

# Problems / Desires / Solutions on CI / CD

### Assessment Requirements

- CI / CD (Locally / Remotely).
- Everything As a Code
- Validation Locally!!!
- OS / Infra. dependencies.

## Justification of Requirements

- Works on my PC, why not on Cloud?
  - Human error (manual).
  - ► Test (automatically).
- ► GitHub Actions, Jenkins

- Solution has to be reproducible locally. Exactly as cloud.
- Minimal human interaction. Auto error handling (Rollback)!
- ► Powerful PCs! (8 CPUs / 35 GB RAM). Browsing (tabs)?
- ▶ No Vendor binding (Azzure DevOps, Jenkins, Bamboo etc)

 $^{\prime}$  ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliograph:  $\circ$   $\circ$   $\circ$   $\circ$ 

# Problems / Desires / Solutions on CI / CD

### Assessment Requirements

- CI / CD (Locally / Remotely).
- Everything As a Code
- Validation Locally!!!
- OS / Infra. dependencies.

## Justification of Requirements

- Works on my PC, why not on Cloud?
  - Human error (manual).
  - ► Test (automatically).
- ► GitHub Actions, Jenkins

- Solution has to be reproducible locally. Exactly as cloud.
- Minimal human interaction. Auto error handling (Rollback)!
- Powerful PCs! (8 CPUs / 35 GB RAM). Browsing (tabs)?
- ▶ No Vendor binding (Azzure DevOps, Jenkins, Bamboo etc).

Contents Introduction CI/CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

How to accomplish requirements

# Solution to problem

#### **Problems**

- ► Solution has to be reproducible locally. Exactly as cloud.
- Minimal human interaction. Auto error handling (Rollback)!
- Powerful PCs! (8 CPUs / 35 GB RAM). Browsing (tabs)?
- No Vendor binding (Azzure DevOps, Jenkins, Bamboo etc)

- Containers. Build / deploy / validate locally (controlled env)!
- ► Fully automated procedure on every step
- Launch a k8s cluster locally and run all tests locally!
- ► High Level Programming Language with error handling!

CI/CD Implementation Architecture Demo Summary Bibliography O O O O

How to accomplish requirements

# Solution to problem

#### **Problems**

- ► Solution has to be reproducible locally. Exactly as cloud.
- Minimal human interaction. Auto error handling (Rollback)!
- ► Powerful PCs! (8 CPUs / 35 GB RAM). Browsing (tabs)?
- ► No Vendor binding (Azzure DevOps, Jenkins, Bamboo etc)

- ► Containers. Build / deploy / validate locally (controlled env)!
- ► Fully automated procedure on every step!
- ► Launch a k8s cluster locally and run all tests locally!
- ► High Level Programming Language with error handling!

 $^{\prime}$  ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliographs  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

# Solution to problem

#### **Problems**

- Solution has to be reproducible locally. Exactly as cloud.
- Minimal human interaction. Auto error handling (Rollback)!
- ► Powerful PCs! (8 CPUs / 35 GB RAM). Browsing (tabs)?
- ▶ No Vendor binding (Azzure DevOps, Jenkins, Bamboo etc)

- Containers. Build / deploy / validate locally (controlled env)!
- ► Fully automated procedure on every step!
- Launch a k8s cluster locally and run all tests locally!
- ► High Level Programming Language with error handling!

CI / CD

How to accomplish requirements

# Solution to problem

#### **Problems**

- Solution has to be reproducible locally. Exactly as cloud.
- Minimal human interaction. Auto error handling (Rollback)!
- Powerful PCs! (8 CPUs / 35 GB RAM). Browsing (tabs)?
- ▶ No Vendor binding (Azzure DevOps, Jenkins, Bamboo etc).

- Containers. Build / deploy / validate locally (controlled env)!
- Fully automated procedure on every step!
- ► Launch a k8s cluster locally and run all tests locally!
- ► High Level Programming Language with error handling!

CI / CD

The correct tool of choice

# Ansible

## Possible Questions

- ► Why Ansible?

- ▶ Written in Python 2/3. Developed and maintained by RedHat.

The correct tool of choice

# Ansible

## Possible Questions

- ▶ Why Ansible?
- ► Are there any benefits of this tool?
- Ansible works on ssh how it will work locally?
- ► How it can interact with Containers, k8s, Cloud, tests?

- ▶ Written in Python 2/3. Developed and maintained by RedHat.
- Woks perfectly without extra configurations on all OS.
- It can be configured to run on localhost without ssh session.
- ▶ It has infinite amount of packages for OS, Containers, Cloud.

 $^{\prime}$  ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliograph  $\circ$ 

The correct tool of choice

# Ansible

### Possible Questions

- ▶ Why Ansible?
- ► Are there any benefits of this tool?
- Ansible works on ssh how it will work locally?
- ► How it can interact with Containers, k8s, Cloud, tests?

- ▶ Written in Python 2/3. Developed and maintained by RedHat.
- ▶ Woks perfectly without extra configurations on all OS.
- ▶ It can be configured to run on localhost without ssh session.
- ▶ It has infinite amount of packages for OS, Containers, Cloud.

 $^{\prime}$  ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliograph  $\circ$ 

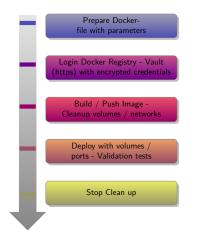
The correct tool of choice

# Ansible

### Possible Questions

- Why Ansible?
- ► Are there any benefits of this tool?
- Ansible works on ssh how it will work locally?
- ► How it can interact with Containers, k8s, Cloud, tests?

- ▶ Written in Python 2/3. Developed and maintained by RedHat.
- Woks perfectly without extra configurations on all OS.
- It can be configured to run on localhost without ssh session.
- ▶ It has infinite amount of packages for OS, Containers, Cloud.

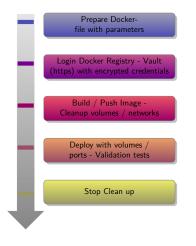


- ► Dockerfile (template).
- ► Vault (https)
- Any socket.

- ▶ Deployment (volume).
- ► Validation (tests).
- Stop, Cleanup

ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

CI / CD Flow Build / Deploy / Test

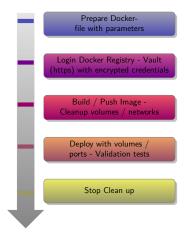


- ► Dockerfile (template).
- ► Vault (https).
- Any socket.

- ▶ Deployment (volume).
- Validation (tests).
- Stop, Cleanup

ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

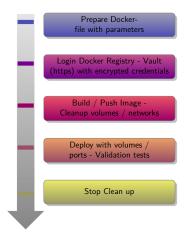
CI / CD Flow Build / Deploy / Test



- Dockerfile (template).
- ► Vault (https).
- ► Any socket.
  - Azzure Registry
  - Build Dockerfile
  - Push Image
    - Logout Azzure.
  - Prune everything
  - Raise error (if).
- Deployment (volume).
- ► Validation (tests).
- Stop, Cleanup

ntents Introduction CI / CD Implementation Architecture Demo Summary Bibliography

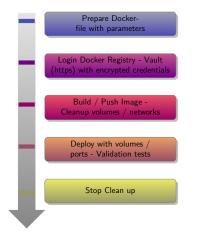
CI / CD Flow Build / Deploy / Test



- ► Dockerfile (template).
- ► Vault (https).
- Any socket.
  - Azzure Registry.
    - Build Dockerfile
  - Push Image
  - Logout Azzure
  - Prune everything.
  - Raise error (if).
- Deployment (volume).
- ► Validation (tests).
- Stop, Cleanup

Atents Introduction CI / CD Implementation Architecture Demo Summary Bibliography

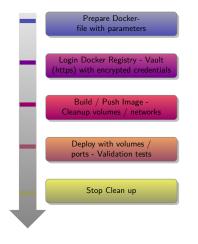
CI / CD Flow Build / Deploy / Test



- ► Dockerfile (template).
- ► Vault (https).
- ► Any socket.
  - Azzure Registry.
  - Build Dockerfile.
  - Push Image
  - Logout Azzure
  - Prune everything.
    - Raise error (if).
- Deployment (volume).
- ► Validation (tests).
- ► Stop, Cleanup

Atents Introduction CI / CD Implementation Architecture Demo Summary Bibliography

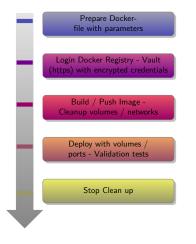
CI / CD Flow Build / Deploy / Test



- ► Dockerfile (template).
- ► Vault (https).
- Any socket.
  - Azzure Registry.
    - Build Dockerfile.
    - Push Image.
    - Logout Azzure.
    - Prune everything.
    - Raise error (if).
- Deployment (volume).
- Validation (tests).
- Stop, Cleanup

ntents Introduction CI / CD Implementation Architecture Demo Summary Bibliography

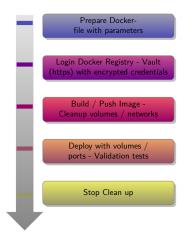
CI / CD Flow Build / Deploy / Test



- ► Dockerfile (template).
- ► Vault (https).
- Any socket.
  - Azzure Registry.
    - Build Dockerfile.
  - Push Image.
  - Logout Azzure.
    - Prune everything.
  - Raise error (if).
- Deployment (volume).
- ► Validation (tests).
- ► Stop, Cleanup

ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

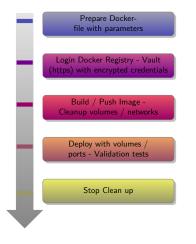
CI / CD Flow Build / Deploy / Test



- ► Dockerfile (template).
- ► Vault (https).
- Any socket.
  - Azzure Registry.
  - Build Dockerfile.
  - Push Image.
  - Logout Azzure.
  - Prune everything.
  - Raise error (if).
- Deployment (volume).
- Validation (tests).
- ► Stop, Cleanup

ntents Introduction CI / CD Implementation Architecture Demo Summary Bibliography

CI / CD Flow Build / Deploy / Test

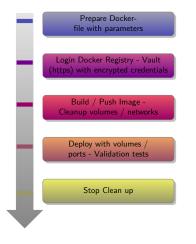


- ► Dockerfile (template).
- Vault (https).
- Any socket.
  - Azzure Registry.
    - Build Dockerfile.
  - Push Image.
  - Logout Azzure.
  - Prune everything.
  - Raise error (if).
- Deployment (volume).
- ► Validation (tests).
- Stop, Cleanup

ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

CI / CD Flow Build / Deploy / Test

#### Local Procedure

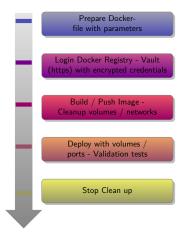


- ► Dockerfile (template).
- Vault (https).
- Any socket.
  - Azzure Registry.
  - Build Dockerfile.
  - Push Image.
  - Logout Azzure.
  - Prune everything.
  - Raise error (if).
- Deployment (volume).
- ► Validation (tests).
- ► Stop, Cleanup

ontents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

CI / CD Flow Build / Deploy / Test

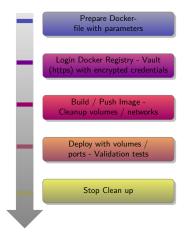
#### Local Procedure



- ► Dockerfile (template).
- ► Vault (https).
- Any socket.
  - Azzure Registry.
  - Build Dockerfile.
  - Push Image.
  - Logout Azzure.
  - Prune everything.
  - Raise error (if).
- Deployment (volume).
- ► Validation (tests).
- Stop, Cleanup

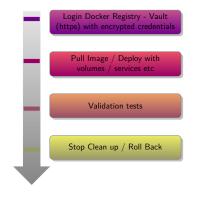
CI / CD Flow Build / Deploy / Test

#### Local Procedure



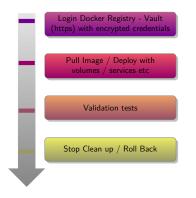
- ► Dockerfile (template).
- ► Vault (https).
- Any socket.
  - Azzure Registry.
  - Build Dockerfile.
  - Push Image.
  - Logout Azzure.
  - Prune everything.
  - Raise error (if).
- Deployment (volume).
- Validation (tests).
- ► Stop, Cleanup.

CI / CD Flow Kubernetes



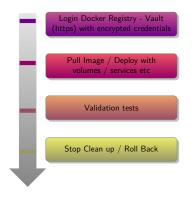
- ► Vault (https).
- Any socket.
- Logout Azzure.
- ► Validation (tests).

CI / CD Flow Kubernetes



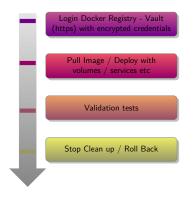
- Vault (https).
  - Any socket.
    - Azzure Registry
      - Pull Image
- ▶ Deployment (volume).
- Logout Azzure.
- ► Validation (tests).

CI / CD Flow Kubernetes



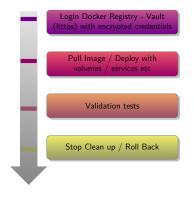
- Vault (https).
  - Any socket.
    - Azzure Registry.
    - Pull Image
- ▶ Deployment (volume).
- Logout Azzure.
- ► Validation (tests).

CI / CD Flow Kubernetes



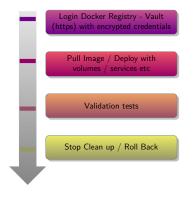
- ► Vault (https).
  - Any socket.
    - Azzure Registry.
    - Pull Image.
- ▶ Deployment (volume).
- Logout Azzure.
- ► Validation (tests).

CI / CD Flow Kubernetes



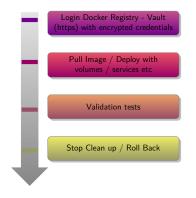
- Vault (https).
  - Any socket.
    - Azzure Registry.
    - Pull Image.
- ► Deployment (volume).
- Logout Azzure.
- ► Validation (tests).

CI / CD Flow Kubernetes



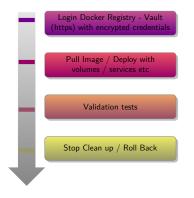
- Vault (https).
  - Any socket.
    - Azzure Registry.
    - Pull Image.
- Deployment (volume).
- Logout Azzure.
- ► Validation (tests)

CI / CD Flow Kubernetes



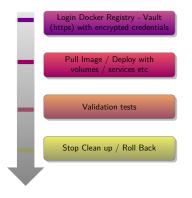
- Vault (https).
  - Any socket.
    - Azzure Registry.
    - Pull Image.
- Deployment (volume).
- Logout Azzure.
- ► Validation (tests).
  - Raise error (if)
  - ► Stop, Cleanup
  - ▶ Roll back

CI / CD Flow Kubernetes



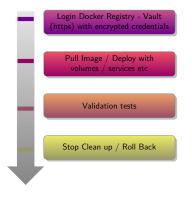
- Vault (https).
  - Any socket.
    - Azzure Registry.
    - Pull Image.
- Deployment (volume).
- Logout Azzure.
- Validation (tests).
  - Raise error (if).
  - Stop, Cleanup.
  - ► Roll back.

CI / CD Flow Kubernetes



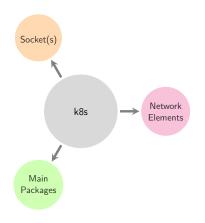
- Vault (https).
  - Any socket.
    - Azzure Registry.
    - Pull Image.
- Deployment (volume).
- Logout Azzure.
- Validation (tests).
  - Raise error (if).
  - ► Stop, Cleanup.
  - Roll back.

CI / CD Flow Kubernetes



- Vault (https).
  - Any socket.
    - Azzure Registry.
    - Pull Image.
- Deployment (volume).
- Logout Azzure.
- Validation (tests).
  - Raise error (if).
  - ► Stop, Cleanup.
  - ► Roll back.

### Kubernetes Minimal Core Components

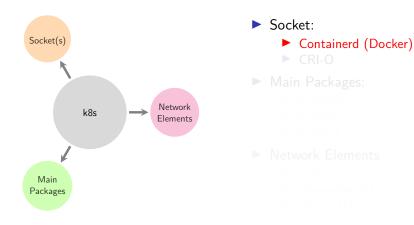


#### Socket:

- ► Containerd (Docker)
- ► CRI-(
- ▶ Main Packages:

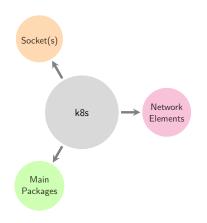
▶ Network Elements

Minimal Kubernetes Components



Minimal Kubernetes Components

#### Kubernetes Minimal Core Components

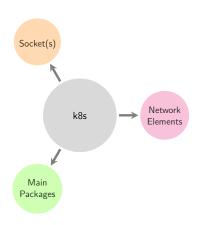


- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:

▶ Network Elements

Contents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$ 

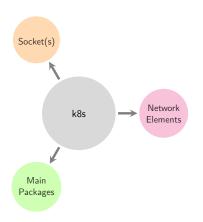
Minimal Kubernetes Components



- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:
  - kubeadm
    - kubelet
    - kubectl
- Network Elements

Contents Introduction CI/CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$ 

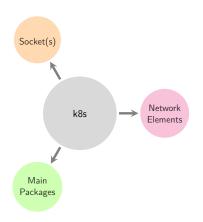
Minimal Kubernetes Components



- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:
  - kubeadm
    - kubelet
    - kubectl
- Network Elements

Contents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$ 

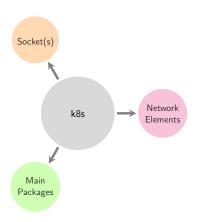
Minimal Kubernetes Components



- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:
  - kubeadm
    - kubelet
    - kubectl
- Network Elements

Contents Introduction CI/CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$ 

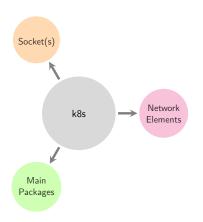
Minimal Kubernetes Components



- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:
  - kubeadm
    - kubelet
    - kubectl
- Network Elements

Contents Introduction CI / CD Implementation **Architecture** Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$ 

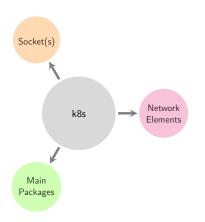
Minimal Kubernetes Components



- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:
  - kubeadm
    - kubelet
  - kubectl
- ► Network Elements
  - Calico [1]
  - ► WeaveNet [1]
  - Cilium [1]

Contents Introduction CI/CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$ 

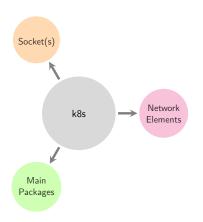
Minimal Kubernetes Components



- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:
  - kubeadm
    - kubelet
  - kubectl
- Network Elements
  - ► Calico [1]
  - ► WeaveNet [1]
  - Cilium [1]

Contents Introduction CI/CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$ 

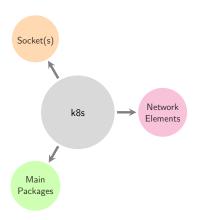
Minimal Kubernetes Components



- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:
  - kubeadm
    - kubelet
    - kubectl
- Network Elements
  - ► Calico [1]
  - ► WeaveNet [1]
  - Cilium [1]

Contents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$ 

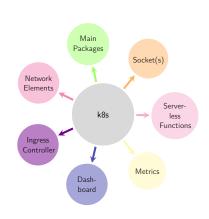
Minimal Kubernetes Components



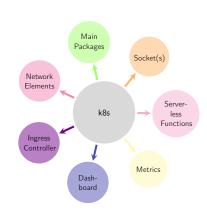
- Socket:
  - Containerd (Docker)
  - ► CRI-O
- ► Main Packages:
  - kubeadm
    - kubelet
    - kubectl
- Network Elements
  - ► Calico [1]
  - ► WeaveNet [1]
  - ► Cilium [1]

#### ▶ Previous Elements

- ► Ingress Controller:
  - ► NGINX [3]
- ► Metrics (HPA)
- Dashboard
- Serveless Functions, Functions as a Service (FaaS)
- Operatin System:

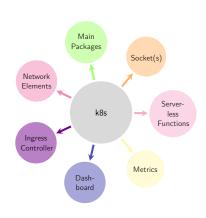


- Previous Elements
- Ingress Controller:

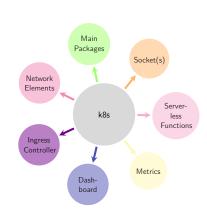


My view of Kubernetes Minimal Core Components

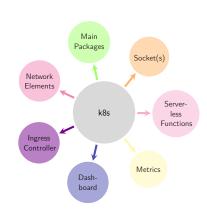
- Previous Elements
- ► Ingress Controller:
  - ► NGINX [3]



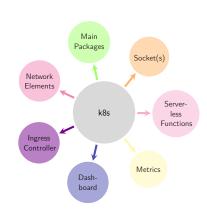
- Previous Elements
- ► Ingress Controller:
  - NGINX [3]
  - ► HaProxy [3]



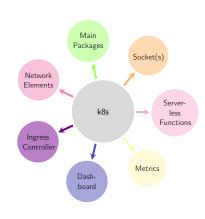
- Previous Elements
- ► Ingress Controller:
  - ► NGINX [3]
  - ► HaProxy [3]
- ► Metrics (HPA)
- Self Signed CA
- Serveless Functions, Functions as a Service (FaaS)
- ▶ Operatin System:



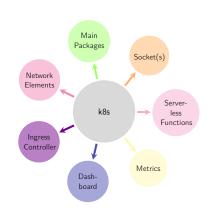
- Previous Elements
- ► Ingress Controller:
  - ▶ NGINX [3]
  - ► HaProxy [3]
- ► Metrics (HPA)
- Dashboard
  - Self Signed CA
- Serveless Functions, Functions as a Service (FaaS)
- Operatin System:



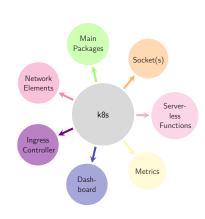
- Previous Elements
- ► Ingress Controller:
  - ► NGINX [3]
  - ► HaProxy [3]
- ► Metrics (HPA)
- Dashboard
  - Self Signed CA
- Serveless Functions, Functions as a Service (FaaS)
- ► Operatin System:



- Previous Elements
- ► Ingress Controller:
  - ► NGINX [3]
  - ► HaProxy [3]
- ► Metrics (HPA)
- Dashboard
  - Self Signed CA
- Serveless Functions, Functions as a Service (FaaS)
- Operatin System:

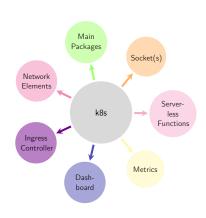


- Previous Elements
- ► Ingress Controller:
  - ► NGINX [3]
  - ► HaProxy [3]
- ► Metrics (HPA)
- Dashboard
  - Self Signed CA
- Serveless Functions, Functions as a Service (FaaS)
- ► Operatin System:
  - ▶ Ubuntu



### Local Auto Deployment

- Previous Elements
- ► Ingress Controller:
  - ► NGINX [3]
  - ► HaProxy [3]
- ► Metrics (HPA)
- Dashboard
  - Self Signed CA
- Serveless Functions, Functions as a Service (FaaS)
- Operatin System:
  - ► Ubuntu





 $Demo\ CI\ /\ CD$ 

## Coming up: Demo

tents Introduction CI / CD Implementation Architecture Demo Summary Bibliography  $\circ$   $\circ$   $\circ$   $\circ$   $\circ$ 

Key points repetition

#### Summary

- ► In section "Introduction" page "2" Container Run Time Interfaces.
- ▶ In section "CI / CD" page "3" Problems / Desires / Solutions on CI / CD.
- ▶ In section "Implementation" page "4" CI / CD Flow Build / Deploy / Test.

#### Extra Notes

- Demo on CI / CD build / deploy / validation and error handling cases.
- ▶ Both the CI / CD and Kubernetes project are provided as open source contribution. The Presentation was written in LATEX

Key points repetition

#### Summary

- ► In section "Introduction" page "2" Container Run Time Interfaces.
- ▶ In section "CI / CD" page "3" Problems / Desires / Solutions on CI / CD.
- In section "Implementation" page "4" CI / CD Flow Build / Deploy / Test.

- Demo on CI / CD build / deploy / validation and error handling cases.
- ▶ Both the CI / CD and Kubernetes project are provided as open source contribution. The Presentation was written in LATEX

Key points repetition

#### Summary

- ► In section "Introduction" page "2" Container Run Time Interfaces.
- ▶ In section "CI / CD" page "3" Problems / Desires / Solutions on CI / CD.
- In section "Implementation" page "4" CI / CD Flow Build / Deploy / Test.

- Demo on CI / CD build / deploy / validation and error handling cases.
- ▶ Both the CI / CD and Kubernetes project are provided as oper source contribution. The Presentation was written in IATEX

Key points repetition

#### Summary

- ► In section "Introduction" page "2" Container Run Time Interfaces.
- ▶ In section "CI / CD" page "3" Problems / Desires / Solutions on CI / CD.
- In section "Implementation" page "4" CI / CD Flow Build / Deploy / Test.

- Demo on CI / CD build / deploy / validation and error handling cases.
- ▶ Both the CI / CD and Kubernetes project are provided as open source contribution. The Presentation was written in LATEX

ntents Introduction CI/CD Implementation Architecture Demo **Summary** Bibliography  $\circ$   $\circ$   $\circ$   $\bullet$   $\circ$ 

Key points repetition

#### Summary

- ► In section "Introduction" page "2" Container Run Time Interfaces.
- ▶ In section "CI / CD" page "3" Problems / Desires / Solutions on CI / CD.
- In section "Implementation" page "4" CI / CD Flow Build / Deploy / Test.

- Demo on CI / CD build / deploy / validation and error handling cases.
- ▶ Both the CI / CD and Kubernetes project are provided as open source contribution. The Presentation was written in LATEX

Summary

Key points repetition

#### Summary

- ▶ In section "Introduction" page "2" Container Run Time Interfaces.
- ► In section "CI / CD" page "3" Problems / Desires / Solutions on CI / CD.
- ▶ In section "Implementation" page "4" CI / CD Flow Build / Deploy / Test.

- Demo on Cl / CD build / deploy / validation and error handling cases.
- ▶ Both the CI / CD and Kubernetes project are provided as open source contribution. The Presentation was written in LATEX

Questions

# Coming up: Q & A

Web and Articles

### References I



## GNU LESSER GENERAL PUBLIC LICENSE

GNU Operating System

available at https://www.gnu.org/licenses/lgpl.html.



Author: K. Community

Container Runtimes

available at https://kubernetes.io/docs/setup/production-environment/container-runtimes/.



Author: K. Community

Cluster Networking

available at https://kubernetes.io/docs/concepts/cluster-administration/networking/.