



kubernetes

Kubernetes Vagrant CI / CD

GitHub Project Repo: [kubernetesVagrant](#)

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

October 3, 2021

Introduction

Kubernetes Elements

CI / CD

Project smooth CI / CD

How to accomplish
requirements

The correct tool of choice

Implementation

CI-CD Local

CI-CD Kubernetes

Architecture

Minimal Kubernetes

Components

My view of Kubernetes

Minimal Core Components

Demo

Demo CI / CD

Summary

Key points repetition

Bibliography

<i>Introduction</i>	<i>Architecture</i>
Kubernetes Elements	<i>Demo</i>
<i>CI / CD</i>	<i>Summary</i>
<i>Implementation</i>	<i>Bibliography</i>

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

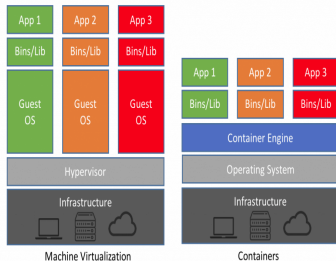


Figure 1: k8s Overview

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 inside Pod

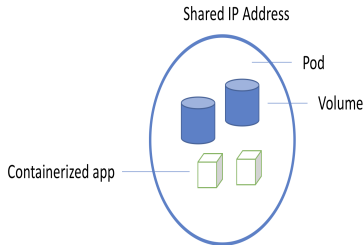


Figure 1: k8s Overview

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?

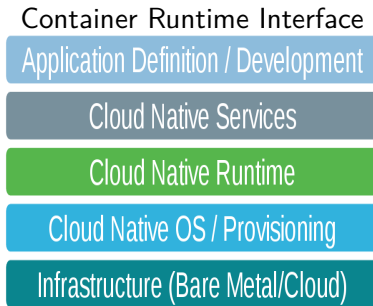


Figure 1: k8s Overview

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

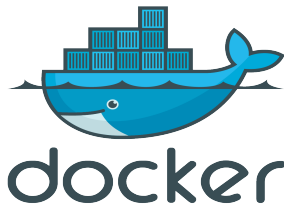


Figure 1: k8s Overview

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



podman



buildah

Figure 1: k8s Overview

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

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?

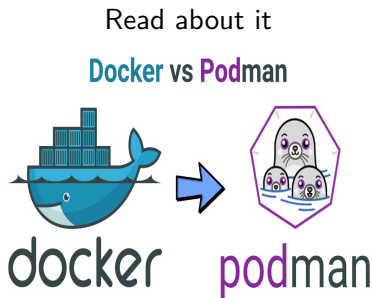


Figure 1: k8s Overview

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

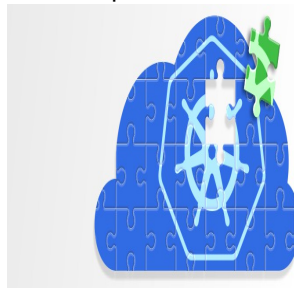


Figure 1: k8s Overview

Introduction

CI / CD

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

Implementation

Architecture

Demo

Summary

Bibliography

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

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

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

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

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

Solutions to Requirements

- ▶ 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).

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

Solutions to Requirements

- ▶ 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).

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

Solutions to Requirements

- ▶ 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).

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**

Solutions to Requirements

- ▶ 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).

Introduction

CI / CD

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

Implementation

Architecture

Demo

Summary

Bibliography

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).

Solutions

- ▶ 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!**

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).

Solutions

- ▶ **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!**

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).

Solutions

- ▶ **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!**

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).

Solutions

- ▶ **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!**

Introduction

CI / CD

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

Implementation

Architecture

Demo

Summary

Bibliography

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?

Possible Answers

- ▶ **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.

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?

Possible Answers

- ▶ 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.

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?

Possible Answers

- ▶ 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.

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?

Possible Answers

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

Architecture

Introduction

Demo

CI / CD

Summary

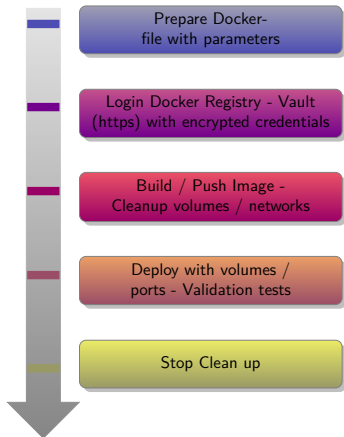
Implementation

Bibliography

CI-CD Local

CI-CD Kubernetes

Local Procedure



► Dockerfile (template).

► Vault (<https>).

► Any socket.

► Build / Push Image - Cleanup volumes / networks.

► Deploy with volumes / ports - Validation tests.

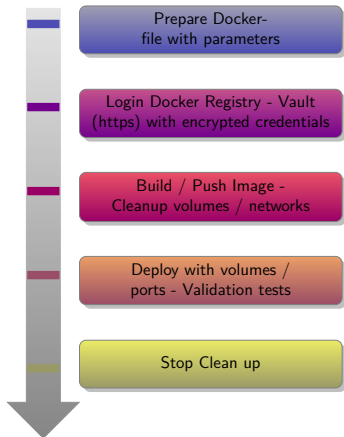
► Stop Clean up.

► Deployment (volume).

► Validation (tests).

► Stop, Cleanup.

Local Procedure



► Dockerfile (template).

► Vault (<https>).

► Any socket.

► Azure Registry.

► Build Dockerfile

► Push Image

► Login Docker

► Run container

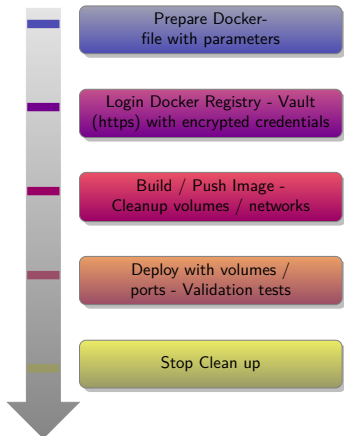
► Stop container

► Deployment (volume).

► Validation (tests).

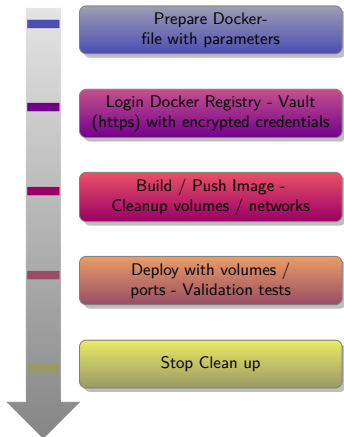
► Stop, Cleanup.

Local Procedure



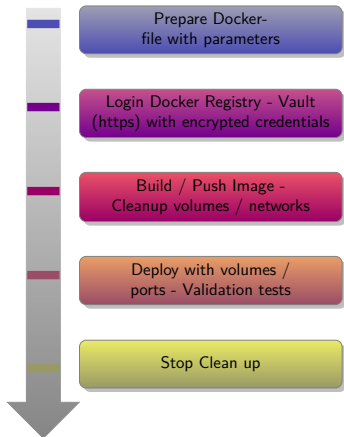
- ▶ Dockerfile (template).
- ▶ Vault (https).
- ▶ **Any socket.**
 - ▶ Azzure Registry.
 - ▶ Build Dockerfile.
 - ▶ Push Image.
 - ▶ Logout Azzure.
 - ▶ Prune everything.
 - ▶ Raise error (if).
- ▶ Deployment (volume).
- ▶ Validation (tests).
- ▶ Stop, Cleanup.

Local Procedure



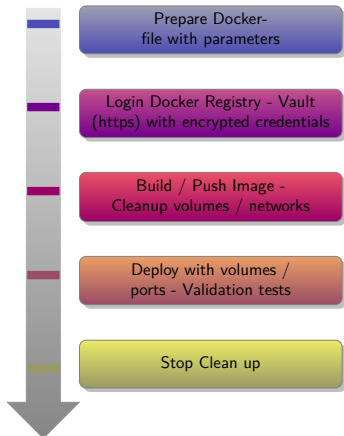
- ▶ Dockerfile (template).
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ **Azzure Registry.**
 - ▶ Build Dockerfile.
 - ▶ Push Image.
 - ▶ Logout Azzure.
 - ▶ Prune everything.
 - ▶ Raise error (if).
- ▶ Deployment (volume).
- ▶ Validation (tests).
- ▶ Stop, Cleanup.

Local Procedure



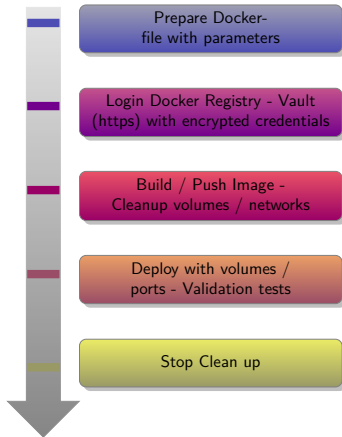
- ▶ Dockerfile (template).
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azure Registry.
 - ▶ **Build Dockerfile.**
 - ▶ Push Image.
 - ▶ Logout Azure.
 - ▶ Prune everything.
 - ▶ Raise error (if).
- ▶ Deployment (volume).
- ▶ Validation (tests).
- ▶ Stop, Cleanup.

Local Procedure



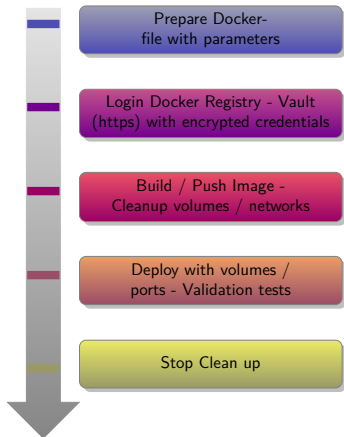
- ▶ Dockerfile (template).
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azure Registry.
 - ▶ Build Dockerfile.
 - ▶ **Push Image.**
 - ▶ Logout Azure.
 - ▶ Prune everything.
 - ▶ Raise error (if).
- ▶ Deployment (volume).
- ▶ Validation (tests).
- ▶ Stop, Cleanup.

Local Procedure



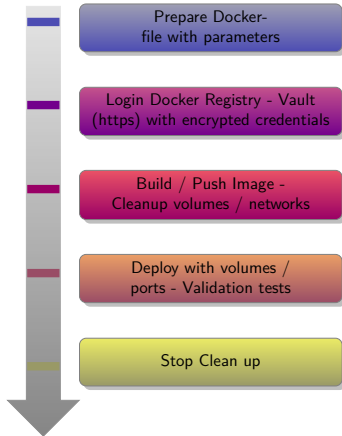
- ▶ Dockerfile (template).
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Build Dockerfile.
 - ▶ Push Image.
 - ▶ Logout Azzure.
 - ▶ Prune everything.
 - ▶ Raise error (if).
- ▶ Deployment (volume).
- ▶ Validation (tests).
- ▶ Stop, Cleanup.

Local Procedure



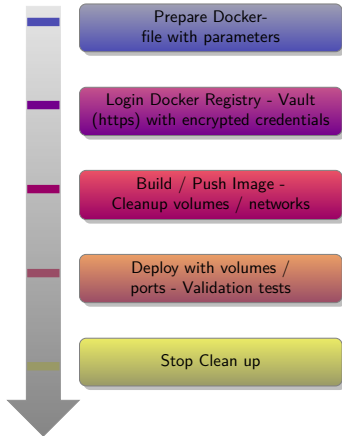
- ▶ Dockerfile (template).
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Build Dockerfile.
 - ▶ Push Image.
 - ▶ Logout Azzure.
 - ▶ **Prune everything.**
 - ▶ Raise error (if).
- ▶ Deployment (volume).
- ▶ Validation (tests).
- ▶ Stop, Cleanup.

Local Procedure



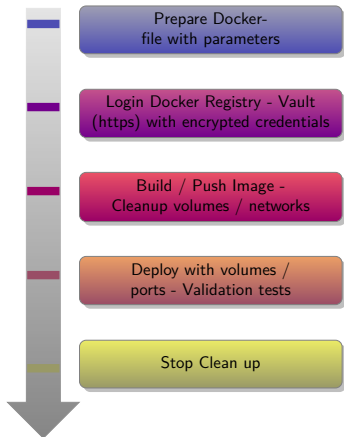
- ▶ Dockerfile (template).
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Build Dockerfile.
 - ▶ Push Image.
 - ▶ Logout Azzure.
 - ▶ Prune everything.
 - ▶ Raise error (if).
- ▶ Deployment (volume).
- ▶ Validation (tests).
- ▶ Stop, Cleanup.

Local Procedure



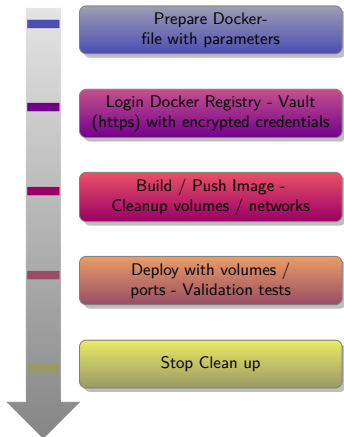
- ▶ Dockerfile (template).
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Build Dockerfile.
 - ▶ Push Image.
 - ▶ Logout Azzure.
 - ▶ Prune everything.
 - ▶ Raise error (if).
- ▶ Deployment (volume).
- ▶ Validation (tests).
- ▶ Stop, Cleanup.

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.

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.

Architecture

Introduction

Demo

CI / CD

Summary

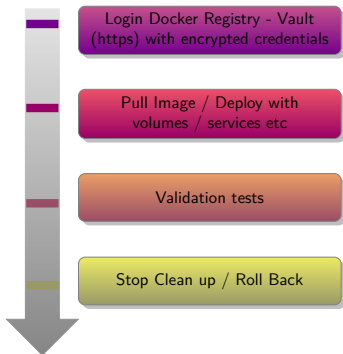
Implementation

CI-CD Local

CI-CD Kubernetes

Bibliography

Local k8s Deployment Procedure



► Vault (https).

► Any socket.

► Azure Registry.

► Pull Image.

► Deployment (volume).

► Logout Azzure.

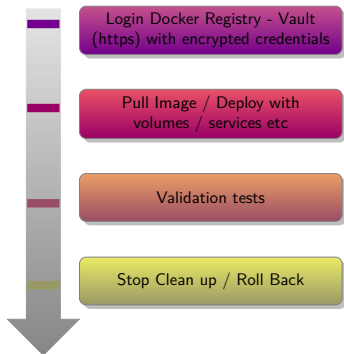
► Validation (tests).

► Roll back (if).

► Stop, Clean up.

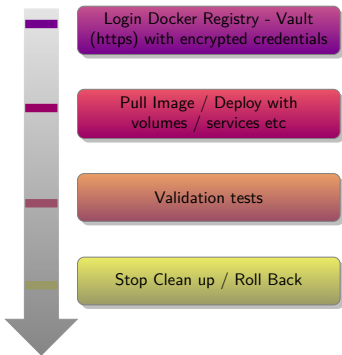
► Roll back.

Local k8s Deployment Procedure



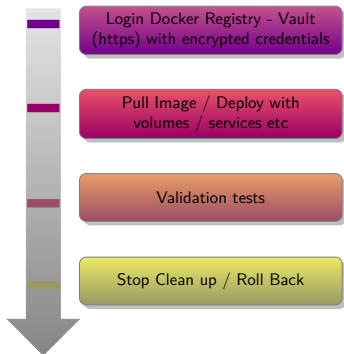
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Pull Image.
- ▶ Deployment (volume).
- ▶ Logout Azzure.
- ▶ Validation (tests).

Local k8s Deployment Procedure



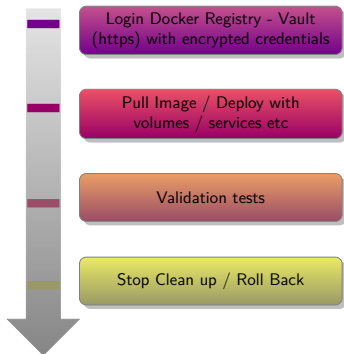
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Pull Image.
- ▶ Deployment (volume).
- ▶ Logout Azzure.
- ▶ Validation (tests).

Local k8s Deployment Procedure



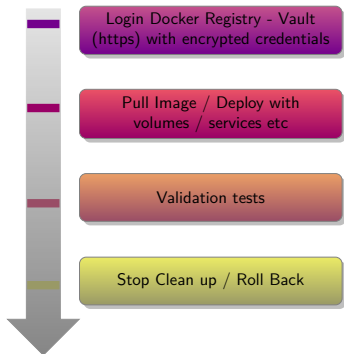
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Pull Image.
- ▶ Deployment (volume).
- ▶ Logout Azzure.
- ▶ Validation (tests).

Local k8s Deployment Procedure



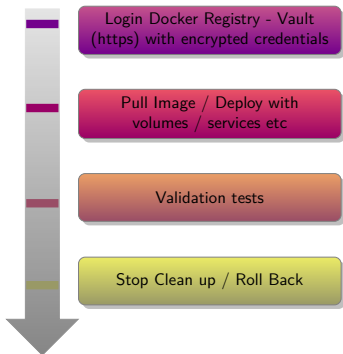
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Pull Image.
- ▶ Deployment (volume).
- ▶ Logout Azzure.
- ▶ Validation (tests).

Local k8s Deployment Procedure



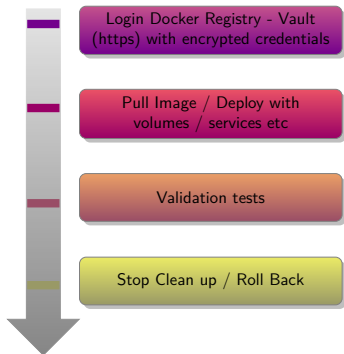
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Pull Image.
- ▶ Deployment (volume).
- ▶ Logout Azzure.
- ▶ Validation (tests).
 - ▶ Raise error (if).

Local k8s Deployment Procedure



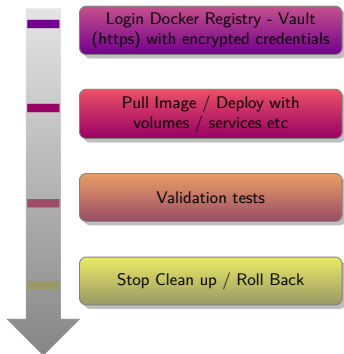
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Pull Image.
- ▶ Deployment (volume).
- ▶ Logout Azzure.
- ▶ **Validation (tests).**
 - ▶ Raise error (if).
 - ▶ Stop, Cleanup.
 - ▶ Roll back.

Local k8s Deployment Procedure



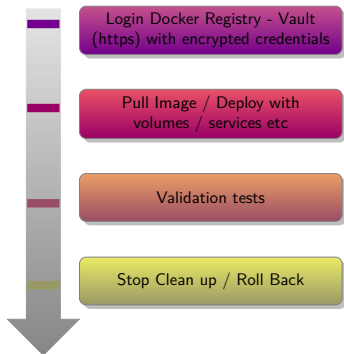
- ▶ Vault (https).
- ▶ Any socket.
 - ▶ Azzure Registry.
 - ▶ Pull Image.
- ▶ Deployment (volume).
- ▶ Logout Azzure.
- ▶ Validation (tests).
 - ▶ Raise error (if).
 - ▶ Stop, Cleanup.
 - ▶ Roll back.

Local k8s Deployment Procedure



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

Local k8s Deployment Procedure



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

Introduction

CI / CD

Implementation

Architecture

Minimal Kubernetes

Components

My view of Kubernetes

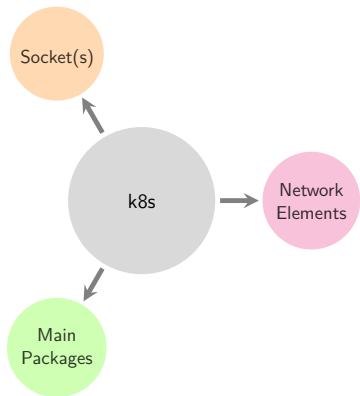
Minimal Core Components

Demo

Summary

Bibliography

Kubernetes Minimal Core Components



► Socket:

- Containerd (Docker)
- CRI-O

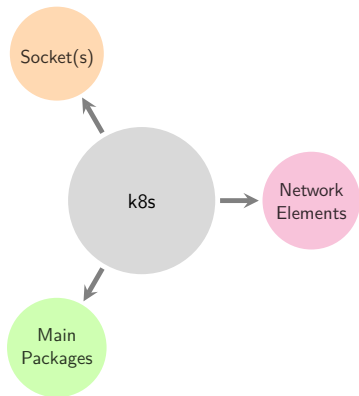
► Main Packages:

- kubelet
- kubectl
- kubeadm

► Network Elements

- Calico [1]
- WeaveNet [2]
- Flannel [3]

Kubernetes Minimal Core Components



► Socket:

► Containerd (Docker)

► CRI-O

► Main Packages:

► kubelet

► kubectl

► kubeadm

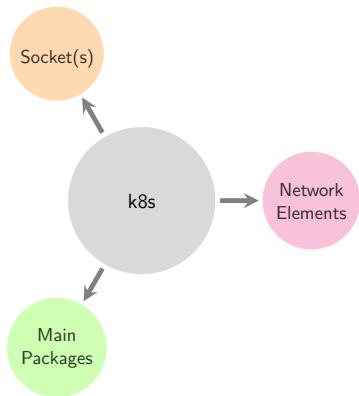
► Network Elements

► Calico [1]

► WeaveNet [2]

► Flannel [3]

Kubernetes Minimal Core Components



- ▶ Socket:
 - ▶ Containerd (Docker)
 - ▶ CRI-O

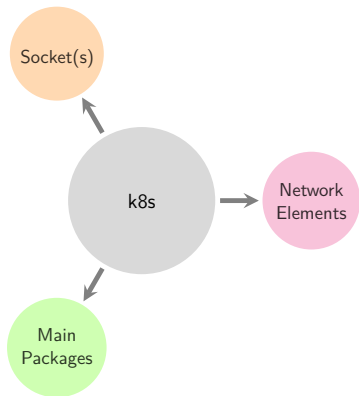
▶ Main Packages:

- ▶ kubeadm
- ▶ kubectl
- ▶ kubeadm

▶ Network Elements

- ▶ Calico [1]
- ▶ WeaveNet [2]
- ▶ Flannel [3]

Kubernetes Minimal Core Components



- ▶ Socket:
 - ▶ Containerd (Docker)
 - ▶ CRI-O

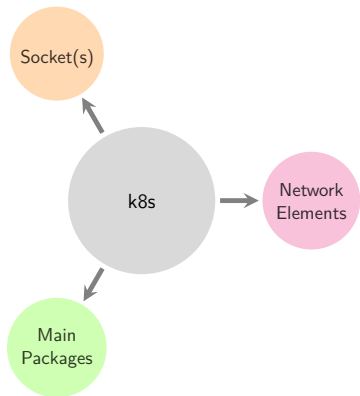
▶ Main Packages:

- ▶ kubeadm
- ▶ kubelet
- ▶ kubectl

▶ Network Elements

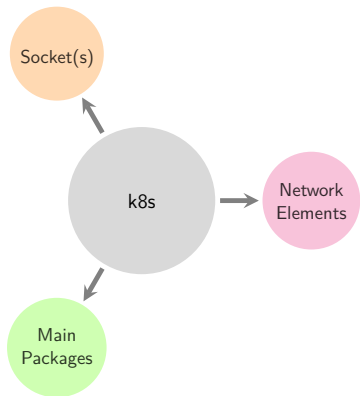
- ▶ Calico [1]
- ▶ WeaveNet [2]
- ▶ Cilium [3]

Kubernetes Minimal Core Components



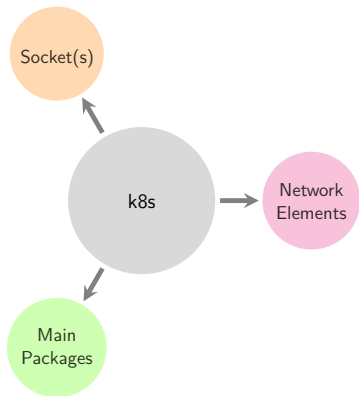
- ▶ Socket:
 - ▶ Containerd (Docker)
 - ▶ CRI-O
- ▶ Main Packages:
 - ▶ **kubeadm**
 - ▶ kubelet
 - ▶ kubectl
- ▶ Network Elements
 - ▶ Calico [1]
 - ▶ Weave [2]
 - ▶ Flannel [3]
 - ▶ Cilium [4]

Kubernetes Minimal Core Components



- ▶ Socket:
 - ▶ Containerd (Docker)
 - ▶ CRI-O
- ▶ Main Packages:
 - ▶ kubeadm
 - ▶ **kubelet**
 - ▶ kubectrl
- ▶ Network Elements

Kubernetes Minimal Core Components

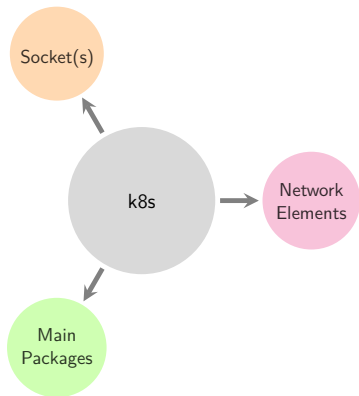


- ▶ Socket:
 - ▶ Containerd (Docker)
 - ▶ CRI-O
- ▶ Main Packages:
 - ▶ kubeadm
 - ▶ kubelet
 - ▶ **kubectl**

▶ Network Elements

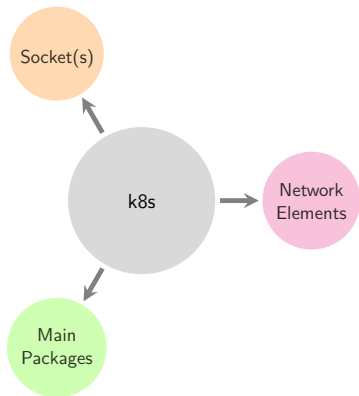
- ▶ Calico [1]
- ▶ Weave [2]
- ▶ Flannel [3]

Kubernetes Minimal Core Components



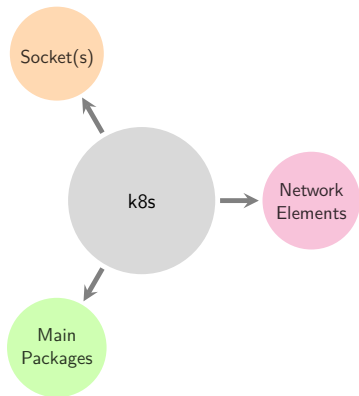
- ▶ Socket:
 - ▶ Containerd (Docker)
 - ▶ CRI-O
- ▶ Main Packages:
 - ▶ kubeadm
 - ▶ kubelet
 - ▶ kubectl
- ▶ Network Elements
 - ▶ Calico [1]
 - ▶ WeaveNet [1]
 - ▶ Cilium [1]

Kubernetes Minimal Core Components



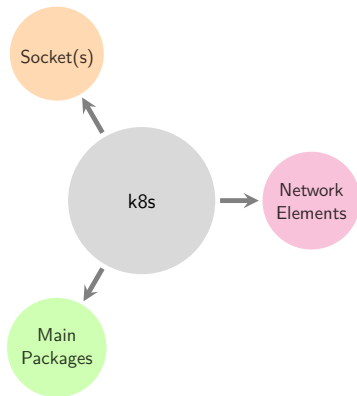
- ▶ Socket:
 - ▶ Containerd (Docker)
 - ▶ CRI-O
- ▶ Main Packages:
 - ▶ kubeadm
 - ▶ kubelet
 - ▶ kubectl
- ▶ Network Elements
 - ▶ Calico [1]
 - ▶ WeaveNet [1]
 - ▶ Cilium [1]

Kubernetes Minimal Core Components



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

Kubernetes Minimal Core Components



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

Introduction

CI / CD

Implementation

Architecture

Minimal Kubernetes

Components

My view of Kubernetes

Minimal Core Components

Demo

Summary

Bibliography

Local Auto Deployment

► Previous Elements

► Ingress Controller:

► NGINX [3]

► HAProxy [4]

► Metrics (HPA)

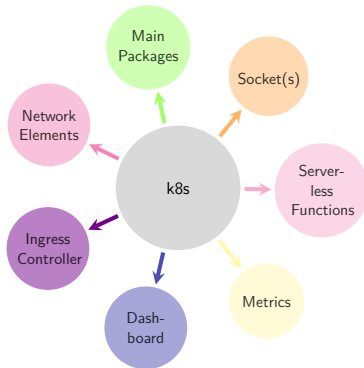
► Dashboard

► <https://kubernetes.io/docs/tasks/debug-application-cluster/kubectl-1.10/#kubectl-explain>

► Serveless Functions, Functions as a Service (FaaS)

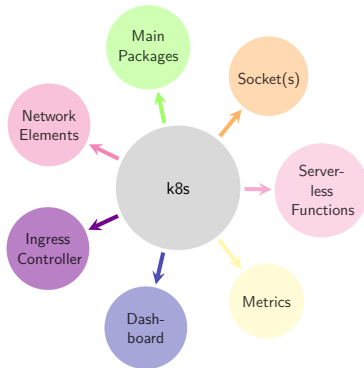
► Operatin System:

► <https://kubernetes.io/docs/tasks/debug-application-cluster/kubectl-1.10/#kubectl-explain>



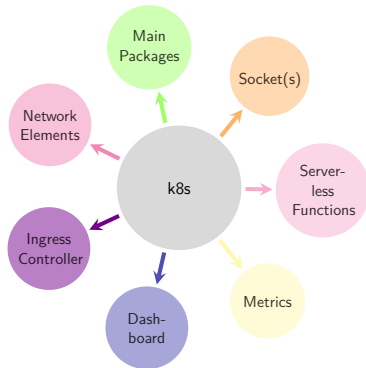
Local Auto Deployment

- ▶ Previous Elements
- ▶ **Ingress Controller:**
 - ▶ NGINX [3]
 - ▶ HaProxy [3]
- ▶ Metrics (HPA)
- ▶ Dashboard
- ▶ **Network Elements**
- ▶ **Serverless Functions, Functions as a Service (FaaS)**
- ▶ **Operating System:**



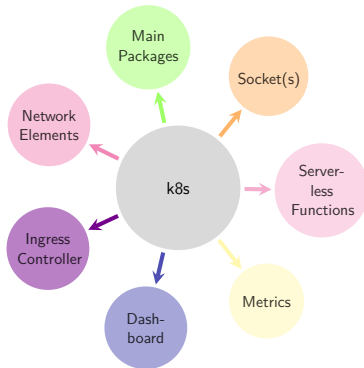
Local Auto Deployment

- ▶ Previous Elements
- ▶ Ingress Controller:
 - ▶ NGINX [3]
 - ▶ HaProxy [3]
- ▶ Metrics (HPA)
- ▶ Dashboard
- ▶ Network Elements
- ▶ Main Packages
- ▶ Socket(s)
- ▶ Serverless Functions
- ▶ Metrics
- ▶ Ingress Controller
- ▶ Dashboard
- ▶ Metrics
- ▶ Serveless Functions, Functions as a Service (FaaS)
- ▶ Operatin System:



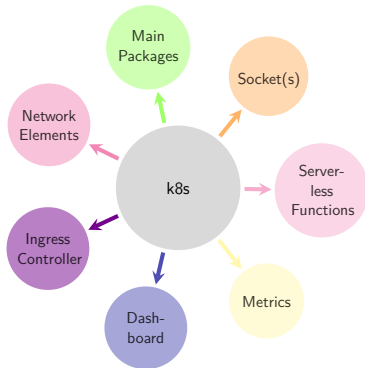
Local Auto Deployment

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



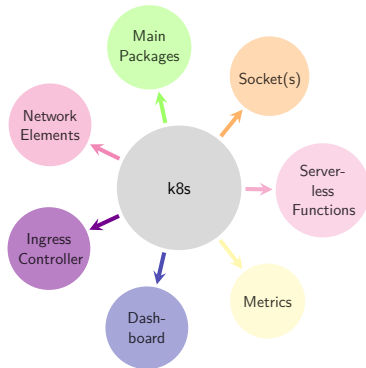
Local Auto Deployment

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



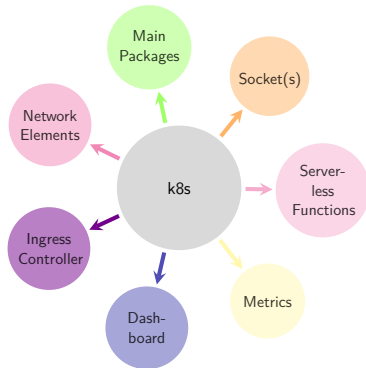
Local Auto Deployment

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



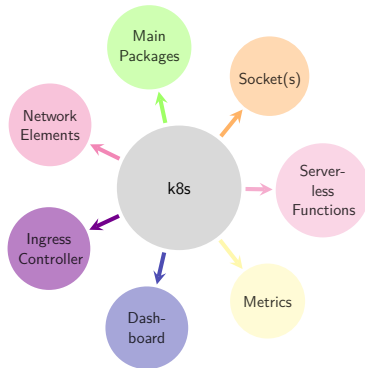
Local Auto Deployment

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



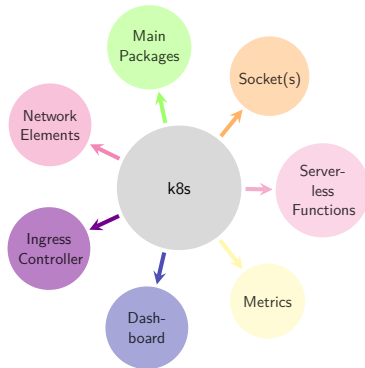
Local Auto Deployment

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



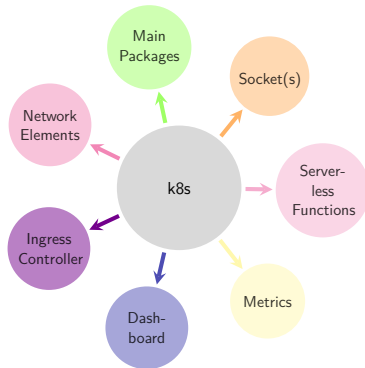
Local Auto Deployment

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



Architecture

Introduction

Demo

Demo CI / CD

CI / CD

Summary

Implementation

Bibliography

Coming up: Demo

Architecture

Introduction

Demo

CI / CD

Summary

Key points repetition

Implementation

Bibliography

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.

Future Notes

- ▶ Summary on CI / CD Build / Deploy / Validation and Release.
- ▶ Summary on CI / CD and Kubernetes and how to build a container image.
- ▶ Summary on “Future” Page “5” and how to build a container image.

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.

Future Notes

- ▶ 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.

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.

Future Work

- * Demo on CI / CD build / deploy / validation and error handling cases.

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.

Future Work

- ▶ 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 *RTX*.

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

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 **L^AT_EX**

Coming up: Q & A

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/>.