SUSE

Introduction to containers

All the way from Docker to Kubernetes cluster based on k3s

Our agenda *

1. **Part One**
   Introduction to containers with Docker
2. **Part Two**
   Getting to know Kubernetes with k3s
3. **Part Two ***
   Kubernetes cluster with k3s

Introduction to containers with Docker

Get on speed with containers and Docker

Containerization

OS-level virtualization refers to an operating system paradigm in which the kernel allows the existence of **multiple isolated user space instances** known as **containers**, **zones**, **jails**, **partitions**, etc.

https://en.wikipedia.org/wiki/OS-level_virtualization

Container Types and Solutions

- System Containers
  - OS-centric
  - Multiple processes
  - LXC (+ LXD + LXCF) by Canonical
  - https://linuxcontainers.org
- Application Containers
  - App-centric
  - Single process *
  - Docker by Docker Inc
  - https://www.docker.com

Virtual Machines vs Containers

Virtual Machines

- Virtualize the hardware
- Complete isolation
- Complete OS installation
- Require more resources
- Run almost any OS

Containers

- Virtualize the OS
- Lightweight isolation
- Shared kernel
- Require fewer resources
- Run on the same OS

Definitions

- **Image**

  Read-only template build from layers

- **Container**

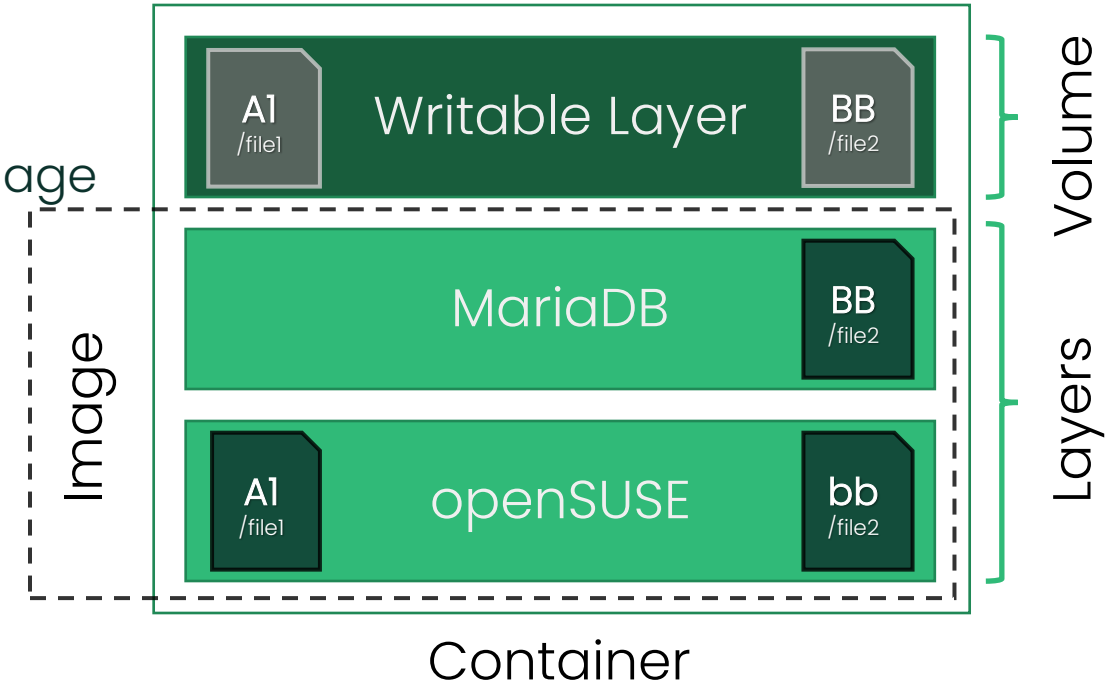
  Runnable instance of an image

- **Repository**

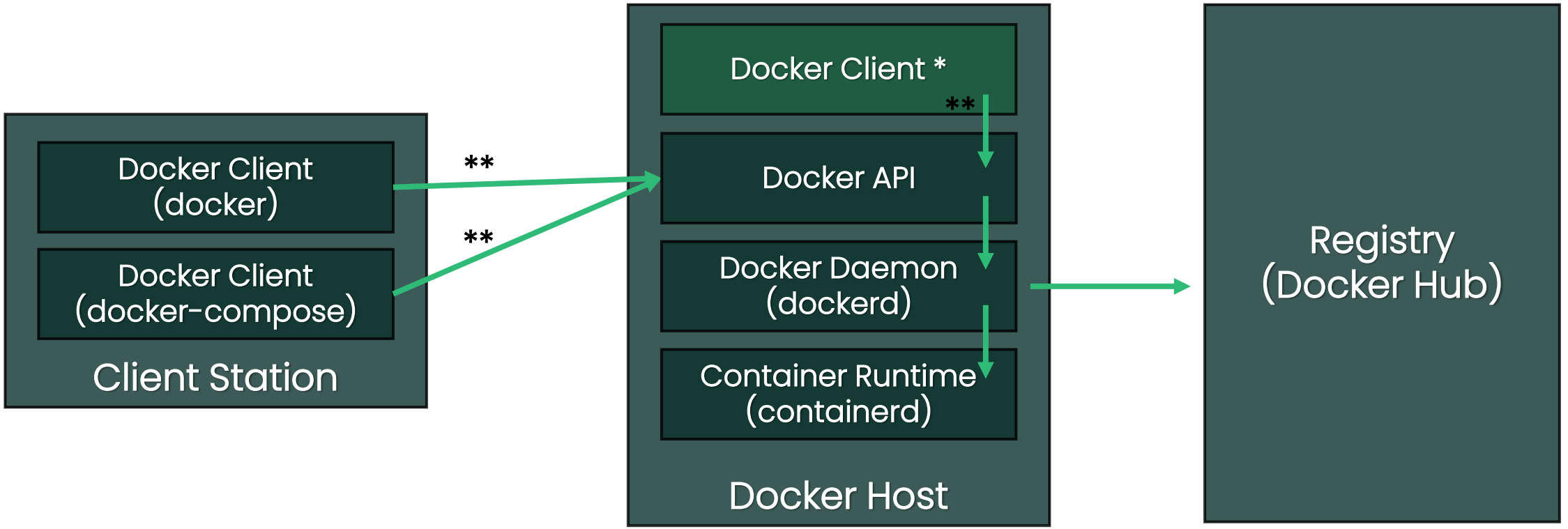
  Collection of different versions (tags) of an image

- **Registry**

  Collection of repositories

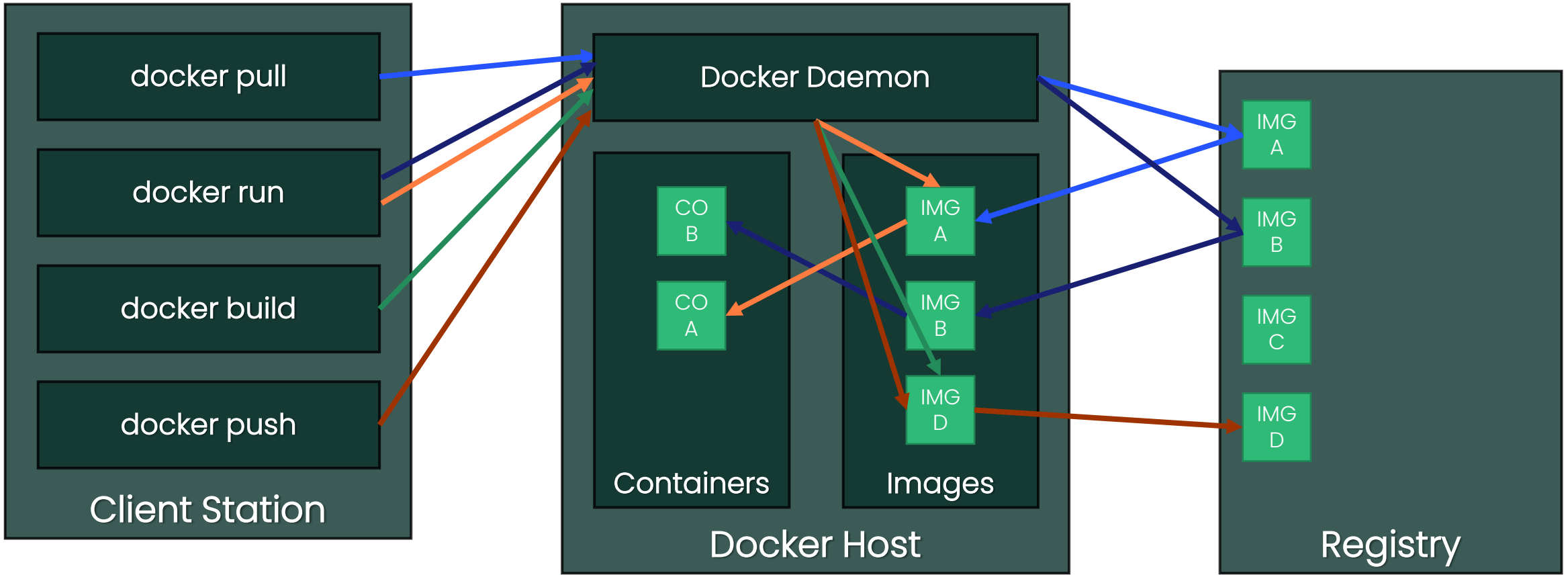


Docker Platform



* Not required, but we could install it there as well
** TCP or UNIX socket

Workflow



https://docs.docker.com/get-started/overview/

Dockerfile

- Script that contains the steps to produce a container image
- It contains various instructions and arguments
- Always begins with **FROM** instruction
- Comments start with **#**

```
# Set the base image
FROM nginx

# Set the author
LABEL author="John Smith <js@xyz.co>"

# Copy files
COPY index.html /usr/share/nginx/html/
```

https://docs.docker.com/engine/reference/builder

Demo Time

Let’s see it in action

Getting to know Kubernetes with k3s

Switch to Kubernetes with k3s

New Demands

- Workload deployment and distribution
- Resource governance
- Scalability and availability
- Automatization and management
- Internal and external communication

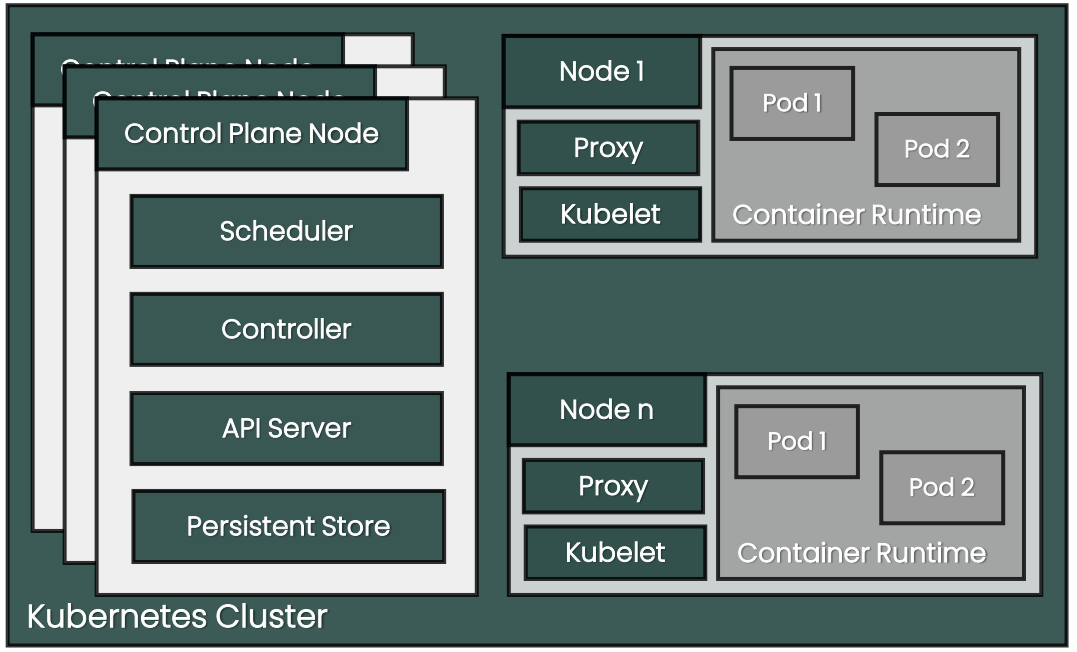
Container Orchestration

Kubernetes is the Answer

- What does it offer?
  - Runs a cluster of hosts
  - Schedules containers to run on different hosts
  - Facilitates the communication between the containers
  - Provides and controls access to/from outside world
  - Tracks and optimizes the resource usage
- Where it came from?
  - Born out of projects like **Borg** and **Omega** at **Google**. Written in **Go**
  - The name comes from the Greek word **κυβερνήτης** and means **Helmsman**
  - It is shortened often to **k8s**
- Is it the only one?
  - Other solutions include **Docker Swarm**, **HachiCorp Nomad**, **Apache Mesos + Marathon**

Kubernetes Architecture

- **Control plane** nodes are responsible for managing the cluster
  - Usually, more than one is installed, and they are work-free
  - **Persistent Sore** contains cluster state and configuration
  - **API Server** is the front-end of control plane
  - **Controller** maintains the desired state
  - **Scheduler** assigns work to nodes
- Nodes handle the actual work
  - Proxy provides the networking
  - Kubelet agent talks to the control plane
  - Container Runtime works with images and containers



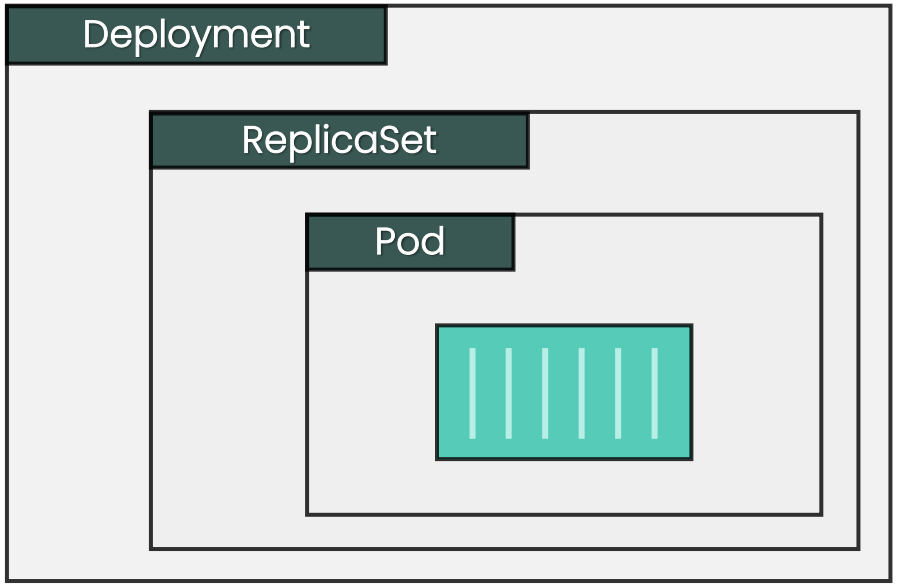
Pods, Services, and Labels

- **Pods** are the **smallest unit of scheduling**
  - Scheduled on nodes and consist of **one or more containers** with **shared environment**
  - They are **atomic** and are **deployed as one** and on **one node**
  - Each pod has **unique address**
- **Services** provide **reliable network endpoint** (**DNS name**, **IP address**, and **port**)
  - Expose pods to the outside world and use end points to track them
  - They can be **ClusterIP**, **NodePort**, **LoadBalancer**, and **ExternalName**
  - Use **label selectors** to select the pods
- Labels are **key-value pairs** attached to objects and are used to select group of objects
  - Each object my have multiple labels
  - Each label may be attached to multiple objects

Deployments and Replica Sets

- **ReplicaSets** are higher-level workload compared to the pods
  - They look after pod or set of pods
  - Facilitate the **scaling** both **up** and **down** of pods

- **Deployments** are even higher-level workload
  - Simplify the process of update and rollback
  - Self documenting and suitable for versioning



SUSE Rancher Products

K3s and many more

Brief Overview

- **Rancher** makes the process of managing Kubernetes installed in your local or remote development environment a piece of cake
- **Hosted Rancher Service** is the fastest, safest, most cost-effective path to multi-cluster Kubernetes in your enterprise
- **RKE** is a lightning-fast, CNCF-certified Kubernetes distribution that runs entirely within containers and solves the common frustration of installation complexity
- **K3s** is a lightweight, certified Kubernetes distribution built for running production workloads inside IoT appliances or at the network edge
- **Longhorn** is 100% open source, distributed block storage built for Kubernetes

https://rancher.com/products

Why k3s?

- Perfect for Edge
  - K3s is a highly available, certified Kubernetes distribution designed for production workloads in unattended, resource-constrained, remote locations or inside IoT appliances
- Simplified and Secure
  - K3s is packaged as a single <50MB binary that reduces the dependencies and steps needed to install, run and auto-update
- Optimized for ARM
  - Both ARM64 and ARMv7 are supported. It works even on Raspberry Pi
- Simple and blazing fast installation
  - One command and less than 30 seconds are needed

https://k3s.io/

Demo Time

Let’s see it in action

Kubernetes cluster with k3s

Build a small k3s cluster

Requirements and Creation Process

- Requirements
  - The same as with the single node installation
- Creation Process
  - Create a set of virtual machines
  - Ensure they are part of the same network
  - Set names and IP addresses
  - Adjust the firewall
  - Deploy the control plane node(s)
  - Deploy the remaining node(s)

Demo Time

Let’s see it in action