



# Containers and Orchestration Oh My!

Frank Seesink, UNC Chapel Hill



First, a message from  
our sponsor...





What I picture in my head...





What it ends up looking like...



Actually that's not quite right. The guy who made this is **clearly** more talented.



# Who am I?

## Frank Seesink

- Senior Network Engineer, UNC Chapel Hill
- Part of network DevOps group
- Involved in network automation for years
- JOAT - databases, OSes, networking,...
- Using Docker since ~2014(?)
- Red Hat Open Shift, Rancher Desktop, K3s, Helm, Kompose, Lens
- Working towards GitOps



Story Time...





# Current Setup

**django**



Red Hat  
OpenShift

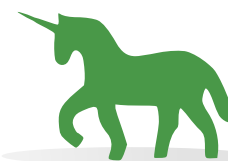
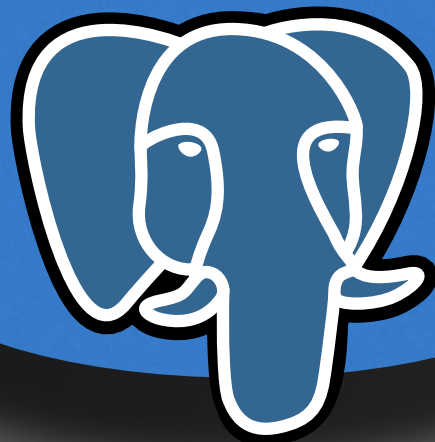
**NETM KO**

SQLite



**Nornir**

db



**gunicorn**

Work environment



# Current Setup



GitLab  
on-prem



Webhook



Repo

OCNES Dashboard Unreachables Recent Devices Admin AKIP

Refresh in 10

New Event Trends

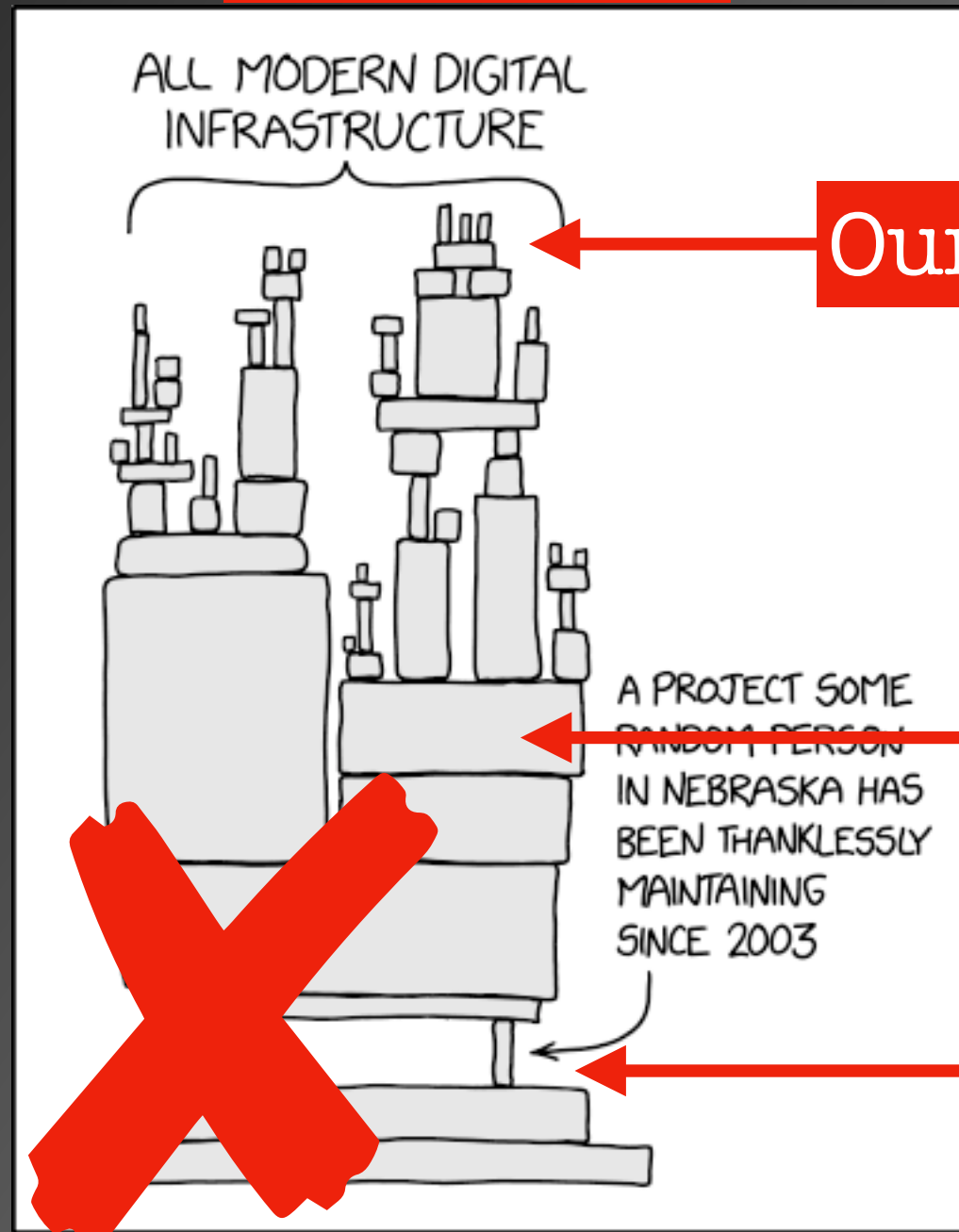
Ack	Tier1 / Building	Switch	AP	Unreachables	% Affected	Last Event	Trend	Incident
<input type="checkbox"/>	<a href="#">CraigN-Resnet-Agg-Sw</a>	2	0	0	0%	04-11 17:51:46		<input type="checkbox"/>
<input type="checkbox"/>	<a href="#">Ehringhaus-Residence-Hall</a>	1	0	0	0%	04-11 10:31:06		<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">Horton-Residence-Hall</a> INC0366426	1	0	0	0%	04-11 05:22:11		<a href="#">INC0366426</a>
<input checked="" type="checkbox"/>	<a href="#">MetroE-Tier1</a>	0	0	0	0%	04-11 05:22:22		<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">TEACCH-Ashville</a>	0	0	1	17%	04-11 05:22:11		<a href="#">INC0364710</a>

Ack	Device	Trap	Last	Incident	Clear
<input type="checkbox"/>	Core-P.unc.edu	OSPF-TRAP-MIB.ospfIfAuthFailure	04-11 17:51:46	<input type="checkbox"/>	Clear
<input type="checkbox"/>	Datacenter-M.net.unc.edu	IF-MIB.linkUp	04-11 17:49:58	<input type="checkbox"/>	Clear



# Current Setup

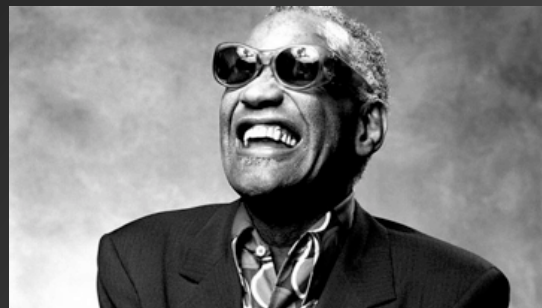
## Datacenter



Our Monitoring Apps



Potential issue



Us

<https://xkcd.com/2347>

# The Plan





# Out-Of-Band Server



## Server

Webserver (NGINX)

Django/Gunicorn

PostgreSQL

Celery

Redis

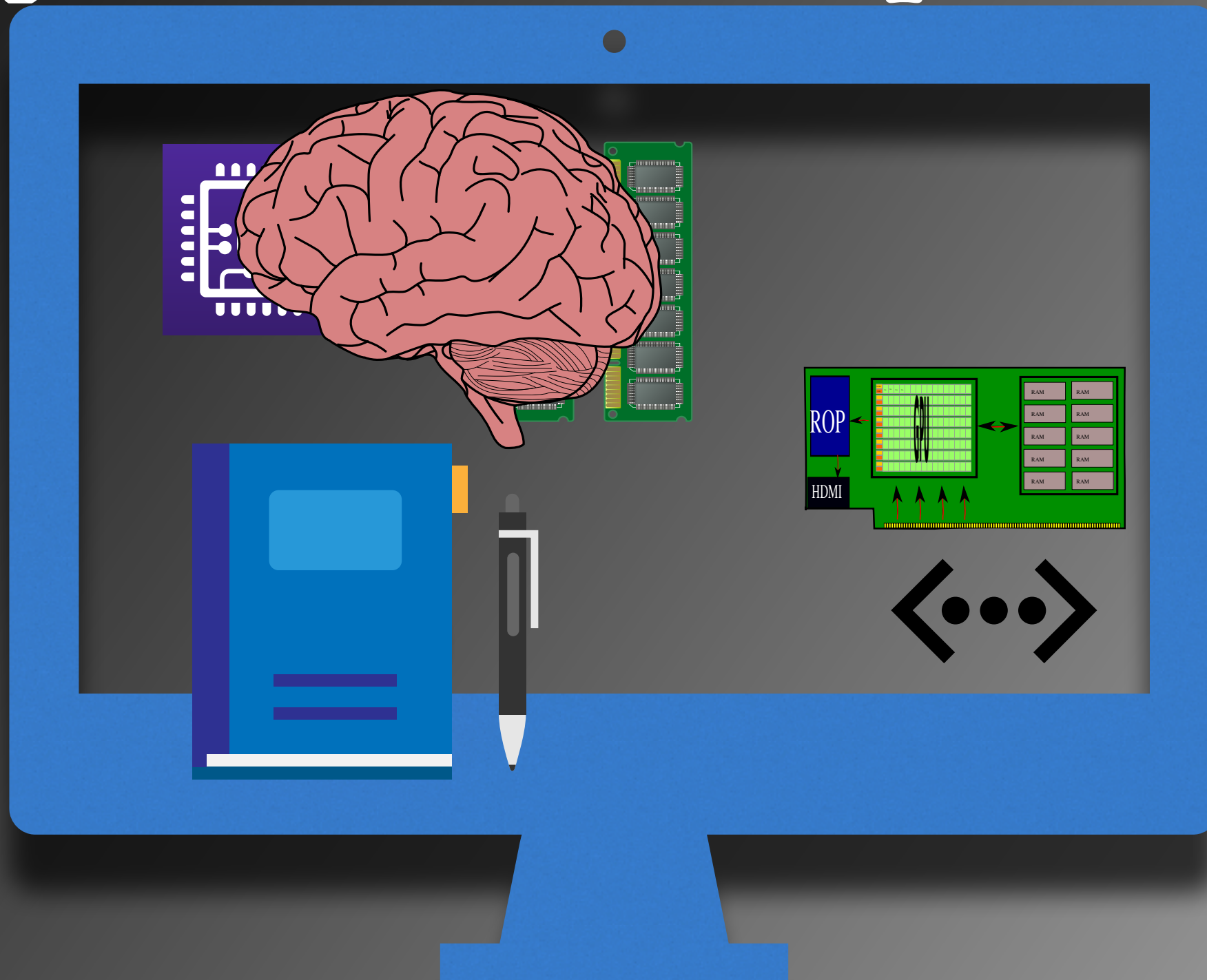


That's the backstory.  
Now a flashback...





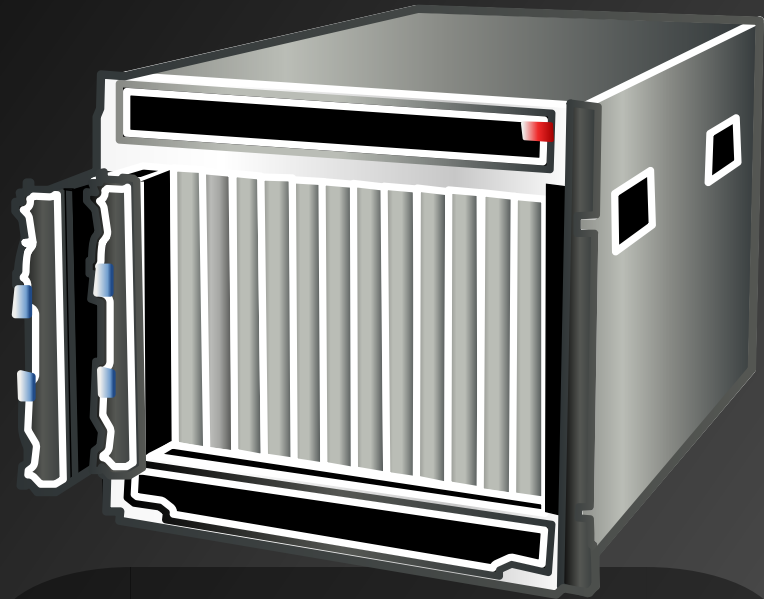
# In the beginning... you had a computer



# That computer ran an OS...



# Then came Virtual Machines (VMs)...



vmware®

<https://www.vmware.com/>



**XP**ROXMOX

<https://www.proxmox.com>



<https://www.virtualbox.org/>

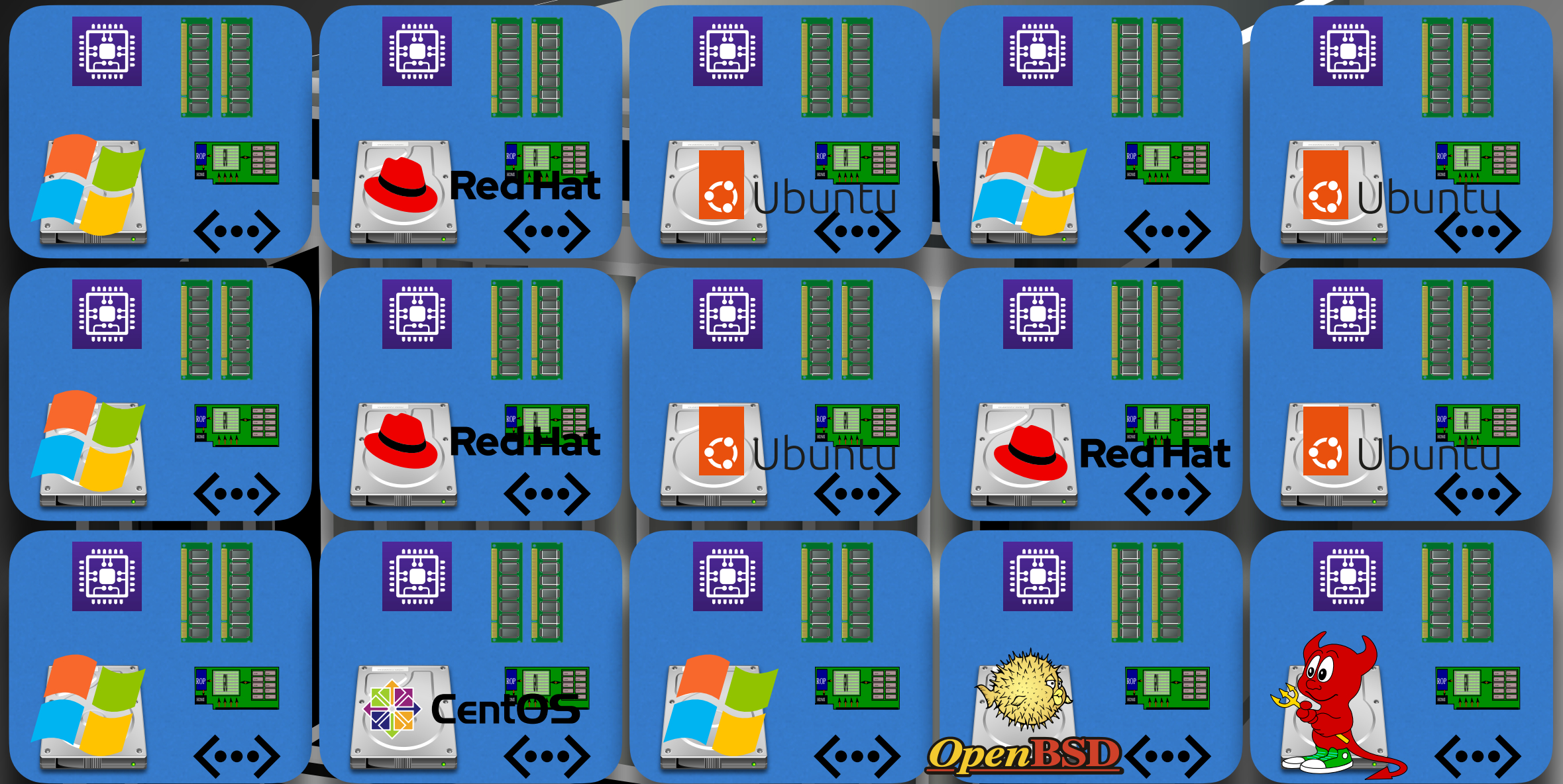
**|| Parallels™**

<https://www.parallels.com/>





# Then came Virtual Machines (VMs)...



Now what about  
containers?



# Container Terms

**container:** A container is a running process with resource and capability constraints managed by a computer's operating system. The files available to the container process are packaged as a container image. Containers run adjacent to each other on the same machine, but typically the operating system prevents the separate container processes from interacting with each other.

— <https://glossary.cncf.io/container/>





# Container Terms

**image:** a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

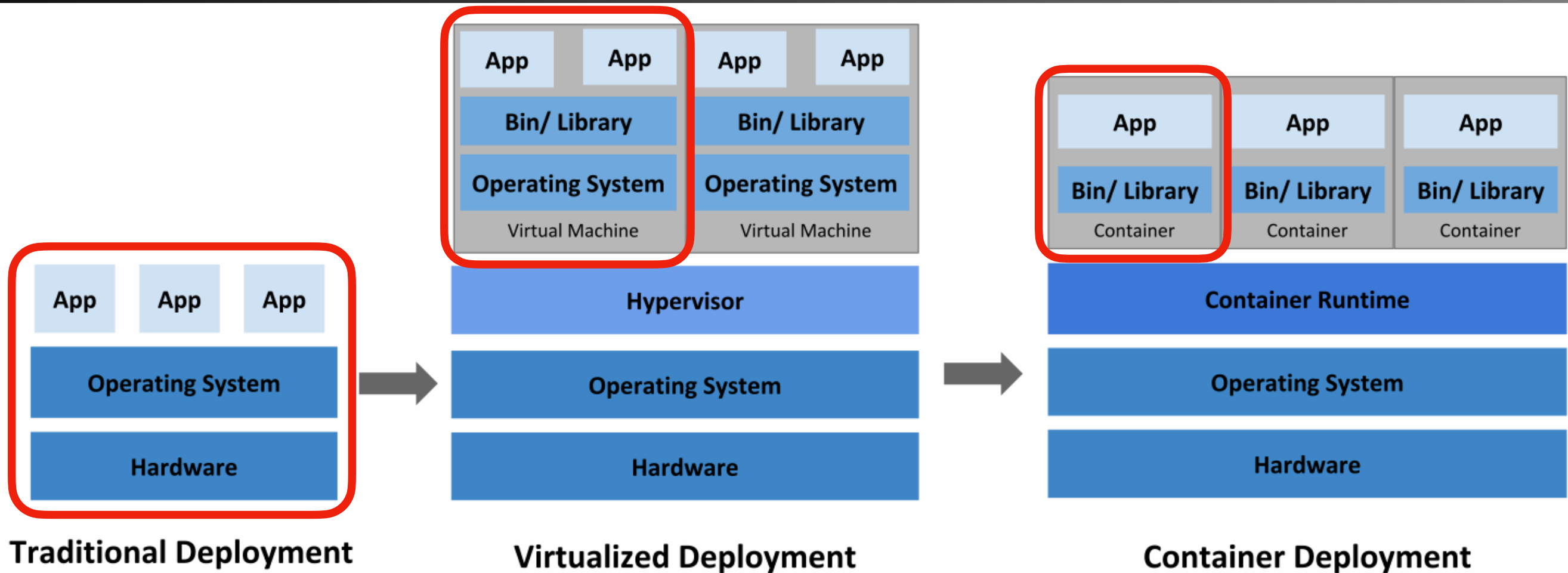
**ephemeral:** lasting for a very short time.

“Containers are ephemeral.”

“Containers should be ephemeral.”



# Short History

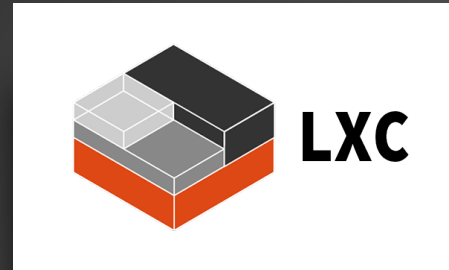


Installation time  
**hours/days**  
Startup time  
**minutes**

**hours/days**  
**minutes**

**minutes**  
**seconds**

# Container History



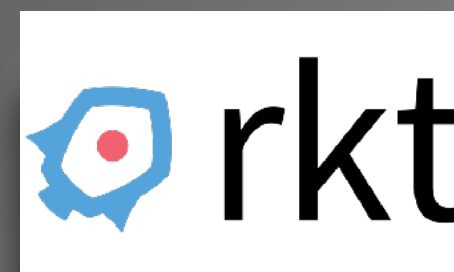
August 2008



March 2013



September 2014



December 2014



# Container History



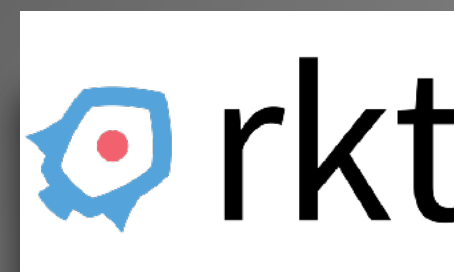
August 2008



March 2013



September 2014



December 2014





# Container History



# Container History



donated  
image format

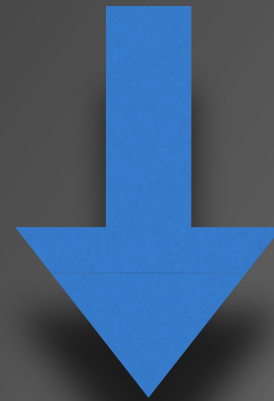


# Docker



To install on Linux:

- yum/dnf
- apt-get



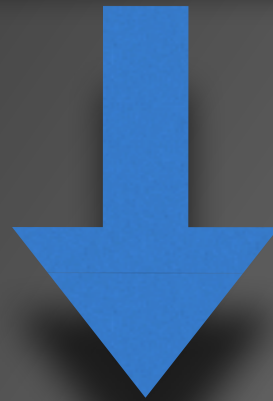
## Docker Desktop

- for Linux
- for macOS
- for Windows

<https://www.docker.com/products/docker-desktop/>



# Docker



## **Docker Desktop**

- for Linux
- for macOS
- for Windows





# Docker



What does it do?

- CLI driven
- Download/build/upload images
- Run/manage containers built from images
- Combine containers into apps using **Docker Compose**
- Run **Kubernetes!**  
(disabled by default)

## **Docker Desktop**

- for Linux
- for macOS
- for Windows



# Docker

How does it do it?

- Dockerd\* daemon
- '**docker**' CLI

```
docker run -it ubuntu:22.04 /bin/bash  
docker ps -a  
docker volume ls  
docker build ...
```

- Dockerfiles (YAML) to create images
  - Layers
    - e.g., alpine:latest + app
- Docker Compose files (YAML) to create multi-container apps



## Docker Desktop

- for Linux
- for macOS
- for Windows

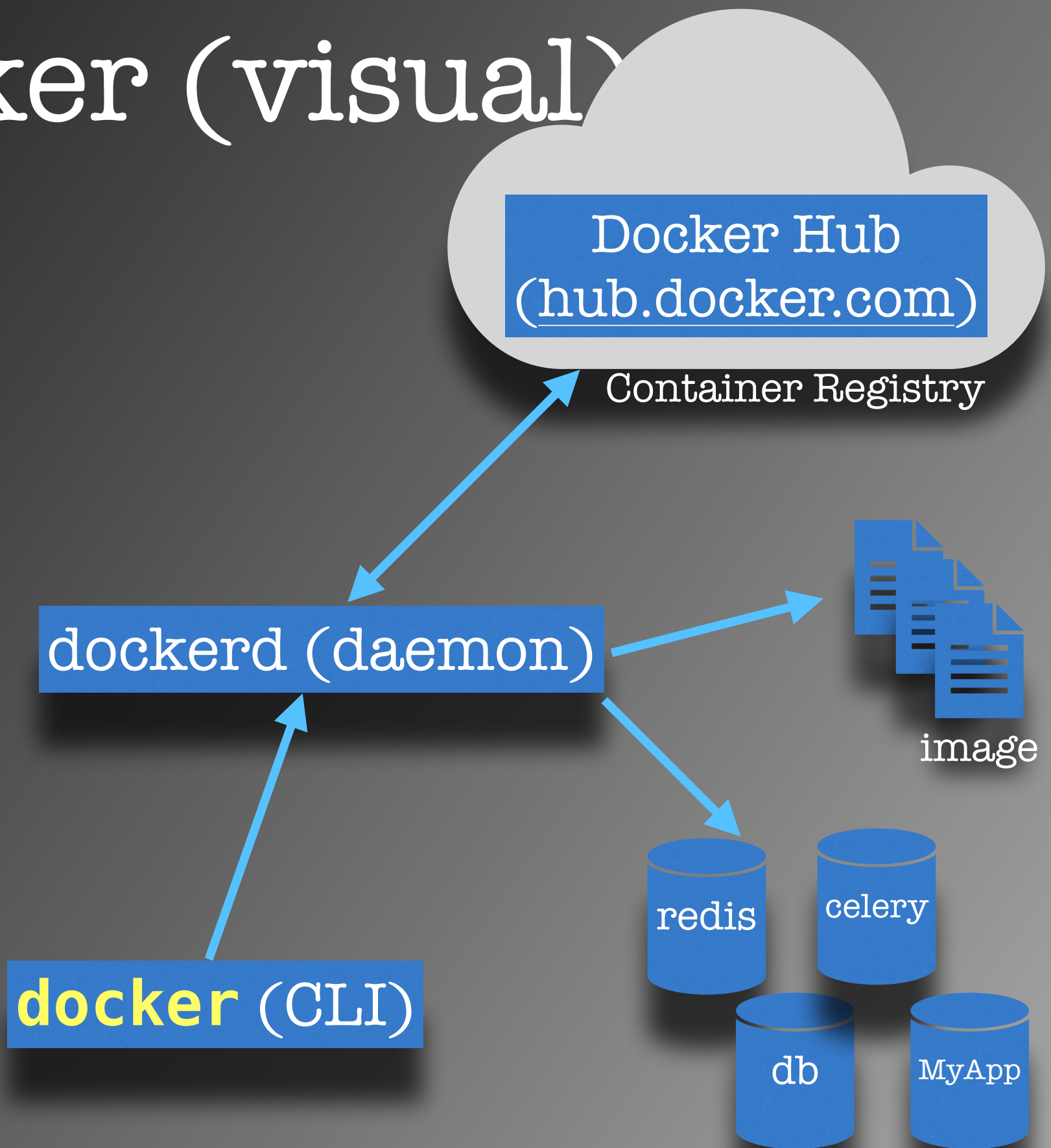


# Docker (visual)

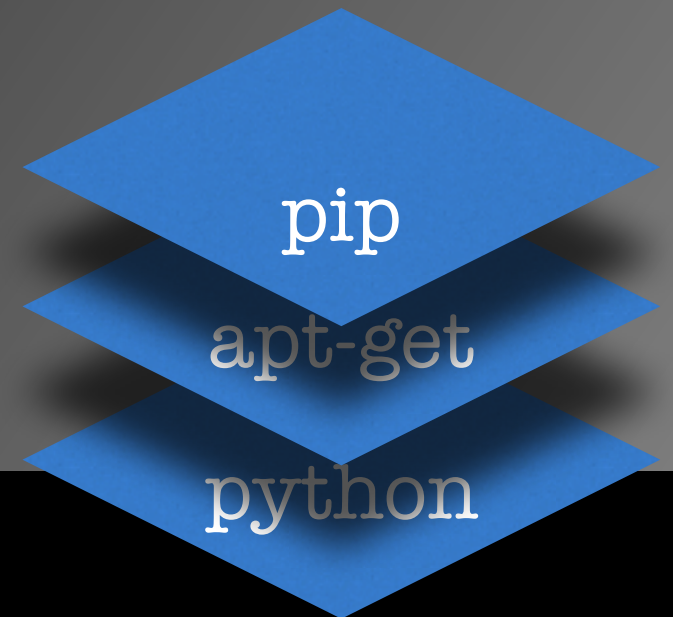


## Docker Desktop

- for Linux
- for macOS
- for Windows



# Dockerfile



```
# Use official Python image from hub.docker.com
# If you have M1 Mac, use
# FROM: arm64v8/python
FROM ${DOCKER_ARCH}python

# NOTE: To fully mimic production, we can set tags to match specific versions
#       of things like Python.
#       Also be aware that trying to use python:3-alpine will fail due to the
#       need for additional packages for LDAP/etc. Hence using stock image.

# Install needed modules so LDAP hits in pip modules will install properly
RUN apt-get update
RUN apt-get -y install libsasl2-dev python-dev libldap2-dev libssl-dev

RUN cd /

# Mount host directory where docker-compose.yml is located, which should be
# root of MyApp project source code so it appears as /MyApp in the container.
# This exposes things like the requirements.txt file so we can install all the
# relevant Python modules for this project
RUN --mount=type=bind,source=.,target=/MyApp pip install -r /MyApp requirements.txt
RUN pip install gunicorn

RUN cd /MyApp

CMD [ "echo", "MyApp is running..." ]
```



## Docker Desktop

- for Linux
- for macOS
- for Windows



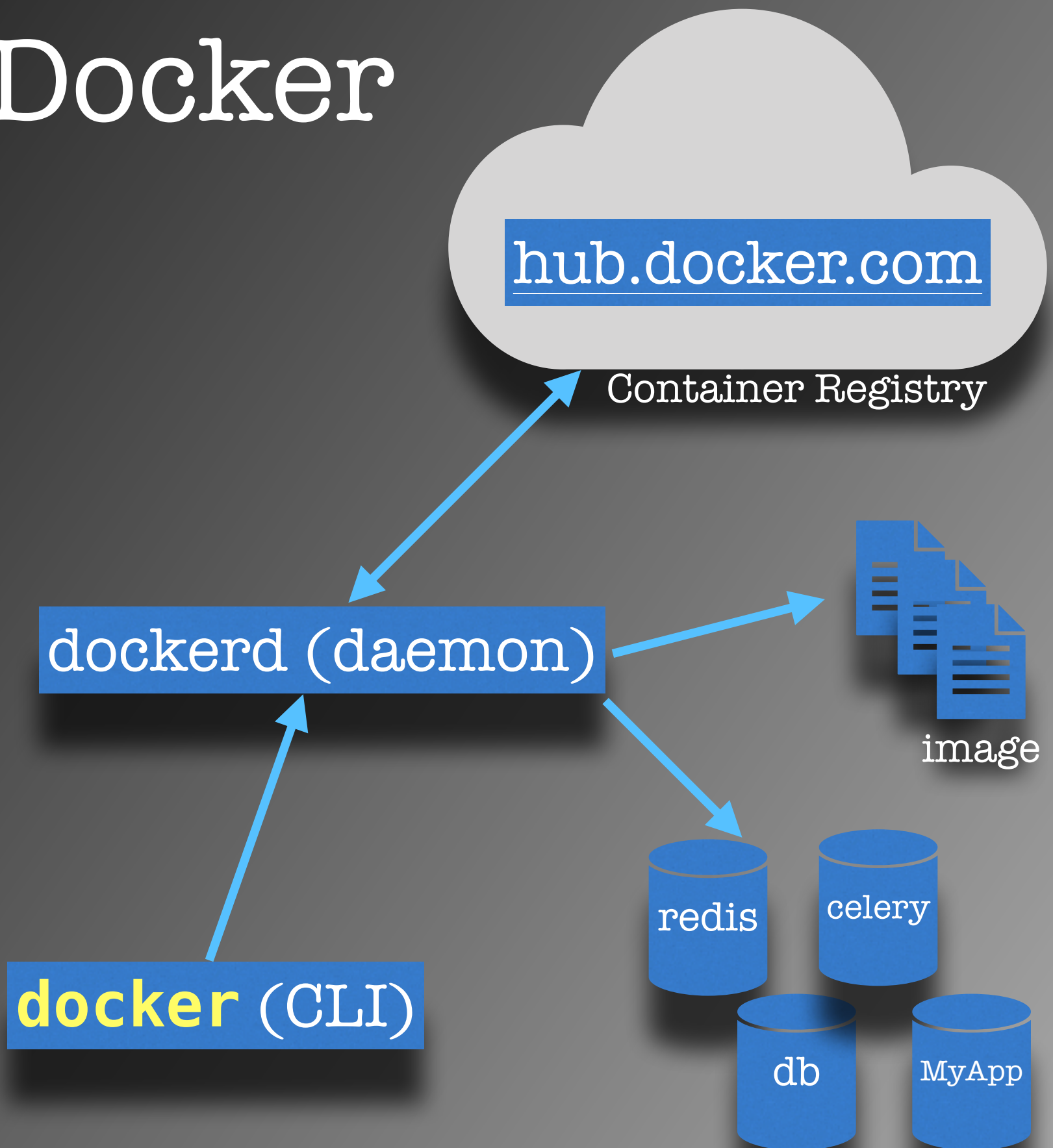


# Docker



## Docker Desktop

- for Linux
- for macOS
- for Windows



# Docker



dockerd (daemon)



## Docker Desktop

- for Linux
- for macOS
- for Windows



# Docker

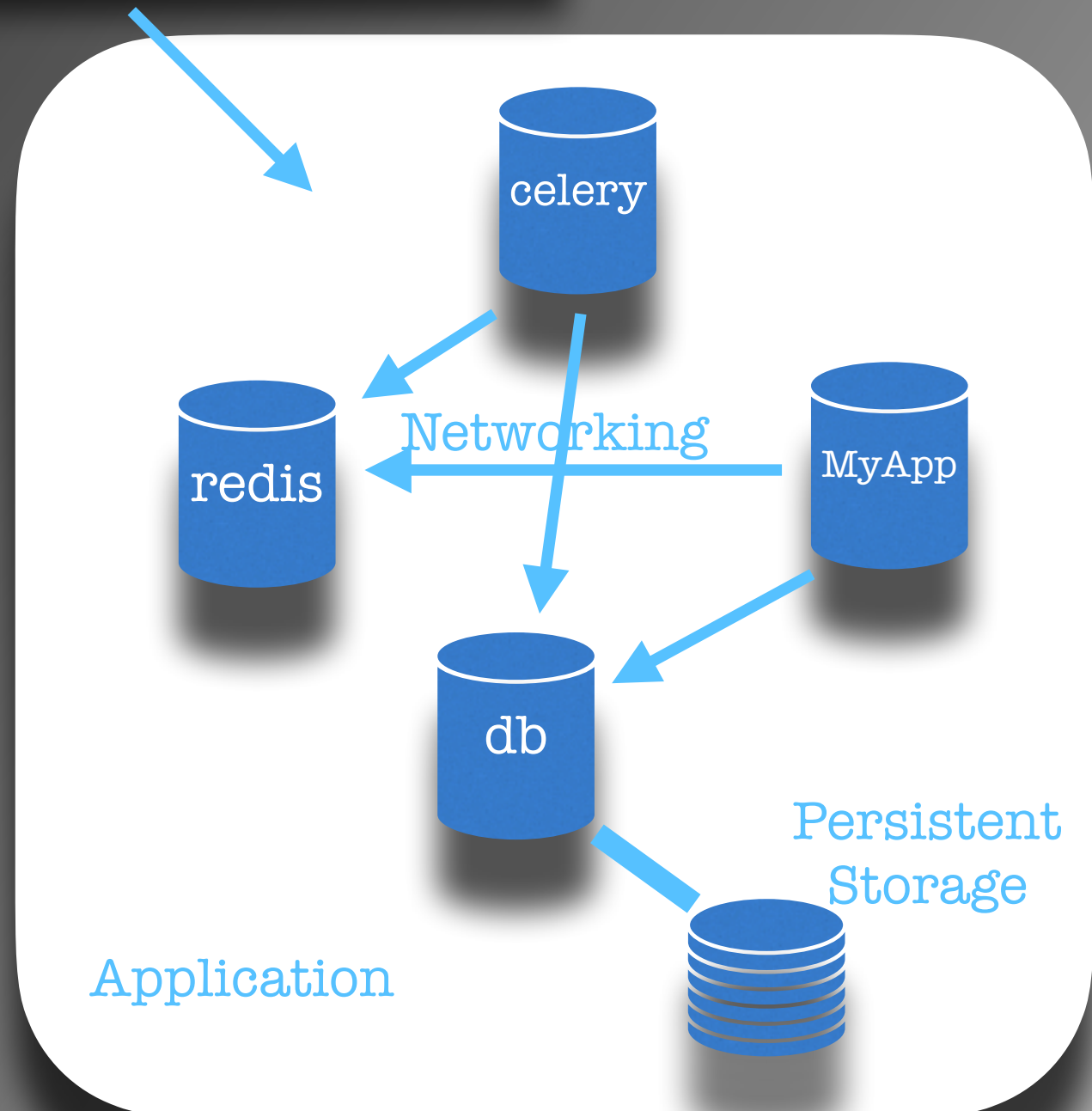


dockerd (daemon)



## Docker Desktop

- for Linux
- for macOS
- for Windows



# Docker Compose

docker-compose.yml



## Docker Desktop

- for Linux
- for macOS
- for Windows

```
services:
  redis:
    image: "${DOCKER_ARCH}redis:7-alpine"
    ports:
      - 6379:6379
    restart: unless-stopped
  db:
    image: "${DOCKER_ARCH}postgres:latest"
    ports:
      - 5432:5432
    volumes:
      - db-data:/var/lib/postgresql/data
    env_file: .env
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres"]
      interval: 5s
      timeout: 5s
      retries: 5
    restart: unless-stopped
  celery:
    build:
      context: .
      dockerfile: Dockerfile.Celery
    tags:
      - "celery:latest"
    ports:
      - 5672:5672
    volumes:
      - ${PWD}:/MyApp
    working_dir: /MyApp
    env_file: .env
    restart: unless-stopped
    command: [ 'celery', '--app', 'project', 'worker', '-B' ]
    depends_on:
      db:
        condition: service_healthy
      redis:
        condition: service_started
```





# Docker Limitations



What are Docker's limitations?

- Dockerd\* daemon runs as **root** by default
- Docker Engine is open source (Apache License v2). Docker Desktop is NOT. Nor is it free for all use.

"Commercial use of Docker Desktop at a company of more than 250 employees OR more than \$10 million in annual revenue requires a paid subscription (Pro, Team, or Business)."

– <https://www.docker.com/pricing/>

"4.2 Specific License Limitations – Docker Desktop.

(a) The Docker Desktop component of the Service at the level of the Personal Offering (as described on the Pricing Page) is further restricted to: (i) your "Personal Use", (ii) your "Educational Use", (iii) your use for a non-commercial open source project, and (iv) your use in a "Small Business Environment".

(b) For purposes of this Section 4.2: (i) "Personal Use" is the use by an individual developer for personal use to develop free or paid applications, (ii) "Educational Use" is the use by members of an educational organization in a classroom learning environment for academic or research purposes or contribution to an open source project..."

– <https://www.docker.com/legal/docker-subscription-service-agreement/>

## Docker Desktop

- for Linux
- for macOS
- for Windows



# Docker Alternatives



What are the alternatives?

- Colima - container runtimes on macOS (and Linux) [CLI]
- Red Hat OpenShift Local (formerly Red Hat CodeReady Containers)
- Podman
- Podman Desktop

## **Docker Desktop**

- for Linux
- for macOS
- for Windows

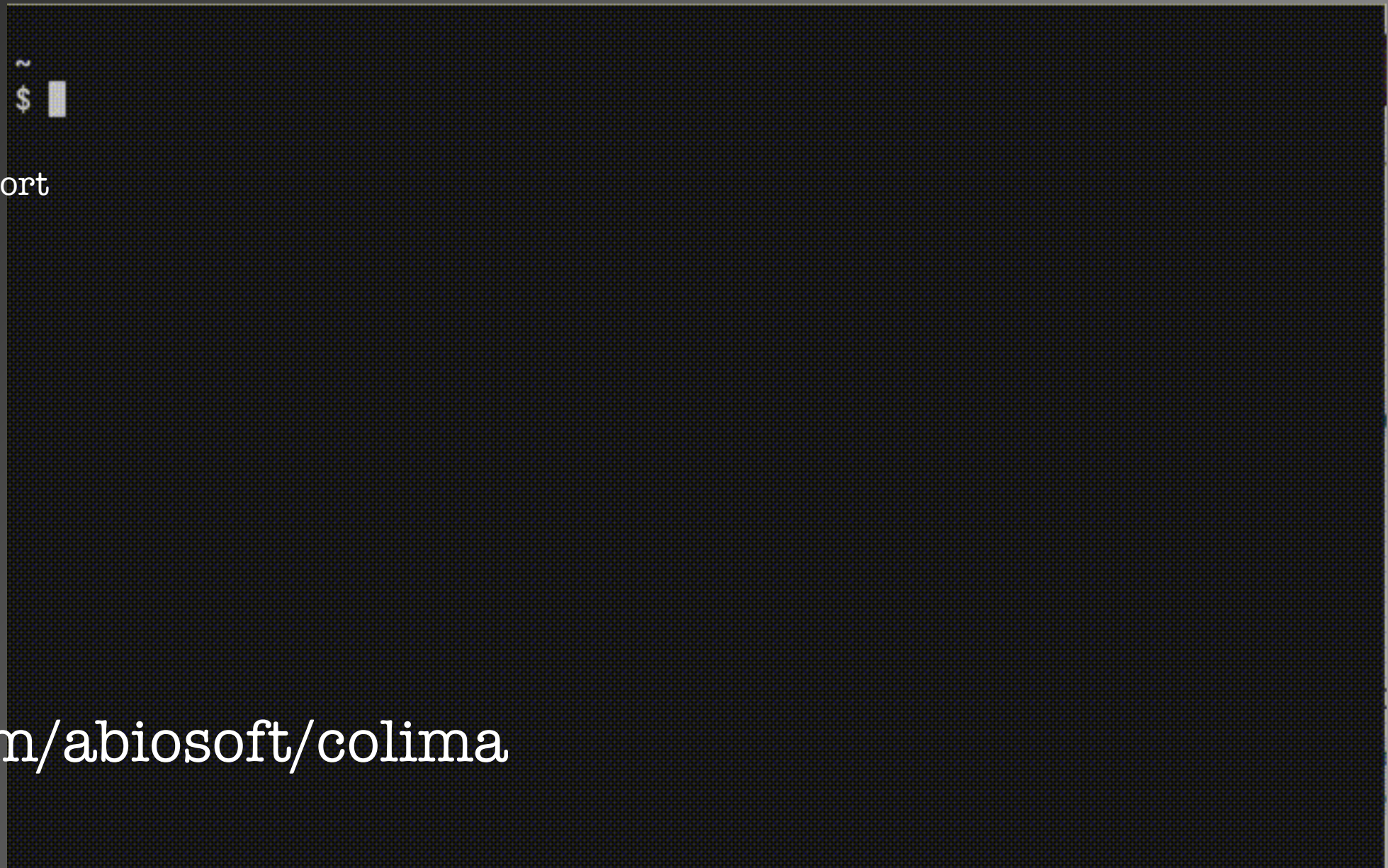




# Colima - container runtimes on macOS (and Linux) with minimal setup.

## Features

- Intel and M1 Macs support
- Simple CLI interface
- Docker and Containerd support
- Port Forwarding
- Volume mounts
- Kubernetes

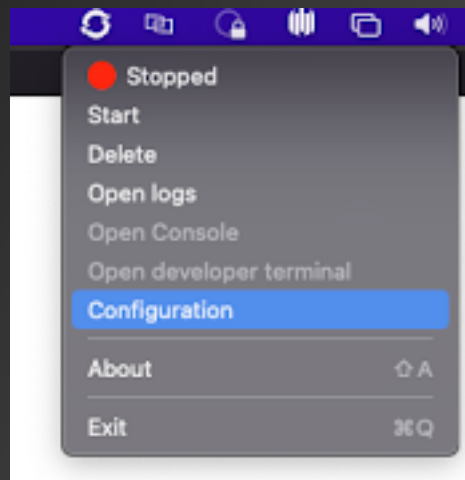


<https://github.com/abiosoft/colima>





# Red Hat OpenShift Local



## Hardware Requirements:

Red Hat OpenShift Local is supported on AMD64, Intel 64, and Apple M1 processor architectures.

For OpenShift Container Platform:

- 4 physical CPU cores
- 9 GB of free memory
- 35 GB of storage space

For the Podman container runtime:

- 2 physical CPU cores
- 2 GB of free memory
- 35 GB of storage space

## OS Requirements:

- Windows 10 Fall Creators Update (version 1709) or later
- macOS 11 Big Sur or later
- only on the latest two Red Hat Enterprise Linux/CentOS 8 and 9 minor releases and on the latest two stable Fedora releases

From [https://access.redhat.com/documentation/en-us/red\\_hat\\_openshift\\_local/2.25/html/getting\\_started\\_guide/installing](https://access.redhat.com/documentation/en-us/red_hat_openshift_local/2.25/html/getting_started_guide/installing)

<https://developers.redhat.com/products/openshift-local/overview>





# podman

"Podman is a daemonless container engine for developing, managing, and running OCI Containers on your Linux System. Containers can either be run as root or in rootless mode."

<https://podman.io/>



# Red Hat





# Podman Desktop

Podman Desktop interface showing a list of containers. The sidebar includes Dashboard, Containers (7), Images (15), Pods (1), Volumes (4), and Extensions. The main area displays a table of containers with columns for STATUS, NAME, STARTED, and ACTIONS.

STATUS	NAME	STARTED	ACTIONS
Running	nginx-pod (pod) 2 containers		
Running	nginx-pod-nginx-in-a-pod docker.io/library/nginx:1.14.2 PORT: 8088		
Running	99aaa69ebf98-infra localhost/podman-pause:4.3.0-1666340195 PORT: 8088		
Exited	quarkus-container quay.io/quarkus/ubi-quarkus-native-s2i:19.3.1-java11 PORT: 8081		
Running	redis quay.io/centos7/redis-5-centos7:latest PORT: 6379		
Running	redis-stack docker.io/redis/redis-stack:latest		

Podman Desktop interface showing a deployment configuration window titled "Deploy generated pod to Kubernetes". The window displays a generated pod manifest and configuration options.

```
Generated pod to deploy to Kubernetes:
1 # Save the output of this file and use kubectl create -f to import
2 # it into Kubernetes.
3 #
4 # Created with podman-4.3.0
5 apiVersion: v1
6 kind: Pod
7 metadata:
8   annotations:
9     io.kubernetes.cri-o.ContainerType/python-app-podified: container
10    io.kubernetes.cri-o.ContainerType/redis-podified: container
11    io.kubernetes.cri-o.SandboxID/python-app-podified: c2ab52854fb2
12    io.kubernetes.cri-o.SandboxID/redis-podified: c7ab52854fb2
```

Pod Name: my-pod

Use Kubernetes Services:  Replace .hostPort exposure on containers by Services. It is the recommended way to expose ports, as a cluster policy may prevent to use hostPort.

Kubernetes Context: crc-developer

Kubernetes Namespace: default

Deploy

Podman Desktop interface showing the Trivy vulnerability scanner. The sidebar includes Dashboard, Containers (10), Images (16), Pods (2), Volumes (6), Extensions, Trivy, Swagger Editor, OpenShift, and Settings.

quay.io/slemeur/python-app:latest

Only show vulnerabilities that have fixes

Severity	CVE	Package
HIGH	CVE-2022-2588	kernel-headers
HIGH	CVE-2022-3515	libksba
MEDIUM	CVE-2020-35527	sqlite
MEDIUM	CVE-2020-35527	sqlite-devel
MEDIUM	CVE-2020-35527	sqlite-libs

<https://podman-desktop.io/>



# Orchestration

"In system administration, orchestration is the automated configuration, coordination, and management of computer systems and software."  
— [https://en.wikipedia.org/wiki/Orchestration\\_\(computing\)](https://en.wikipedia.org/wiki/Orchestration_(computing))

## Examples of Container Orchestration Tools

- Kubernetes
- Docker Swarm
- Apache Mesos
- HashiCorp Nomad





Kubernetes (K8s)

Google







# Kubernetes (K8s)

## Cluster

### • Control Plane

- API
- Etcd
  - distributed key/value store

**kubectl**

“Cube cuddle”

### • Nodes (workers)

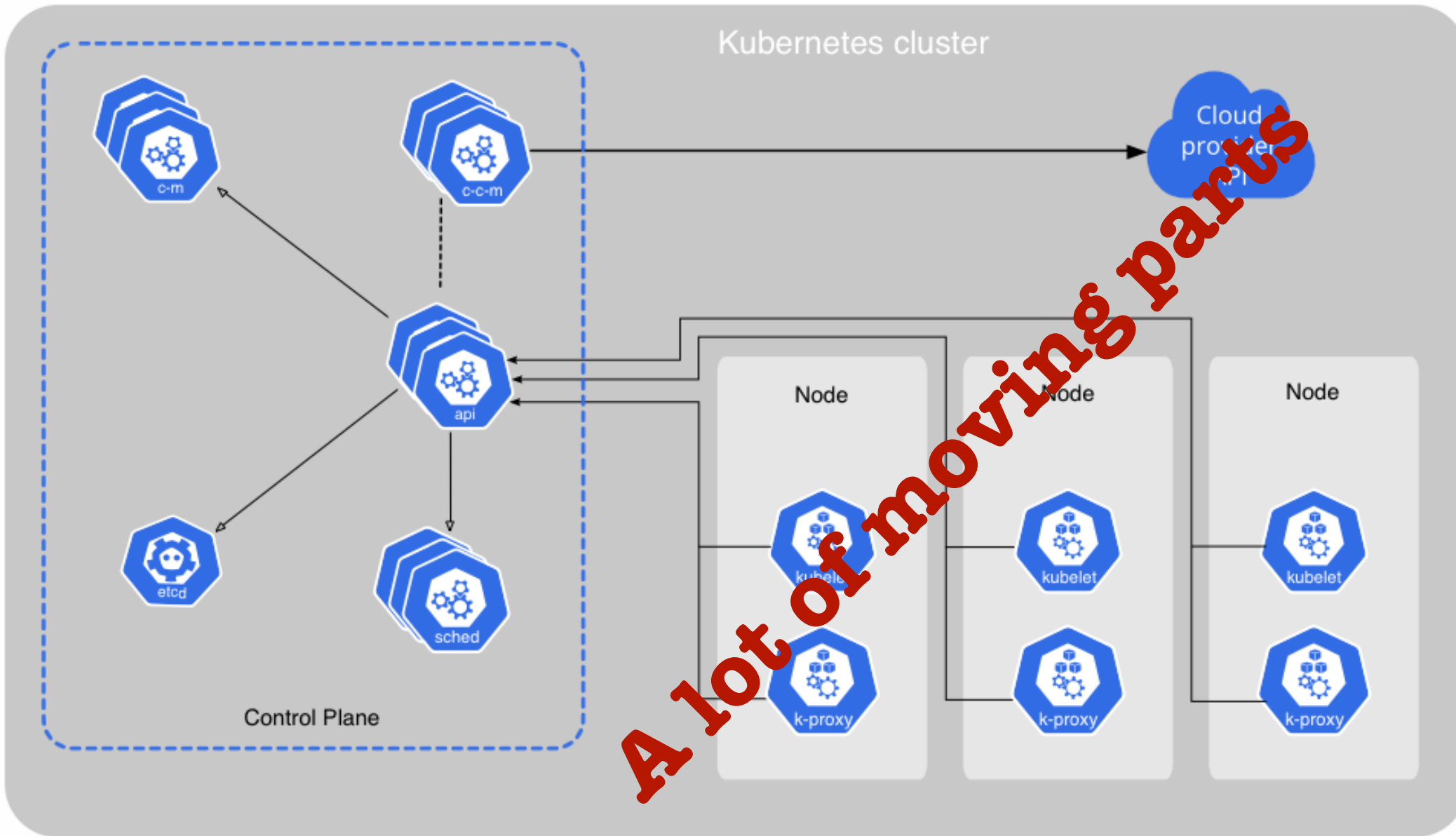
- Node1
  - Kubelet1
  - Pod1 ("Application 1")
    - Container1
    - Container2
  - Pod2 ("Application 2")
  - Pod3 ("Application 3")
- Node2
  - Kubelet2
  - Pod1
  - Pod2
- Node3
  - Kubelet3
  - Pod1
- ...

"Containers on steroids"





# Kubernetes (K8s)



- API server 
- Cloud controller manager (optional) 
- Controller manager 
- etcd (persistence store) 
- kubelet 
- kube-proxy 
- Scheduler 
- Control plane 
- Node 

"Containers on steroids"



# Kubernetes (K8s)





# Kubernetes (K8s)







# Kubernetes (K8s)

"The package manager for Kubernetes"

— <https://helm.sh/>



- Docker Compose file
- Dev tool
- Difference:
  - Mount a local volume

**YAML**

- Helm Chart
- Deployment tool
- Difference:
  - No local volumes

<https://kompose.io/>





# Kubernetes Implementations

Google

K8s

kubectl



minikube



CANONICAL



MicroK8s



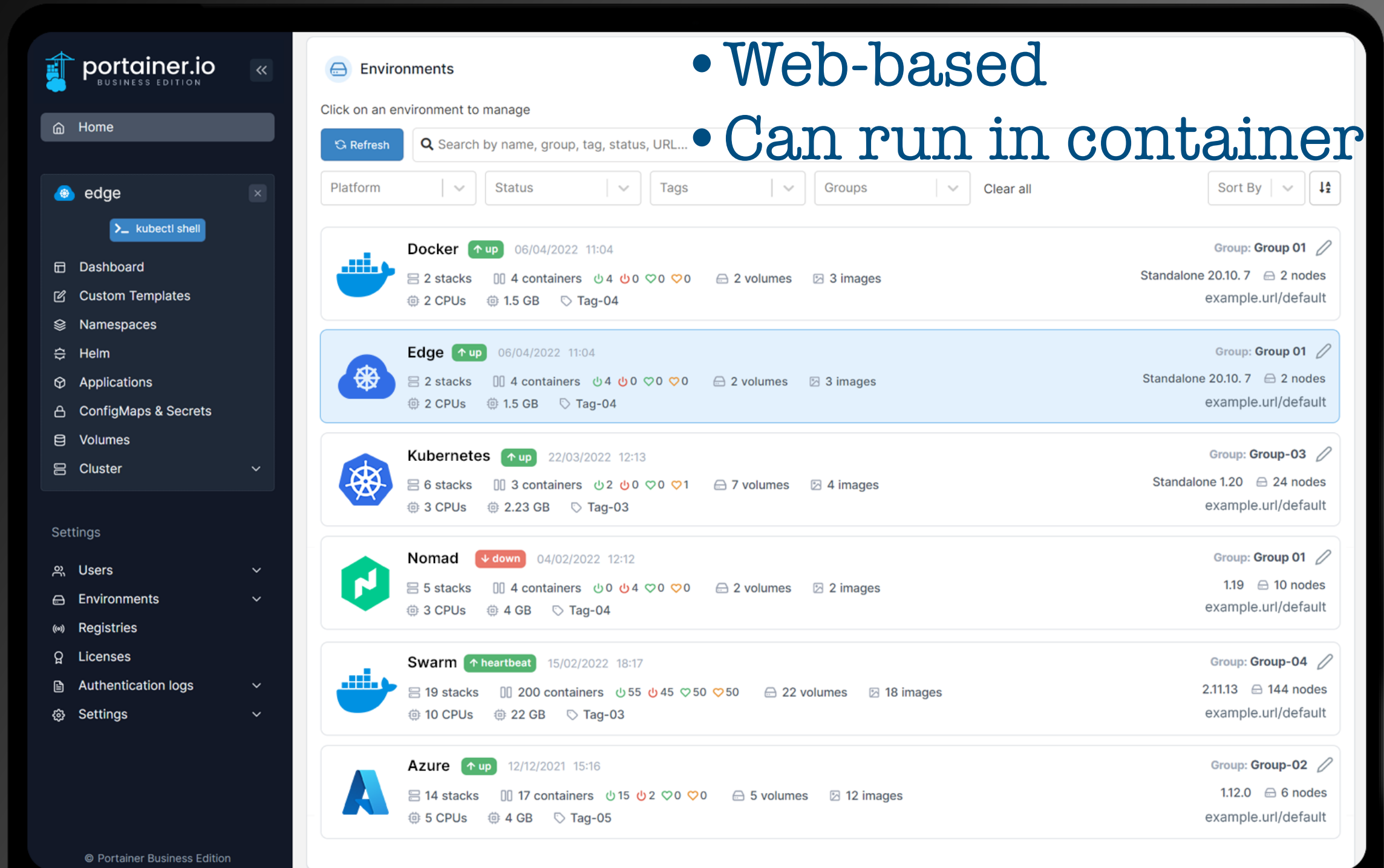
k0sctl



nerdctl



- Web-based
- Can run in container



The screenshot displays the Portainer.io web interface. On the left is a dark sidebar with navigation options: Home, edge (with a 'kubect! shell' button), Dashboard, Custom Templates, Namespaces, Helm, Applications, ConfigMaps & Secrets, Volumes, Cluster, Settings, Users, Environments, Registries, Licenses, Authentication logs, and Settings. The main content area is titled 'Environments' and contains a search bar and filter controls. Below this is a list of environments:

Environment	Status	Created	Time	Group	Version	Nodes	URL
Docker	↑ up	06/04/2022	11:04	Group 01	Standalone 20.10.7	2 nodes	example.url/default
Edge	↑ up	06/04/2022	11:04	Group 01	Standalone 20.10.7	2 nodes	example.url/default
Kubernetes	↑ up	22/03/2022	12:13	Group-03	Standalone 1.20	24 nodes	example.url/default
Nomad	↓ down	04/02/2022	12:12	Group 01	1.19	10 nodes	example.url/default
Swarm	↑ heartbeat	15/02/2022	18:17	Group-04	2.11.13	144 nodes	example.url/default
Azure	↑ up	12/12/2021	15:16	Group-02	1.12.0	6 nodes	example.url/default



Pod: calico-node-rwf4g

CPU Memory Network Filesystem

0.100

0

Created 27d, 5h, 55m ago (2019-08-21T18:50:15Z)

Namespace kube-system

Status Pushing

Node ip-192-168-81-7.eu-north-1.compute.internal

Pod IP 192.168.81.7

Priority Class \_

QoS Class Burstable

Labels controller-revision-hash: 5c9dddc74 k8s-app: calico-node pod-template-generation: 1

Annotations scheduler.alpha.kubernetes.io/critical-pod:

Conditions Initialized Ready ContainersReady PodScheduled

Controlled By DaemonSet calico-node

Tolerations 9 [Show](#)

Affinities 1 [Show](#)

Secrets calico-node-token-8dnkm

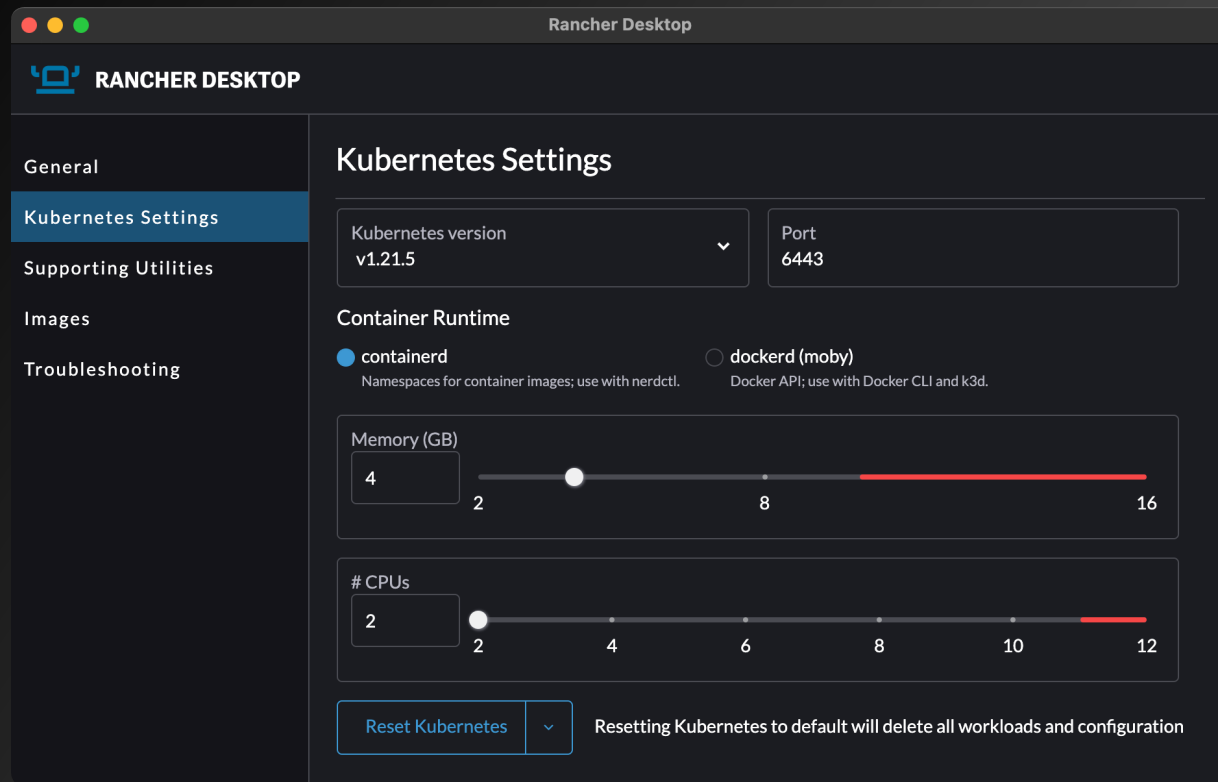
Name	Namespace	Containers
aws-node-hpz7f	kube-system	Running
aws-node-r9rgs	kube-system	Running
calico-node-rwf4g	kube-system	Running
calico-node-x8slr	kube-system	Running
calico-txbr-7k597kfp4-6ev5...	kube-system	Running
calico-txbr-horizontal-autos...	kube-system	Running
cert-manager-739c9c87-d...	kube-system	Running
coredn-5f74965bf-8ttz2	kube-system	Running
coredn-5f74965bf-ppq5j	kube-system	Running
dashboard-58bd48745-qpsft	kontena-lens	Running
helm-apr...	kontena-lens	Running
k8s-resource-applier-6d57d57...	kontena-lens	Running
kube-proxy-svno...	kube-system	Running
kube-proxy-v9x5g	kube-system	Running
kube-state-metrics-76ffbc879...	kontena-stats	Running
license-enforcer-5799c9c94c-...	kontena-lens	Running
mariadb-1568285718-master-0	jakolehm	Running
mariadb-1568285718-slave-0	jakolehm	Running
metrics-server-6bd546f4cd-n...	kube-system	Running

- Desktop app (Linux, macOS, Windows)
- Provides similar experience to Kubernetes Dashboard
- Can connect to any Kubernetes cluster via API using kubeconfig\* file

\* typically `~/.kube/config`







Rancher Desktop

RANCHER DESKTOP

General

Kubernetes Settings

Supporting Utilities

Images

Troubleshooting

### Kubernetes Settings

Kubernetes version v1.21.5 Port 6443

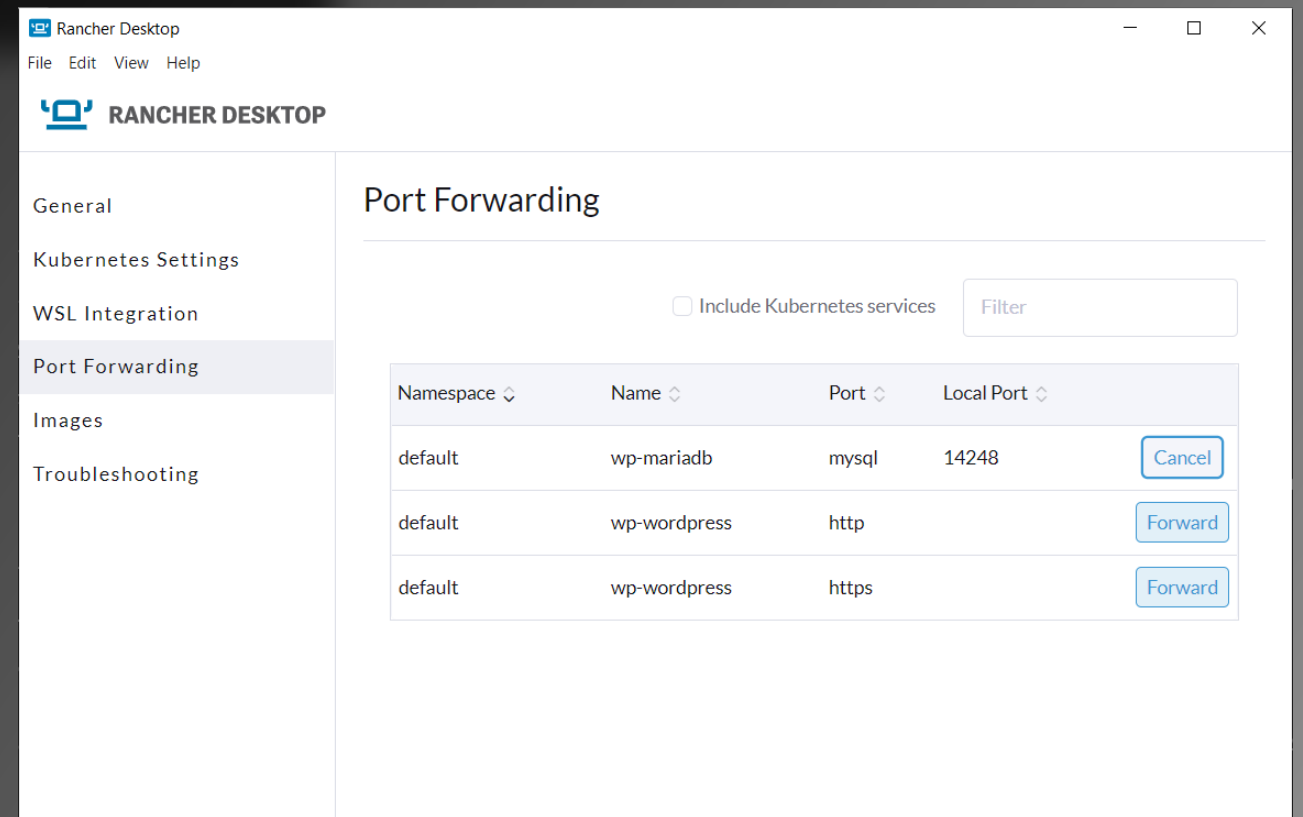
### Container Runtime

containerd Namespaces for container images; use with nerdctl.  
 dockerd (moby) Docker API; use with Docker CLI and k3d.

Memory (GB)  
4 2 8 16

# CPUs  
2 2 4 6 8 10 12

Reset Kubernetes Resetting Kubernetes to default will delete all workloads and configuration



Rancher Desktop

File Edit View Help

RANCHER DESKTOP

General

Kubernetes Settings

WSL Integration

Port Forwarding

Images

Troubleshooting

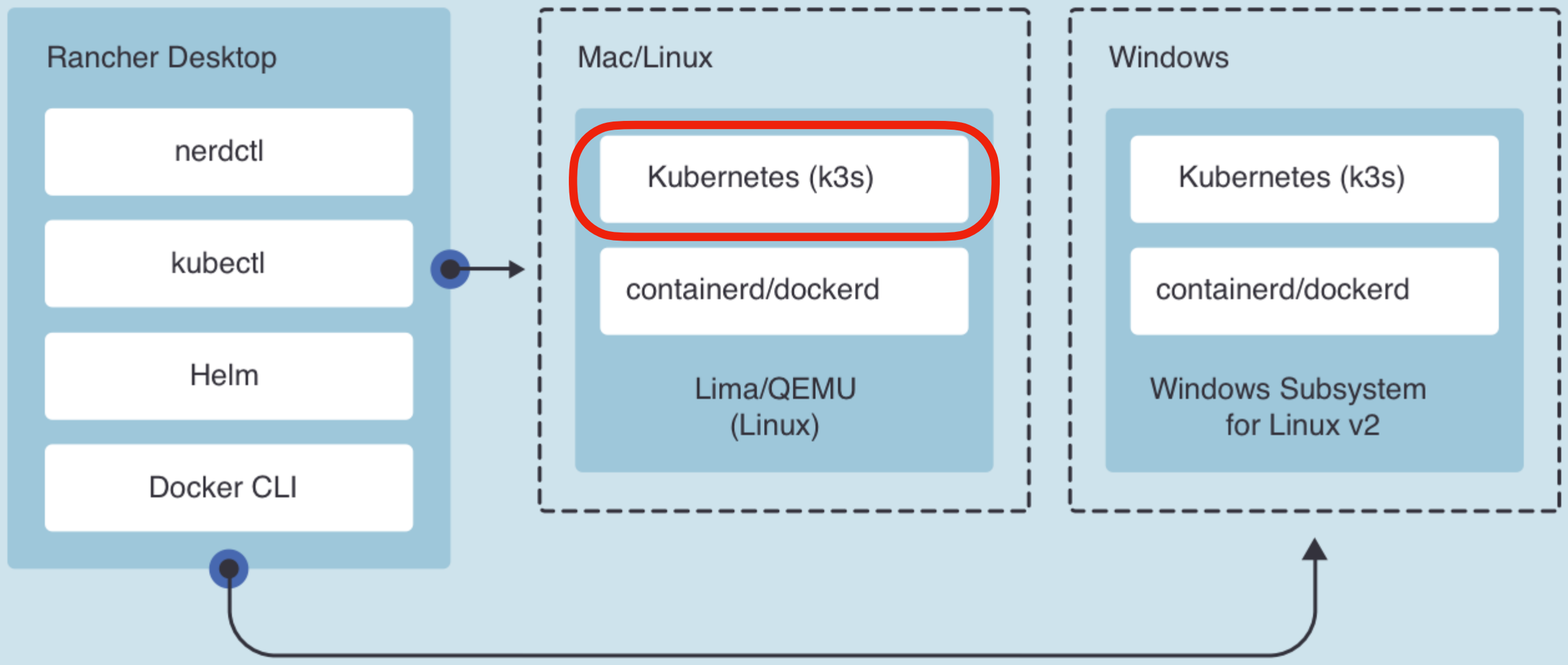
### Port Forwarding

Include Kubernetes services Filter

Namespace	Name	Port	Local Port	
default	wp-mariadb	mysql	14248	Cancel
default	wp-wordpress	http		Forward
default	wp-wordpress	https		Forward

## containerdctl

- a drop-in replacement for **docker** CLI
- You can do all your workloads using containerd



End Flashback  
Let's return to our  
show in progress...



# So How to Build Our Out-of-Band Server?





# OPTION #1: Build from Source

## ADVANTAGES:

- Simplest to understand
- Easiest to do initial build

## DISADVANTAGES:

- No repeatability\*; more difficult to maintain
- Long-term will take most time to maintain
- No rollback/recovery if a build goes bad
- More downtime



# OPTION #2: Docker-ize

## ADVANTAGES:

- More repeatable with Dockerfiles and docker-compose
- Allows for rollback/recovery
- Relatively easy to spin up/down manually

## DISADVANTAGES:

- Still doesn't automate code updates\*
- A dev tool / Security concerns (e.g., Docker daemon runs as root)
- Podman alternative has other issues

\*GitLab CI/CD may help with this



# OPTION #3: Single-node K3s Cluster

## ADVANTAGES:

- Offers full Kubernetes experience (with all benefits that brings) with less sysadmin overhead ("Tastes great; less filling")
- Very robust / scalable / repeatable
- Easiest to maintain long-term

## DISADVANTAGES:

- Most complex to initially understand / setup.



<https://k3s.io/>



**K3S** Docs GitHub

# Lightweight Kubernetes

The certified Kubernetes distribution built for IoT & Edge computing

**This won't take long...**

```
curl -sfL https://get.k3s.io | sh -  
# Check for Ready node, takes ~30 seconds  
sudo k3s kubectl get node
```

For detailed installation, [refer to the docs](#)

**Great For**

- Edge
- IoT
- CI
- ARM

<https://k3s.io/>





# K3s Installation

This is actually easy. Key preparations:

1. Setup FQDN for the server  
(IMPORTANT: K3s uses its own resolver)
2. Configure host firewall for proper communication (80,443,6443/TCP; etc.)
3. Use K3s Quick-Start Guide step to install:

```
sudo curl -sfL https://get.k3s.io | sh -
```



# K3s Installation

Additional steps:

4. Setup local users to have cluster access

5. Install Helm (<https://helm.sh/>)



6. Configure TLS certificates for Traefik



# K3s Maintenance

Step #1:

```
sudo curl -sL https://get.k3s.io | sh -
```

Step #2:

There is no step #2.

Everything else is like any other Linux server, so this server can be managed via Ansible, etc. like any other.



# Current Setup



=



Kubernetes (K8s)

Kubernetes is  
Kubernetes is  
Kubernetes

+





But what does this  
really mean?



What does  
Kubernetes even DO  
for us?

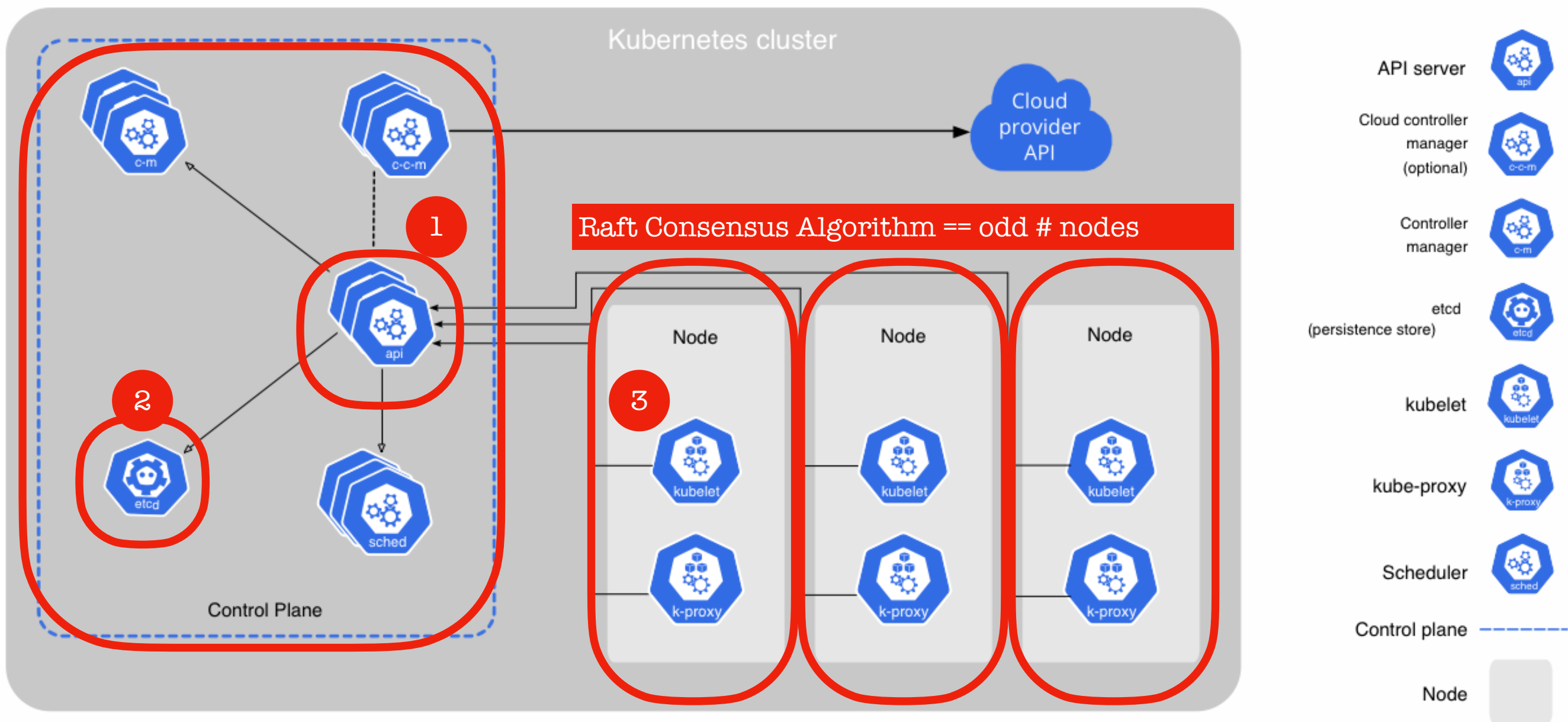


# Traditional App Deployment

- Build a server (e.g., a "LAMP" stack with Linux, Apache, MySQL, Perl/PHP/Python)
- Install application code
- Configure all the bits (“artisanal”)
- Ignore scaling issues, 10K problem, etc.



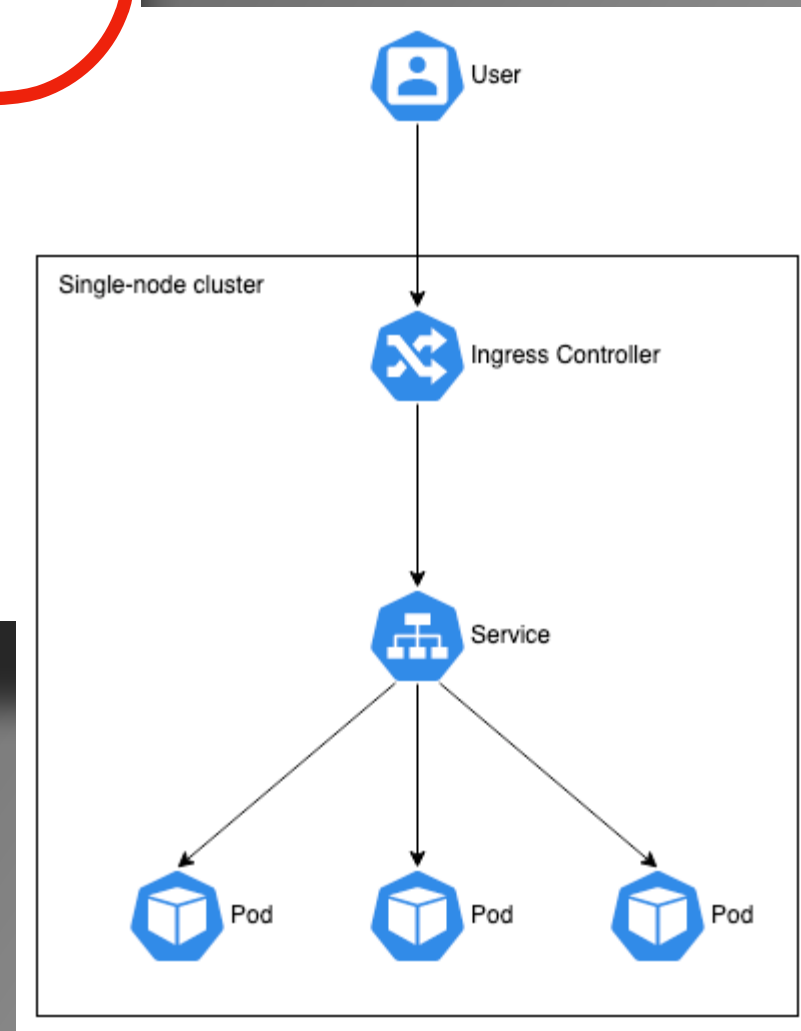
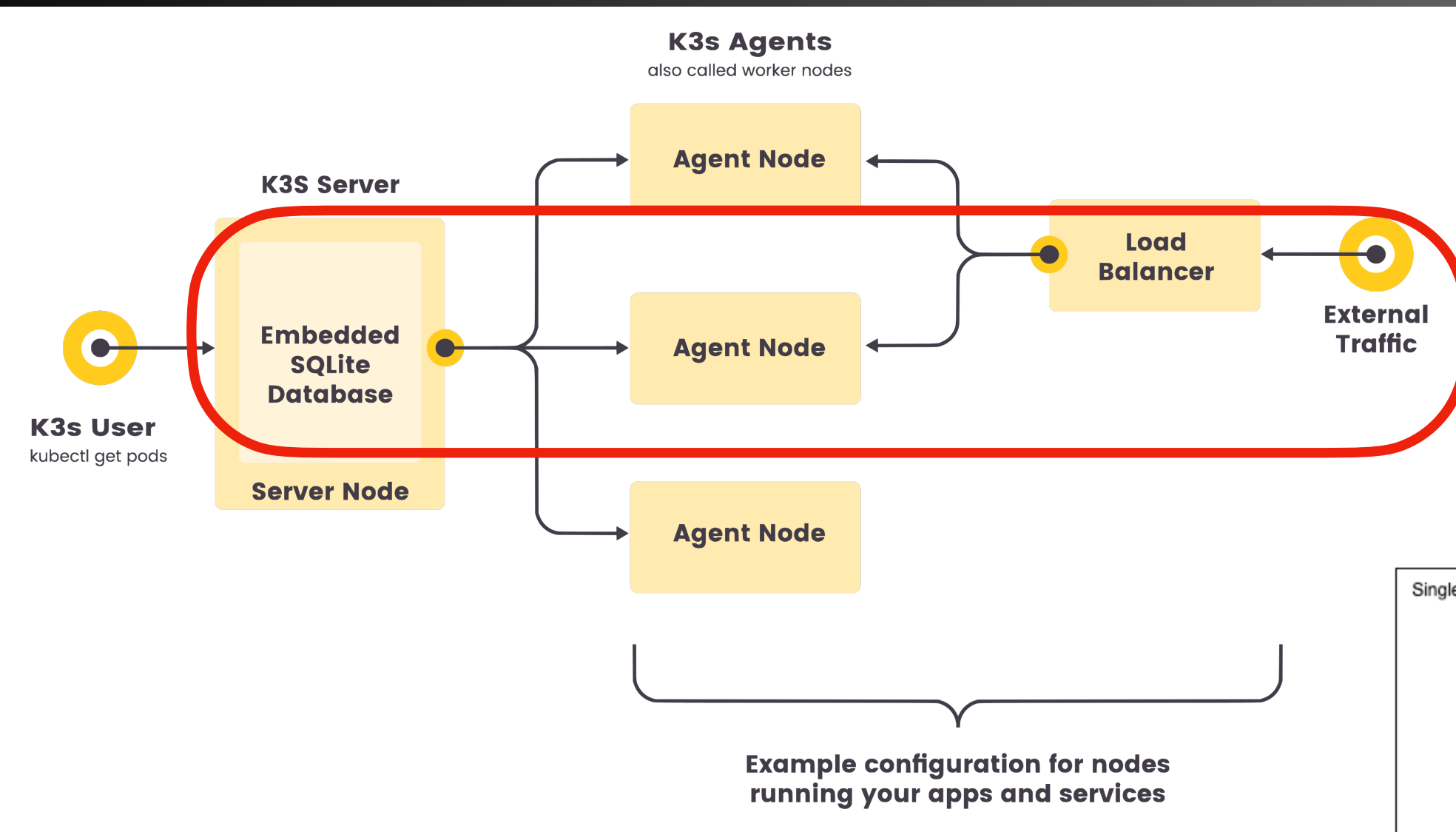
# Full-blown Kubernetes (K8s) Architecture



Normally min. 4 nodes



# K3s - Single Node Cluster



Minimum can be a single node for everything

6443/TCP

# K8s Default Setup

## Namespace

kube-system

53/TCP  
9153/TCP

kube-dns

80/TCP  
443/TCP

ingress cntlr

443/TCP

metrics-server

All pods assigned 10.43.x.x

default

443/TCP

kubernetes API

kube-public

kube-node-lease



6443/TCP

# K3s Default Setup

## Namespace

kube-system

53/TCP  
9153/TCP

CoreDNS

80/TCP  
443/TCP

Traefik

443/TCP

metrics-server

All K3s pods assigned 10.43.x.x

default

443/TCP

kubernetes API



**K3S**



6443/TCP

# K3s Default Setup

Namespace

kube-system

80/TCP  
443/TCP

Traefik

All K3s pods assigned 10.43.x.x

default

443/TCP

kubernetes API



**K3S**





6443/TCP

# K3s Default Setup

Namespace

kube-system

All K3s pods assigned 10.43.x.x

80/TCP  
443/TCP

Traefik

default

All application pods/containers assigned 10.42.x.x

pod (think "application")

container

Django/gunicorn



8000/TCP

container

PostgreSQL



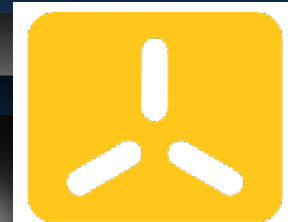
5432/TCP

container

Celery (or Redis,...)



5672/TCP



K3S



6443/TCP

# K3s Default Setup

## Namespace

kube-system

All K3s pods assigned 10.43.x.x

80/TCP  
443/TCP

Traefik

default

All application pods/containers assigned 10.42.x.x

pod (think "application")

container Django/gunicorn

80/TCP

8000/TCP

container PostgreSQL

5432/TCP

5432/TCP

container Celery (or Redis,...)

5672/TCP

5672/TCP



K3S



appFQDN:80

Internet

6443/TCP

# K3s Default Setup

Namespace

kube-system

80/TCP  
443/TCP

Traefik

All K3s pods assigned 10.43.x.x

default

All application pods/containers assigned 10.42.x.x

pod (think "application")

container Django/gunicorn

80/TCP

80/TCP

8000/TCP

container PostgreSQL

5432/TCP

5432/TCP

container Celery (or Redis,...)

5672/TCP

5672/TCP



K3S



# NOTES

- Deployments are configured via the Kubernetes API
- Services are configured via the Kubernetes API
- **Traefik** monitors changes in the cluster (such as service ports being exposed) via the Kubernetes API
- ...



KEY: EVERYTHING  
communicates via the  
Kubernetes API





# Kubectl

**kubectl** is the Command Line Interface (CLI) tool that connects to the Kubernetes API (6443/TCP). e.g.,

```
kubectl cluster-info
```

```
kubectl api-resources
```

```
kubectl get [all | nodes | pods |  
services | endpoints | namespaces/ns ]  
[-A]
```



# Kubectl

Red Hat OpenShift has a tool called simply “**oc**” (“OpenShift CLI”). This is simply OpenShift’s **kubectl**.



# Kubectl

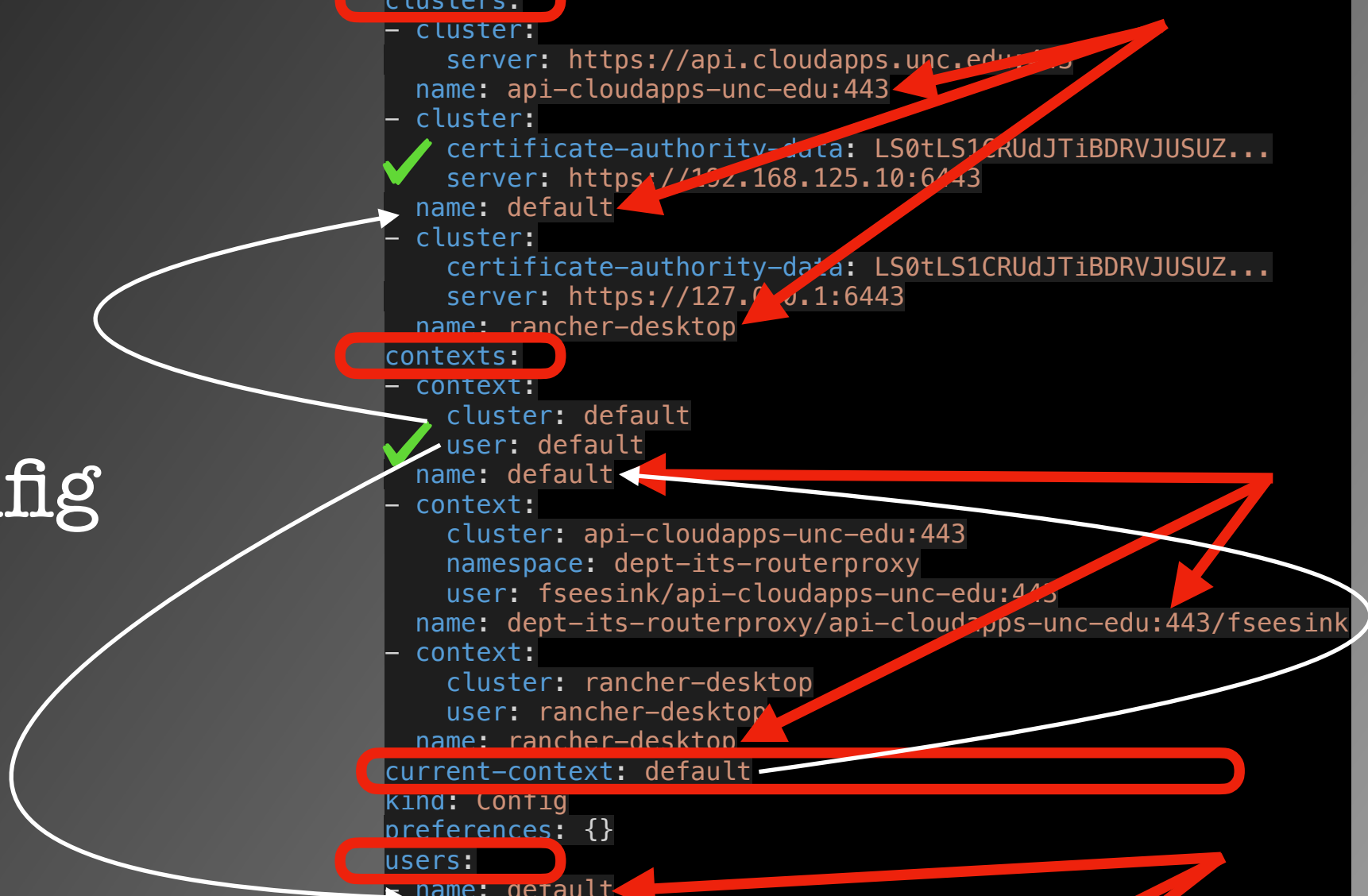
Now if you have multiple Kubernetes clusters you manage/interface with (e.g., OpenShift, K3s, etc.), how do **kubectl** and other apps know which API to connect to?



# Kubeconfig

~/.kube/config

```
apiVersion: v1
clusters:
- cluster:
  server: https://api.cloudapps.unc.edu:443
  name: api-cloudapps-unc-edu:443
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZ...
  server: https://192.168.125.10:6443
  name: default
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZ...
  server: https://127.0.0.1:6443
  name: rancher-desktop
contexts:
- context:
  cluster: default
  user: default
  name: default
- context:
  cluster: api-cloudapps-unc-edu:443
  namespace: dept-its-routerproxy
  user: fseesink/api-cloudapps-unc-edu:443
  name: dept-its-routerproxy/api-cloudapps-unc-edu:443/fseesink
- context:
  cluster: rancher-desktop
  user: rancher-desktop
  name: rancher-desktop
current-context: default
kind: Config
preferences: {}
users:
- name: default
  user:
  client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZ...
  client-key-data: LS0tLS1CRUdJTiBFQyBQUk1wQVRFIet...
- name: fseesink/api-cloudapps-unc-edu:443
  user:
  token: sha256~lR6K25T00CuswoJNtKEG03Fshy-9C028vcMZxmFwKP0
- name: rancher-desktop
  user:
  client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZ...
  client-key-data: LS0tLS1CRUdJTiBFQyBQUk1wQVRFIet...
```







Pod: calico-node-rwf4g

CPU Memory Network Filesystem

0.100

0

Created 27d, 5h, 55m ago (2019-08-21T18:50:15Z)

Namespace kube-system

Status Pushing

Node ip-192-168-81-7.eu-north-1.compute.internal

Pod IP 192.168.81.7

Priority Class \_

QoS Class Burstable

Labels controller-revision-hash: 5c9dddc74 k8s-app: calico-node

pod-template-generation: 1

Annotations scheduler.alpha.kubernetes.io/critical-pod:

Conditions Initialized Ready ContainersReady PodScheduled

Controlled By DaemonSet calico-node

Tolerations 9 [Show](#)

Affinities 1 [Show](#)

Secrets calico-node-token-8dnkm

Name	Namespace	Containers
aws-node-hpz7f	kube-system	Running
aws-node-r9rgs	kube-system	Running
calico-node-rwf4g	kube-system	Running
calico-node-x8slr	kube-system	Running
calico-txbr-7k597kfp4-6ev5...	kube-system	Running
calico-txbr-horizontal-autos...	kube-system	Running
cert-manager-739c9c87-d...	kube-system	Running
coredn-5f74965bf-8ttz2	kube-system	Running
coredn-5f74965bf-ppq5j	kube-system	Running
dashboard-58bd48745-qpsft	kontena-lens	Running
helm-apr...	kontena-lens	Running
k8s-resource-applier-6d57457...	kontena-lens	Running
kube-proxy-svno...	kube-system	Running
kube-proxy-v9x5g	kube-system	Running
kube-state-metrics-76ffbc879...	kontena-stats	Running
license-enforcer-5799c9c94c-...	kontena-lens	Running
mariadb-1568285718-master-0	jakolehms	Running
mariadb-1568285718-slave-0	jakolehms	Running
metrics-server-6bd546f4cd-n...	kube-system	Running

- Desktop app (Linux, macOS, Windows)
- Provides similar experience to Kubernetes Dashboard
- Can connect to any Kubernetes cluster via API using kubeconfig\* file

\* typically `~/.kube/config`



# Where do we get Kubeconfig?

On Server: `/etc/rancher/k3s/k3s.yaml`

```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUS3Z1T1I...
  server: https://127.0.0.1:6443
  name: default
contexts:
- context:
  cluster: default
  user: default
  name: default
current-context: default
kind: Config
preferences: {}
users:
- name: default
  user:
    client-certificate-data: WWVhaCBJIGRvbid0IHRoaW5...
    client-key-data: WW91IHJlYWxseSB0aGluayBWQVRFIEt...
```

1

Copy this file to your local machine (~/.kube/config)

2

Replace IP with K3s server IP/FQDN



# So How are Kubernetes Clusters Managed?



# Application Manifests

```
kubectl apply -f <manifest>.yaml  
kubectl delete -f <manifest>.yaml
```

Namespace (optional)

Deployment

Service

Ingress

---

YAML



# Application Manifests

## Namespace (optional)

```
#####  
# CREATE A NAMESPACE FOR THIS APPLICATION (OPTIONAL)  
#  
# NOTE: If you DO this, be sure to uncomment both the lines below  
#       and relevant `namespace:` attributes in other sections.  
#####  
# ---  
# apiVersion: v1  
# kind: Namespace  
# metadata:  
#   name: sample-appspace
```





# Application Manifests

## Deployment

```
---
#####
# CONFIGURE DEPLOYMENT OF THIS APPLICATION
# This is where you define the containers which make up your app
# and specify what ports each container exposes
#####
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app           # Name of the deployment
  # namespace: sample-appspace # Name of the namespace (optional; see above)
  labels:
    app: sample-app         # Name of your application
spec:
  selector:
    matchLabels:
      app: sample-app       # Name of your application
  replicas: 1               # Number of replicas
  template:
    metadata:
      labels:
        app: sample-app     # Name of your application
    spec:
      containers:
        # Containers are the individual pieces of your application that you want
        # to run. Sample uses NGINX container for testing.
        - name: nginx        # Name of the container
          image: nginx:latest # The image you want to run
          ports:
            # Ports are the ports that your application uses.
            - containerPort: 80 # The port that your application uses
```



# Application Manifests

## Service

```
---
#####
# CONFIGURE SERVICE
#
# This is where you expose your container ports to the rest of the cluster
#####
apiVersion: v1
kind: Service
metadata:
  name: sample-app
  # namespace: sample-appspace
spec:
  selector:
    app: sample-app
  # ---
  type: ClusterIP
  # ClusterIP means this service can be accessed by any pod in the cluster
  ports:
    - name: http
      port: 80
```



# Application Manifests

## Ingress

```
---
#####
# CONFIGURE INGRESS
#
# This is where you configure the external ingress route to your
# application.
#
# NOTE: Traefik should detect this automatically and stitch a path
#       so that inbound traffic destined for this application's
#       FQDN/path goes to this application.
#####
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: sample-app
  # namespace: sample-appspace
  # annotations:
  # ...
spec:
  rules:
  - host: "FQDN.of.sample-app"
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: sample-app
            port:
              number: 80
```



# Manifests

- You can have your namespace, deployment, service, ingress, etc. YAML files all separate, or...
- You can stitch all the separate YAML files of a pod together into a single file, with each segment separated by the usual 3 dashes. So you could use this approach to have one YAML file per application deployed.



# Managing K3s from the CLI





# Basic Kubectl Commands

```
$ kubectl cluster-info
```

```
Kubernetes control plane is running at https://127.0.0.1:6443  
CoreDNS is running at https://127.0.0.1:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy  
Metrics-server is running at https://127.0.0.1:6443/api/v1/namespaces/kube-system/services/https:metrics-server:https/proxy
```

```
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

```
$ kubectl api-resources
```

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
...				
tlsstores		traefik.containo.us/v1alpha1	true	TLSStore
traefikservices		traefik.containo.us/v1alpha1	true	TraefikService

```
$ kubectl get all
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.43.0.1	<none>	443/TCP	40d

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
serverbox01.net.unc.edu	Ready	control-plane,master	40d	v1.26.3+k3s1

```
$ kubectl get pods
```

```
No resources found in default namespace.
```

```
$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.43.0.1	<none>	443/TCP	40d

```
$ kubectl get endpoints
```

NAME	ENDPOINTS	AGE
kubernetes	192.168.125.10:6443	40d

```
$ kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	40d
kube-system	Active	40d
kube-public	Active	40d
kube-node-lease	Active	40d
gitlab-agent-k3s-agent	Active	40d
gitlab-runner	Active	40d

Why so little output?  
This is only the  
'default' namespace



# kubectl get all -A (Output Explained)

```
$ kubectl get all -A
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	pod/helm-install-traefik-crd-rk8vd	0/1	Completed	0	40d
kube-system	pod/helm-install-traefik-q9krp	0/1	Completed	1	40d
kube-system	pod/svclb-traefik-d9306a27-7kqk4	2/2	Running	44 (4h36m ago)	40d
kube-system	pod/coredns-7c444649cb-72kvt	1/1	Running	5 (4h36m ago)	11d
kube-system	pod/local-path-provisioner-79f67d76f8-v842l	1/1	Running	24 (4h36m ago)	40d
kube-system	pod/traefik-66c46d954f-5qhrq	1/1	Running	22 (4h36m ago)	40d
gitlab-agent-k3s-agent	pod/k3s-agent-gitlab-agent-c7d5bcfbd-lbcns	1/1	Running	4 (4h36m ago)	5d4h
kube-system	pod/metrics-server-5f9f776df5-ssm4j	1/1	Running	23 (4h36m ago)	40d
gitlab-runner	pod/gitlab-runner-6db656f9dc-ch9x7	1/1	Running	22 (4h36m ago)	40d

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
default	service/kubernetes	ClusterIP	10.43.0.1	<none>	443/TCP	40d
kube-system	service/kube-dns	ClusterIP	10.43.0.10	<none>	53/UDP,53/TCP,9153/TCP	40d
kube-system	service/metrics-server	ClusterIP	10.43.47.30	<none>	443/TCP	40d
kube-system	service/traefik	LoadBalancer	10.43.63.81	192.168.125.10	80:31967/TCP,443:30819/TCP	40d

NAMESPACE	NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
kube-system	daemonset.apps/svclb-traefik-d9306a27	1	1	1	1	1	<none>	40d

NAMESPACE	NAME	READY	UP-TO-DATE	AVAILABLE	AGE
kube-system	deployment.apps/traefik	1/1	1	1	40d
kube-system	deployment.apps/coredns	1/1	1	1	40d
kube-system	deployment.apps/local-path-provisioner	1/1	1	1	40d
gitlab-agent-k3s-agent	deployment.apps/k3s-agent-gitlab-agent	1/1	1	1	40d
kube-system	deployment.apps/metrics-server	1/1	1	1	40d
gitlab-runner	deployment.apps/gitlab-runner	1/1	1	1	40d

NAMESPACE	NAME	DESIRED	CURRENT	READY	AGE
kube-system	replicaset.apps/traefik-66c46d954f	1	1	1	40d
kube-system	replicaset.apps/coredns-7c444649cb	1	1	1	11d
gitlab-agent-k3s-agent	replicaset.apps/k3s-agent-gitlab-agent-76fc86cb9	0	0	0	40d
gitlab-agent-k3s-agent	replicaset.apps/k3s-agent-gitlab-agent-55bfb5b59c	0	0	0	5d4h
kube-system	replicaset.apps/local-path-provisioner-79f67d76f8	1	1	1	40d
gitlab-agent-k3s-agent	replicaset.apps/k3s-agent-gitlab-agent-c7d5bcfbd	1	1	1	5d4h
kube-system	replicaset.apps/metrics-server-5f9f776df5	1	1	1	40d
gitlab-runner	replicaset.apps/gitlab-runner-6db656f9dc	1	1	1	40d

NAMESPACE	NAME	COMPLETIONS	DURATION	AGE
kube-system	job.batch/helm-install-traefik-crd	1/1	13s	40d
kube-system	job.batch/helm-install-traefik	1/1	15s	40d



# Traefik



- Normally Traefik uses LetsEncrypt to add SSL certificates to apps. This, however, requires that the LetsEncrypt servers can reach into the Kubernetes cluster to verify things on the website.
- This will not work in some environments, notably ones for internal use only.



# Traefik Default SSL Cert.

```
---
apiVersion: traefik.containo.us/v1alpha1
kind: TLSStore
metadata:
  name: default
  namespace: kube-system
spec:
  defaultCertificate:
    secretName: default-certificate
```

```
---
apiVersion: v1
kind: Secret
metadata:
  name: default-certificate
  namespace: kube-system
type: kubernetes.io/tls
```

```
data:
  # To create the following 2 lines, take .pem/.key files and base64 them; e.g.,
  # cat FQDN.of.sample-app.pem | base64
  # cat FQDN.of.sample-app.key | base64
  tls.crt: LS0tLS1CRUdJT...
  tls.key: LS0tLS1CRUdJT...
```



# Cloud Native Computing Foundation (CNCF)

"CNCF is the open source, vendor-neutral hub of cloud native computing, hosting projects like Kubernetes and Prometheus to make cloud native universal and sustainable."

— <https://www.cncf.io/>





**App Definition and Development**

Database: KV, Viteess, Conoco, etc.

Streaming & Messaging: cloudevents, NATS, etc.

Application Definition & Image Build: HELM, Backstage, Buildpacks.io, KubeVela, KubeVirt, etc.

Continuous Integration & Delivery: argo, flux, keptn, OpenKruise, etc.

**Orchestration & Management**

Scheduling & Orchestration: kubernetes, Crossplane, etc.

Coordination & Service Discovery: CoreDNS, etcd, gRPC, etc.

Remote Procedure Call: gRPC, etc.

Service Proxy: envoy, contour, etc.

API Gateway: Emissary Ingress, etc.

Service Mesh: Istio, Linkerd, etc.

**Runtime**

Cloud Native Storage: Rook, Ceph, etc.

Container Runtime: cri-o, Lima, etc.

Cloud Native Network: cilium, CNi, etc.

**Provisioning**

Automation & Configuration: KubeEdge, etc.

Container Registry: Harbor, Dragonfly, etc.

Security & Compliance: Falco, etc.

Key Management: spiffe, SPIRE, etc.

**Special**

Kubernetes Certified Service Provider: acend, etc.

Kubernetes Training Partner: acend, etc.

Certified CNFs: acend, etc.

**Platform**

Certified Kubernetes - Distribution: AWS, etc.

Certified Kubernetes - Hosted: AWS, etc.

Certified Kubernetes - Installer: AWS, etc.

PaaS/Container Service: etc.

**Observability and Analysis**

Monitoring: Prometheus, etc.

Logging: fluentd, etc.

Tracing: etc.

**Chaos Engineering**

Chaos Mesh, Litmus, etc.

Continuous Optimization: etc.

**Serverless**

CNCF Serverless Landscape

**Members**

Member logos and information

**CD Foundation Landscape**

Continuous Delivery landscape

**Cloud Native Landscape**

Cloud Native Computing Foundation logo and QR code

**l.cncf.io**

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

# Thank You



[https://frank.seesink.com/presentations/  
Internet2TechEx-Fall2023/](https://frank.seesink.com/presentations/Internet2TechEx-Fall2023/)

Frank Seesink  
[frank@seesink.com](mailto:frank@seesink.com)  
[frank@unc.edu](mailto:frank@unc.edu)

