Containers and Orchestration Oh My!

Frank Seesink, UNC Chapel Hill

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

First, a message from our sponsor...

What I picture in my head...

What it ends up looking like...

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

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Story Time...

Current Setup

django

Red Hat
OpenShift

NETMIKO

SQLite

db

Nornir

gunicorn

Work environment

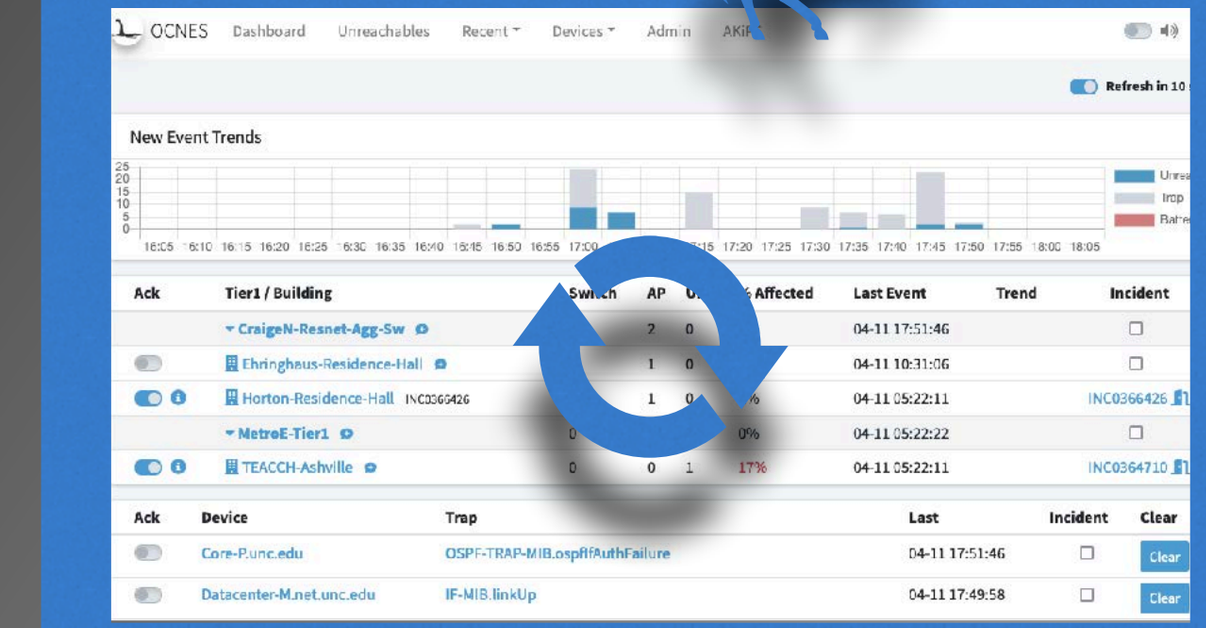
Current Setup

GitLab
on-prem

Webhook

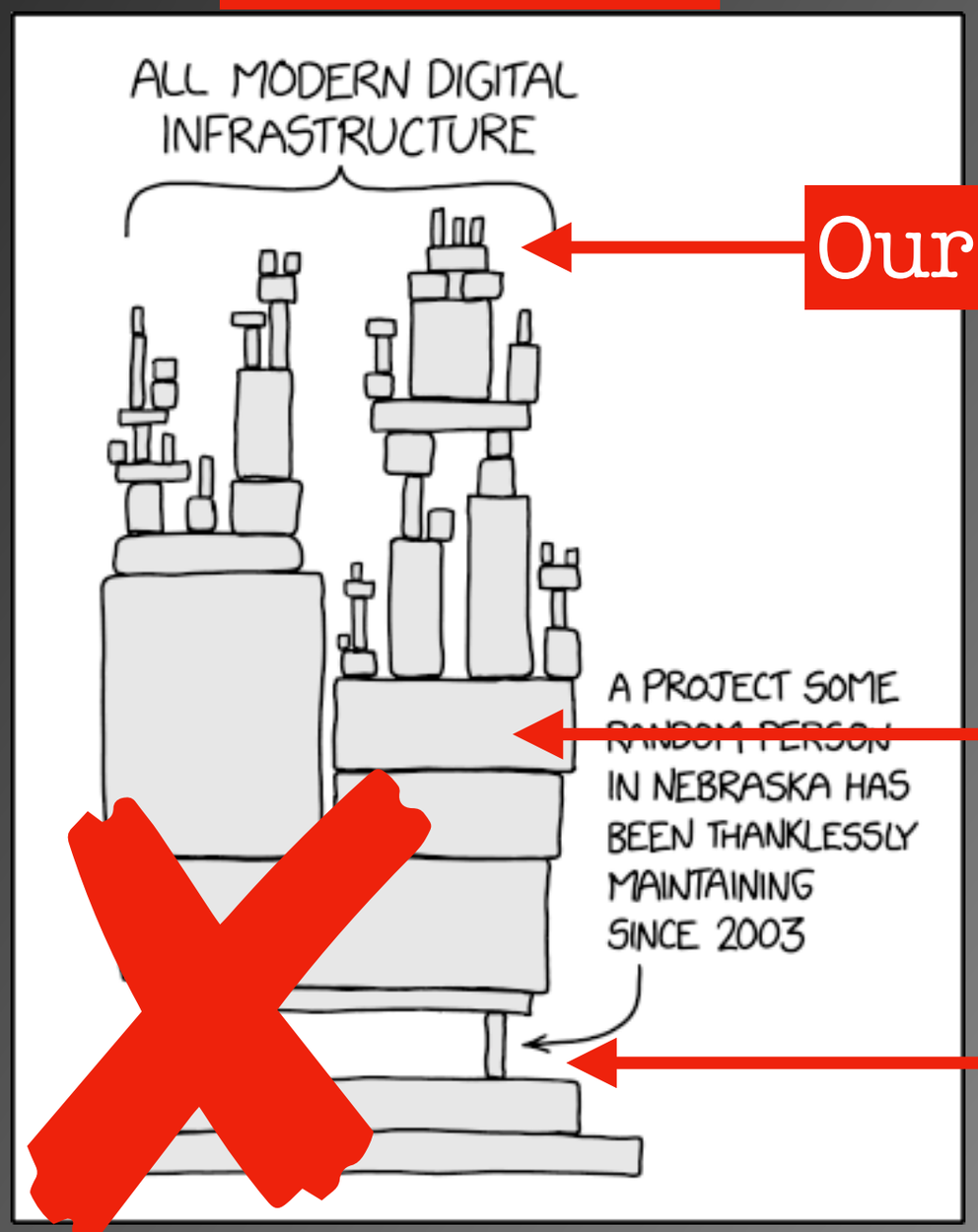
RED HAT® OPENSHIFT Container Platform

Repo



Current Setup

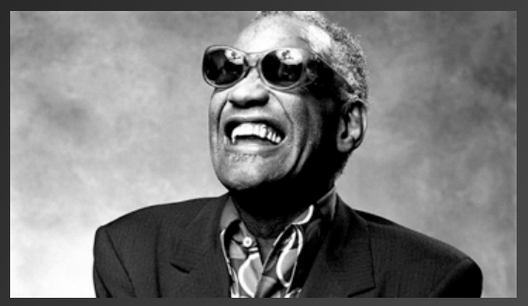
Datacenter



Our Monitoring Apps

RED HAT® OPENSHIFT Container Platform

vmware ESXi



Us

Potential issue

https://xkcd.com/2347

The Plan

Out-Of-Band Server

GitLab

Repo

- GitLab Runners
- CI/CD pipelines

?

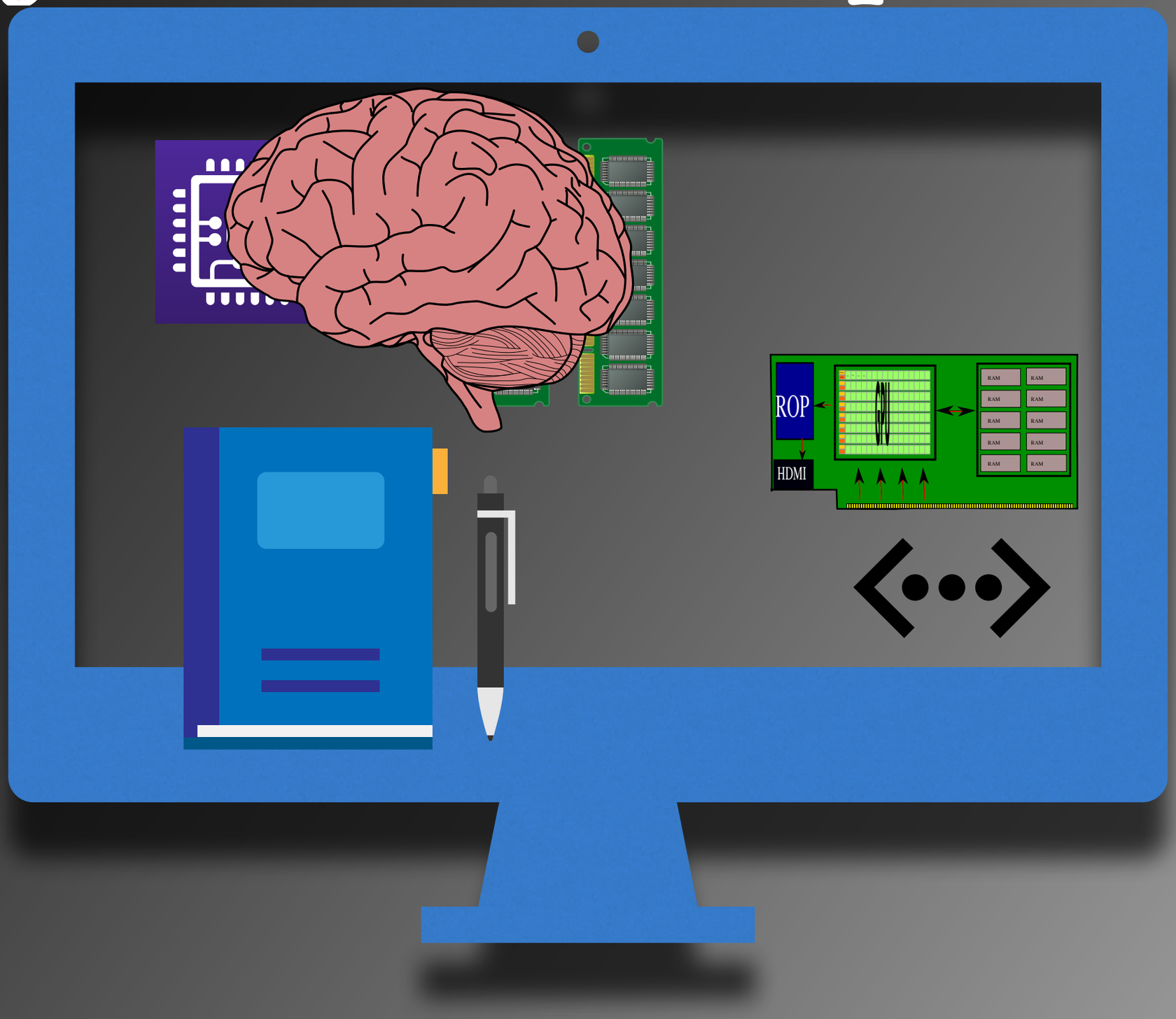
Server

- Webserver (NGINX)
- Django/Gunicorn
- PostgreSQL
- Celery
- Redis

That's the backstory.

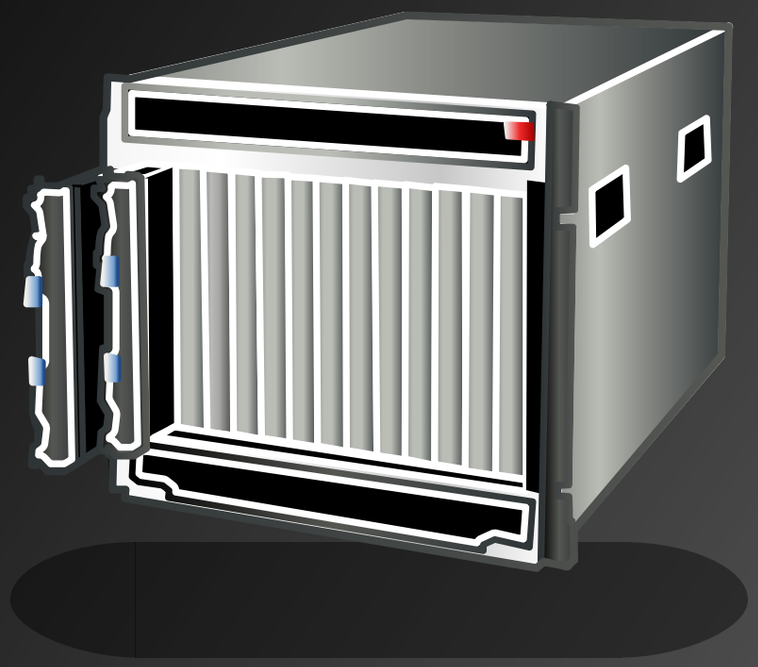
Now a flashback...

In the beginning...
you had a computer



That computer ran an OS...

Then came Virtual Machines (VMs)...



vmware®

https://www.vmware.com/

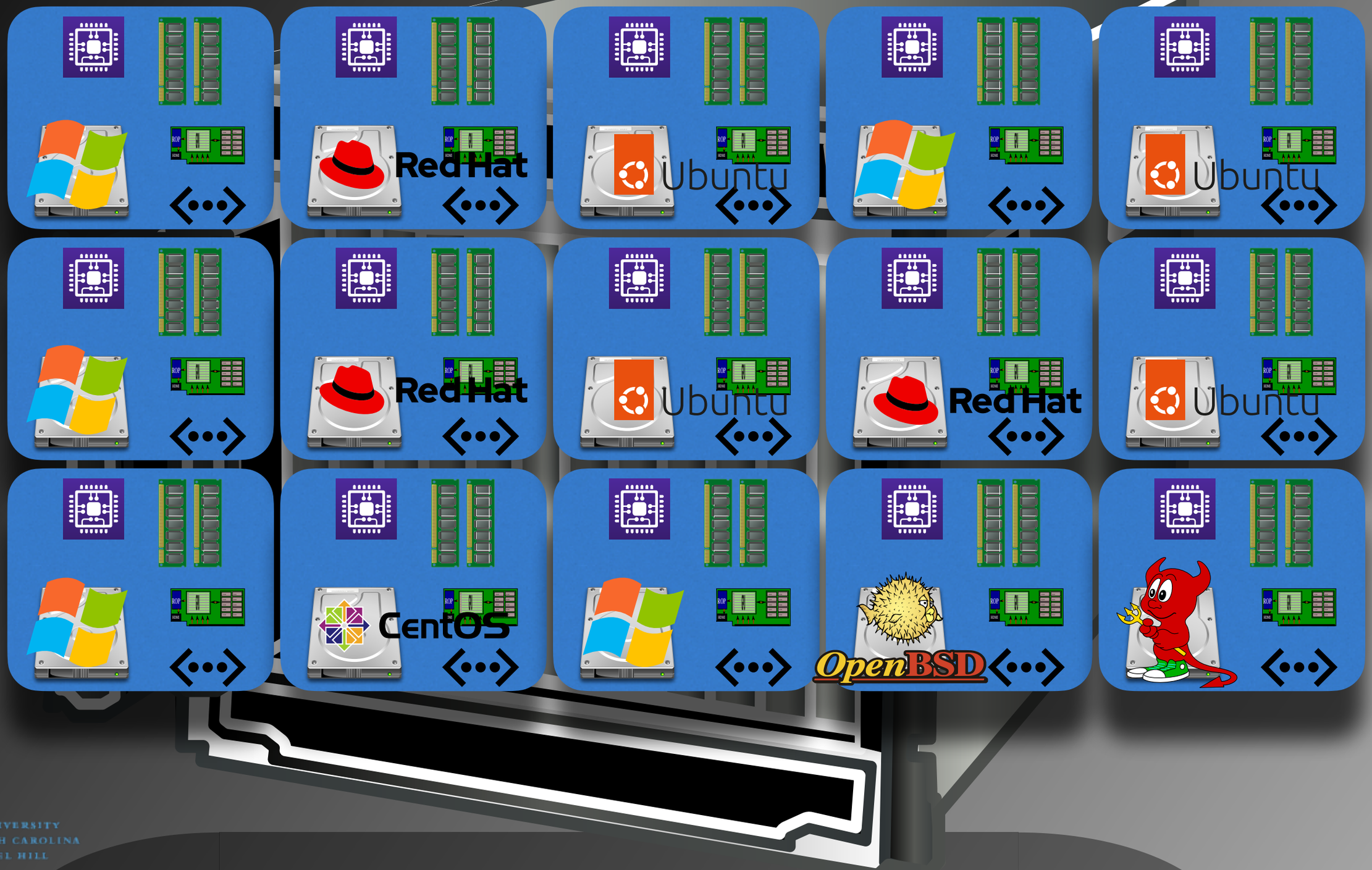
https://www.proxmox.com

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/

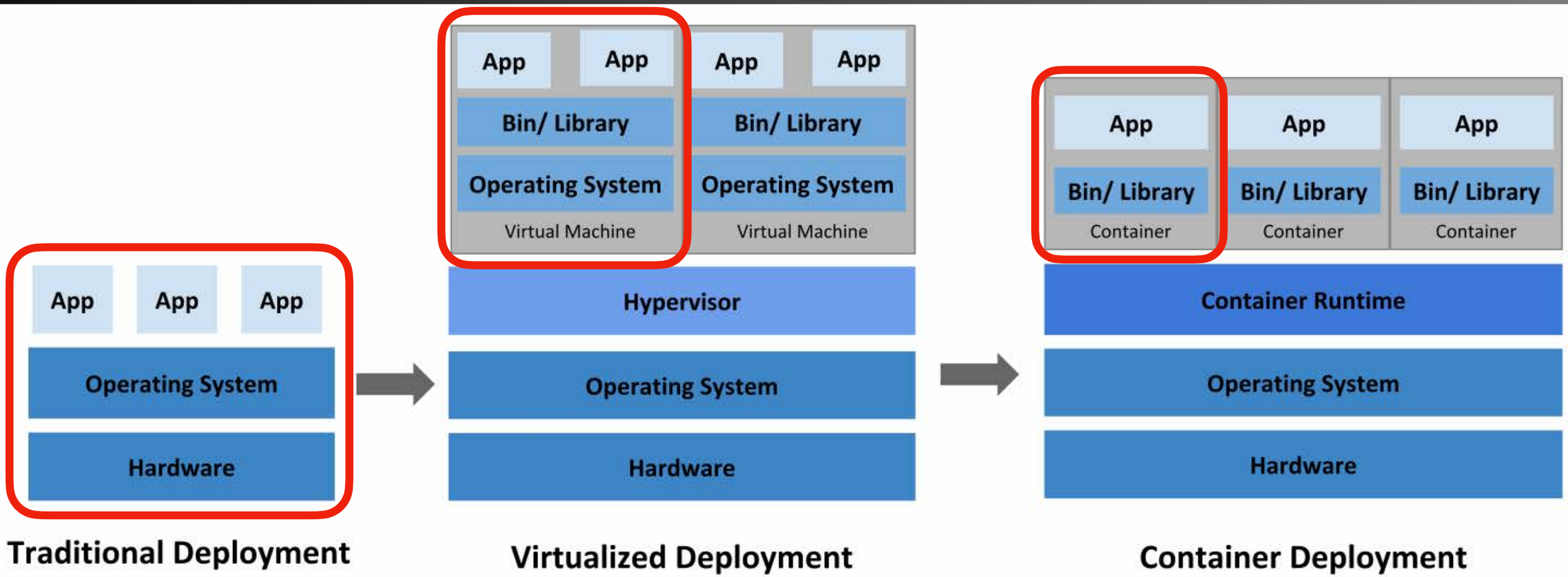
Container Terms

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



	Traditional Deployment	Virtualized Deployment	Container Deployment
Installation time	hours/days	hours/days	minutes
Startup time	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

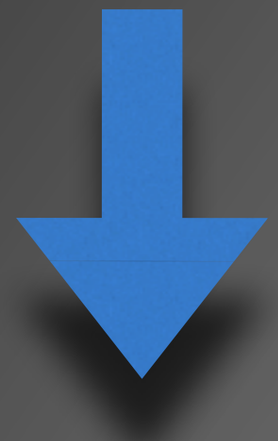
Container History

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

Docker Desktop

- for Linux
- for macOS
- for Windows

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

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

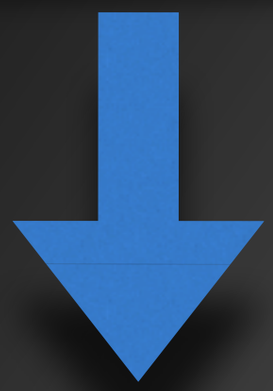
Docker Desktop

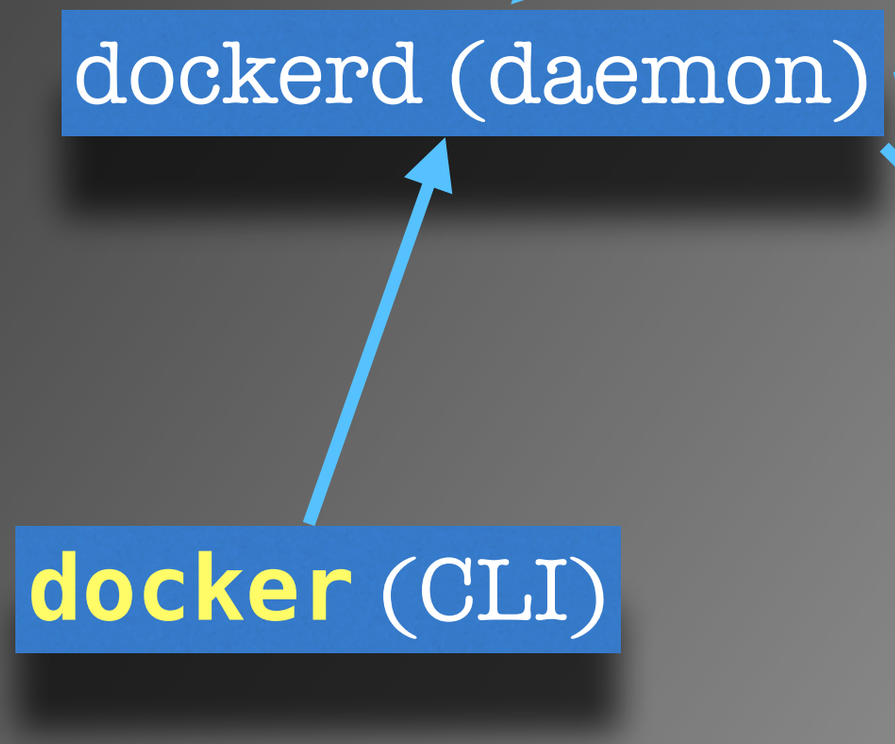
- for Linux
- for macOS
- for Windows

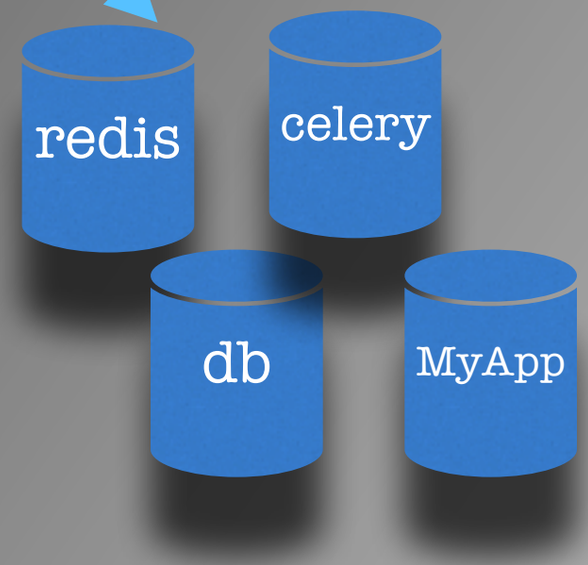
Docker (visual)

Docker Desktop

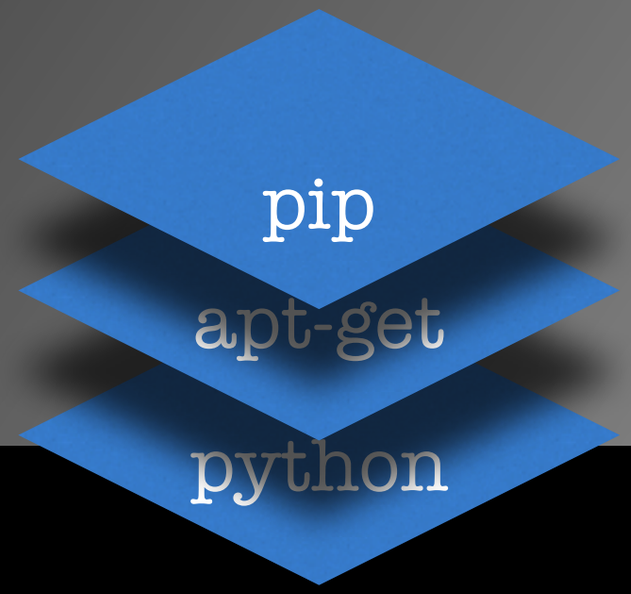
- for Linux
- for macOS
- for Windows







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 bits 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..."]
```

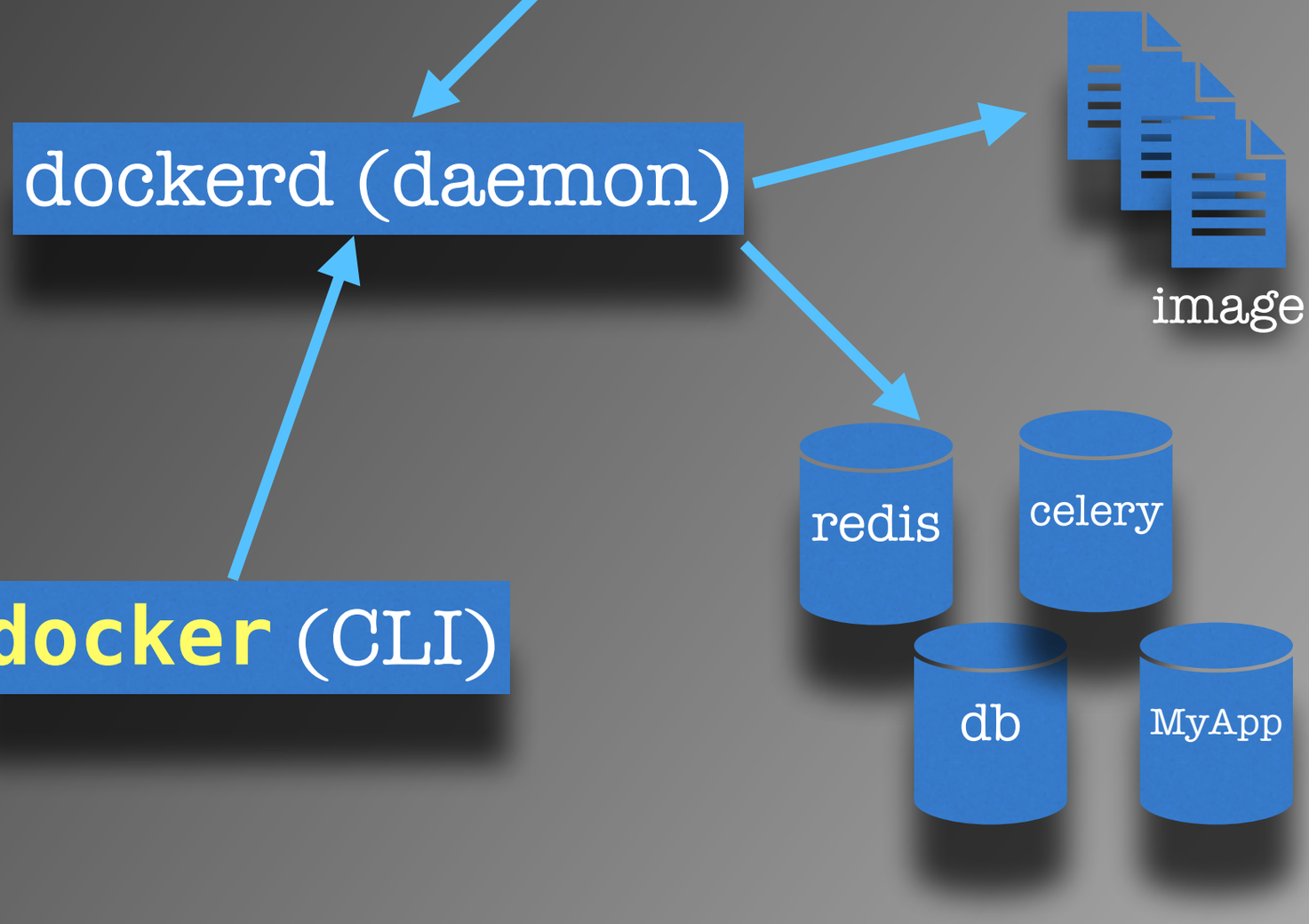
Docker Desktop

- for Linux
- for macOS
- for Windows

Docker

Docker Desktop

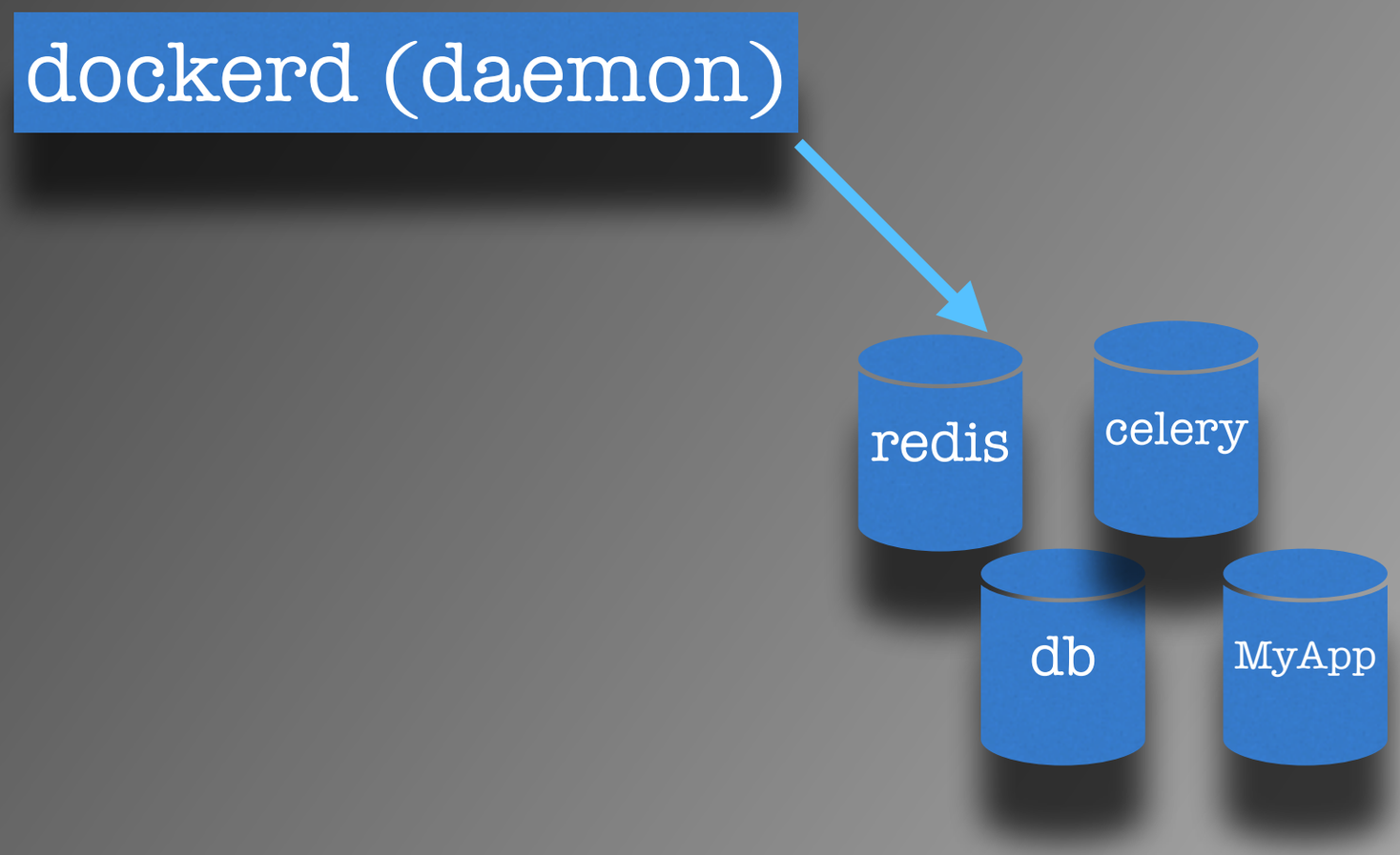
- for Linux
- for macOS
- for Windows



Docker

Docker Desktop

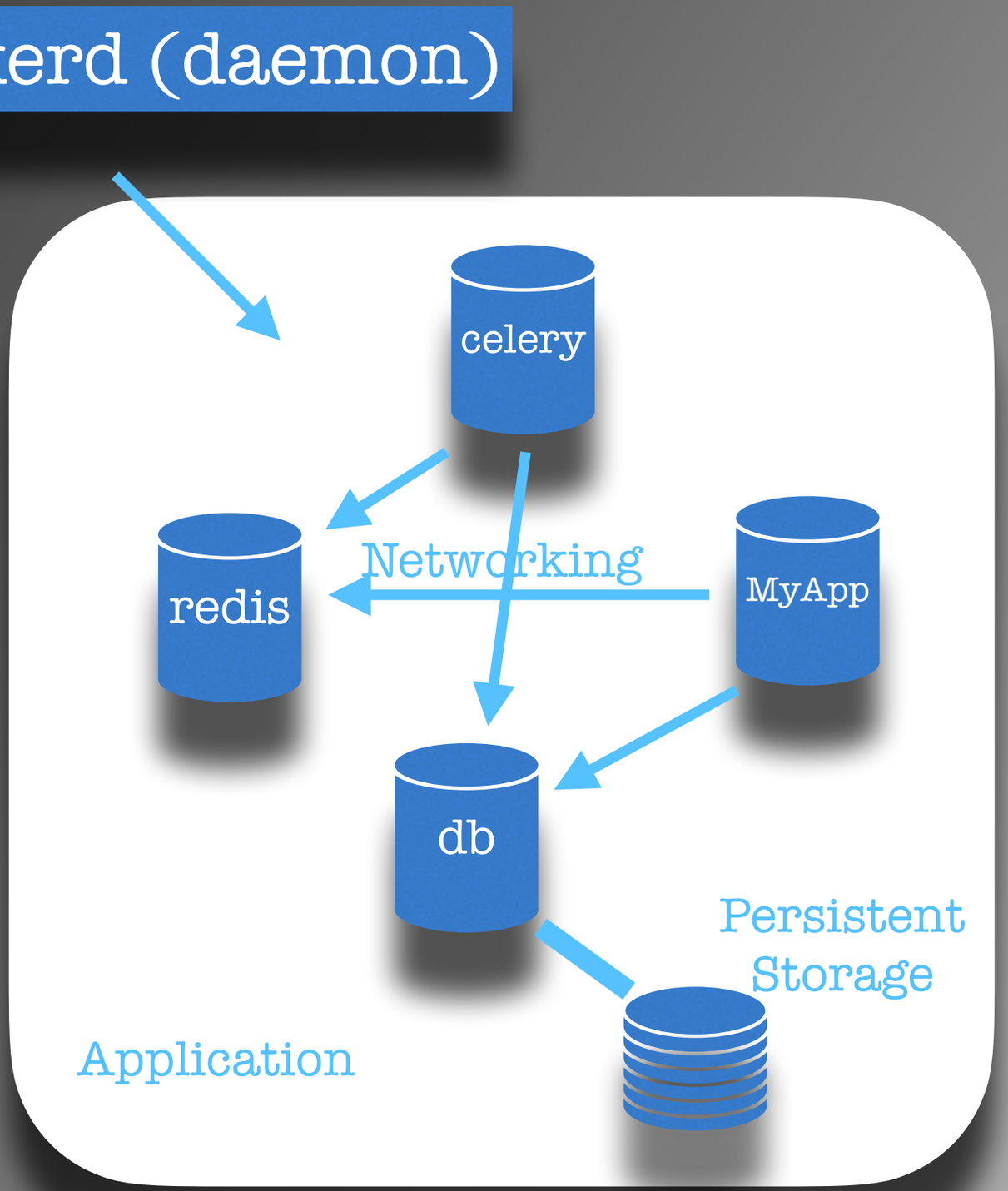
- for Linux
- for macOS
- for Windows



Docker

Docker Desktop

- for Linux
- for macOS
- for Windows



Docker Compose

Docker Desktop

- for Linux
- for macOS
- for Windows

docker-compose.yml

```yaml
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

Docker Desktop

- for Linux
- for macOS
- for Windows

What are the alternatives?

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

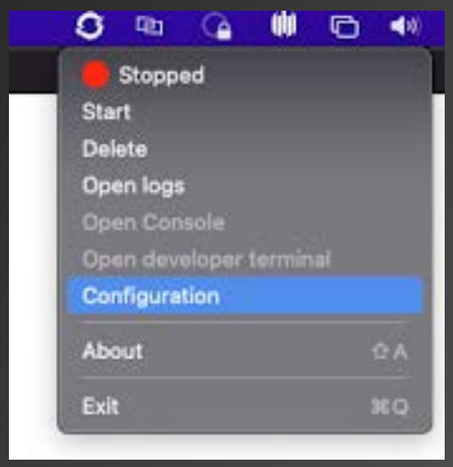
Colima - container runtimes on macOS (and Linux) with minimal setup.

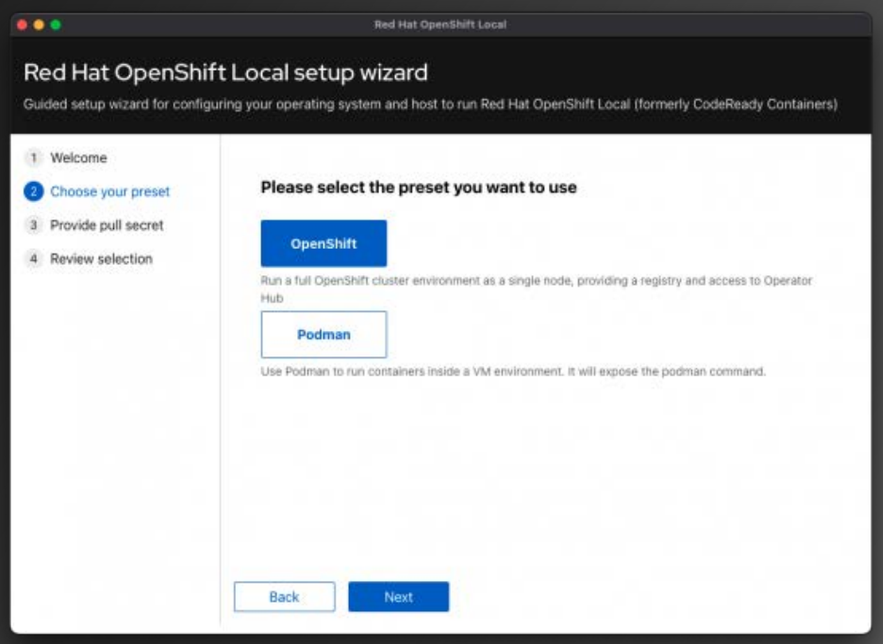
Features

- 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://developers.redhat.com/products/openshift-local/overview

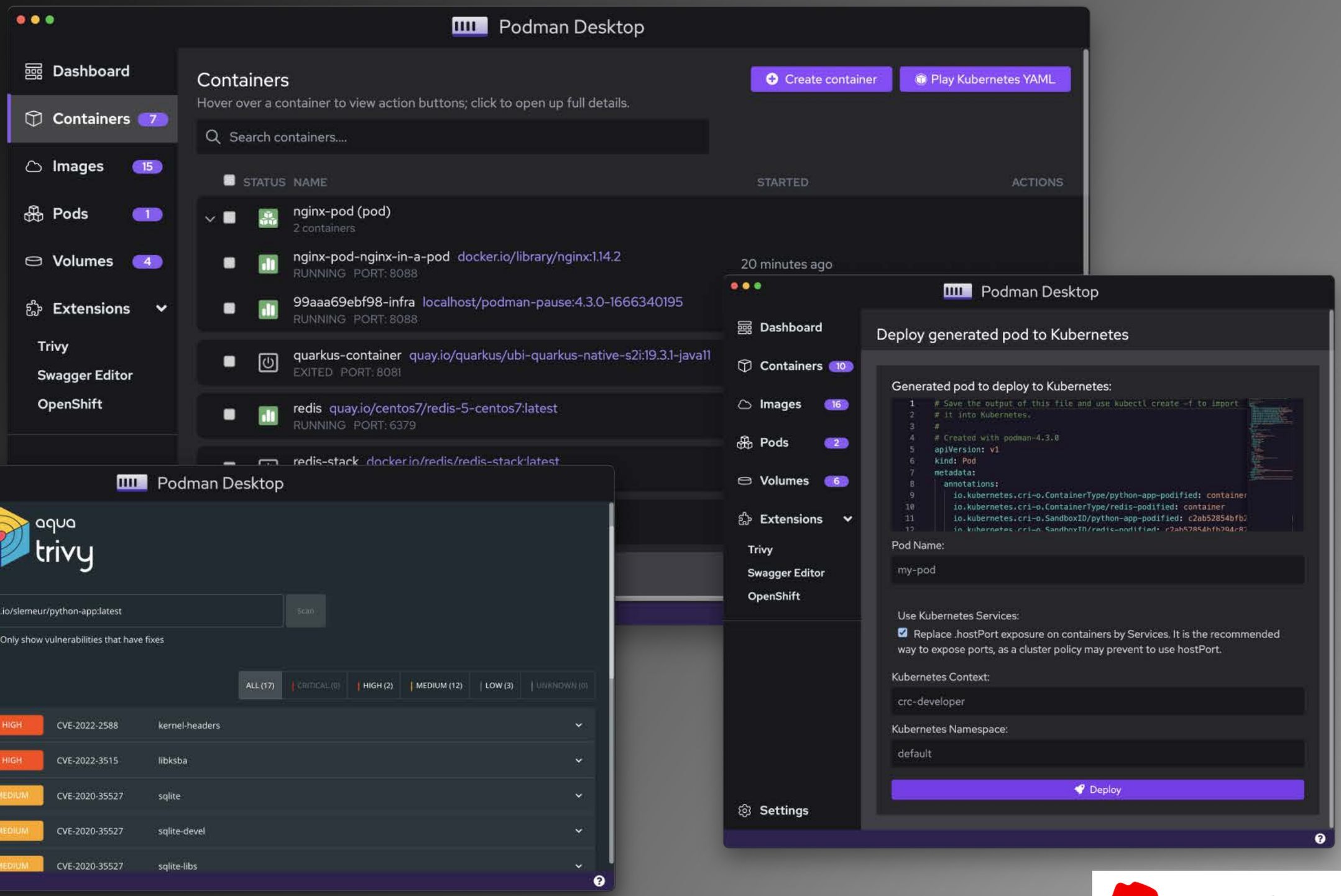
podman

"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/

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Podman Desktop



THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

https://podman-desktop.io/

Red Hat

Orchestration

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

Examples of Container Orchestration Tools

- Kubernetes
- Docker Swarm
- Apache Mesos
- HashiCorp Nomad

Kubernetes (K8s)

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Kubernetes (K8s)

Cluster

- **Control Plane**
  - API
  - Etcd
    - distributed key/ value store

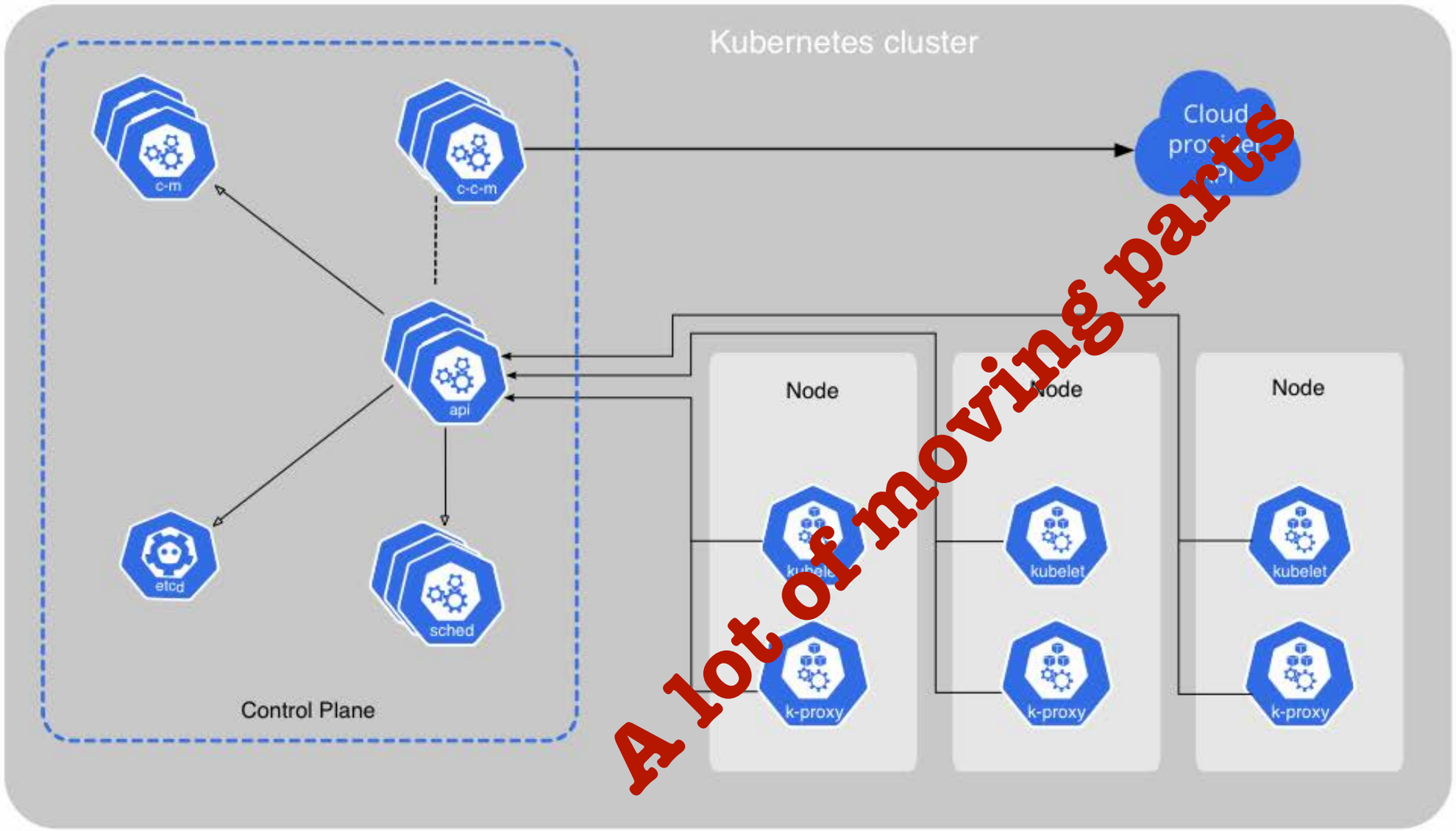
kubectl

"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)



"Containers on steroids"

Kubernetes (K8s)

Kubernetes (K8s)

Kubernetes (K8s)

"The package manager for Kubernetes"
— https://helm.sh/

YAML

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

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

https://kompose.io/

Kubernetes Implementations

Google K8s

kubectl

minikube

kind

Red Hat

okd → RED HAT OPENSHIFT Container Platform

CANONICAL Ubuntu

MicroK8s

RANCHER

K3S

RANCHER DESKTOP

nerdctl

MIRANTIS

kOS

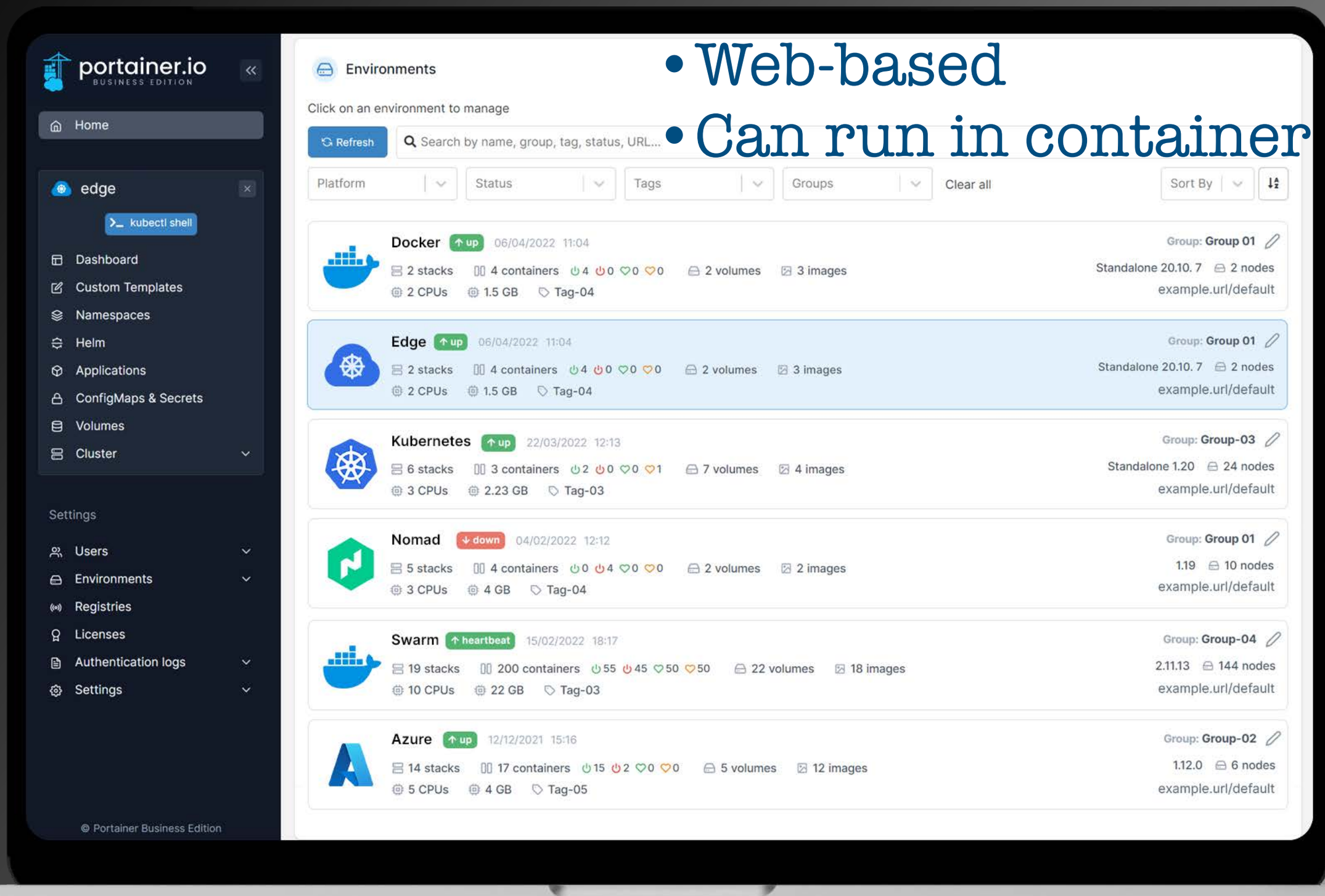
kOsctl

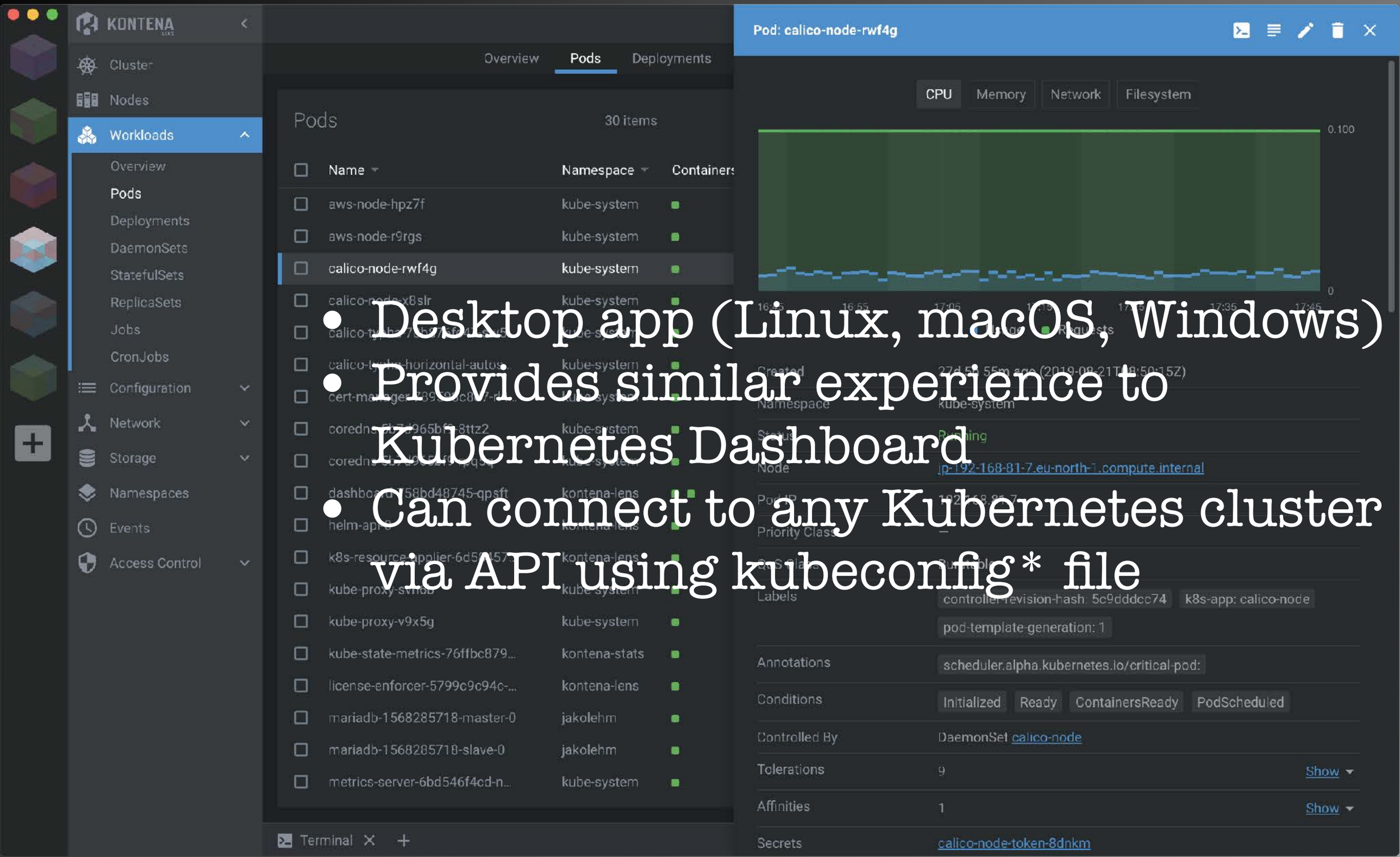
LENS

portainer.io

portainer.io

- Web-based
- Can run in container



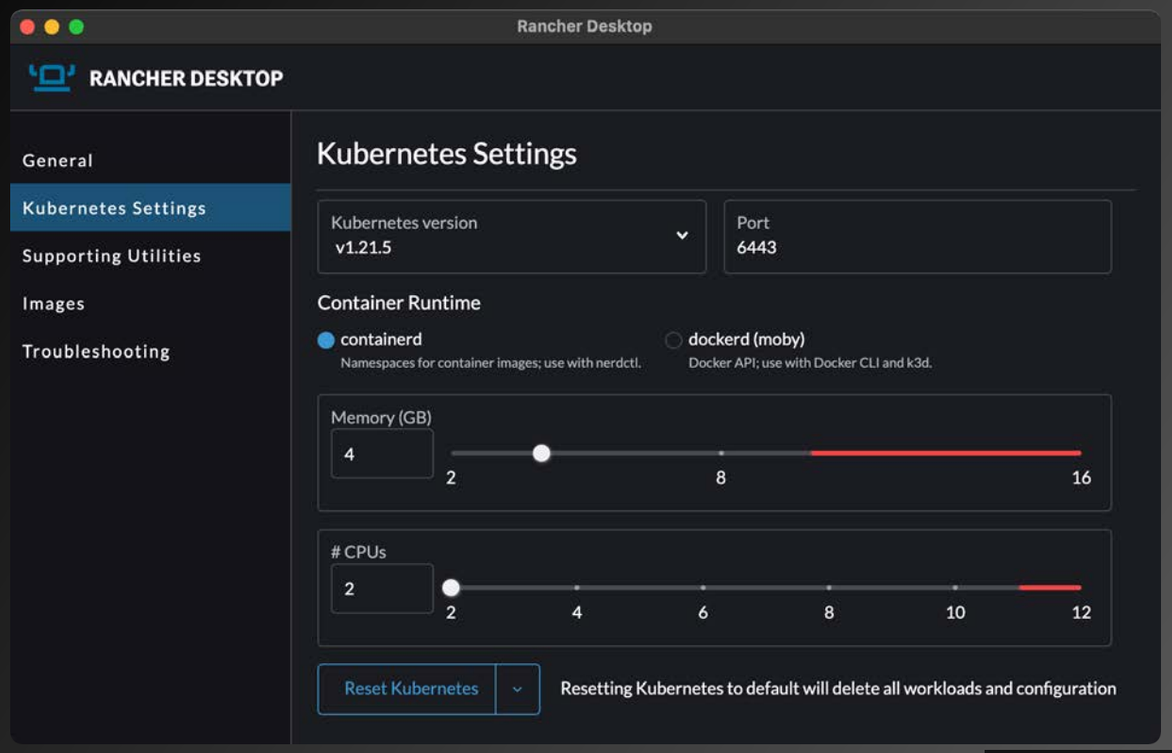


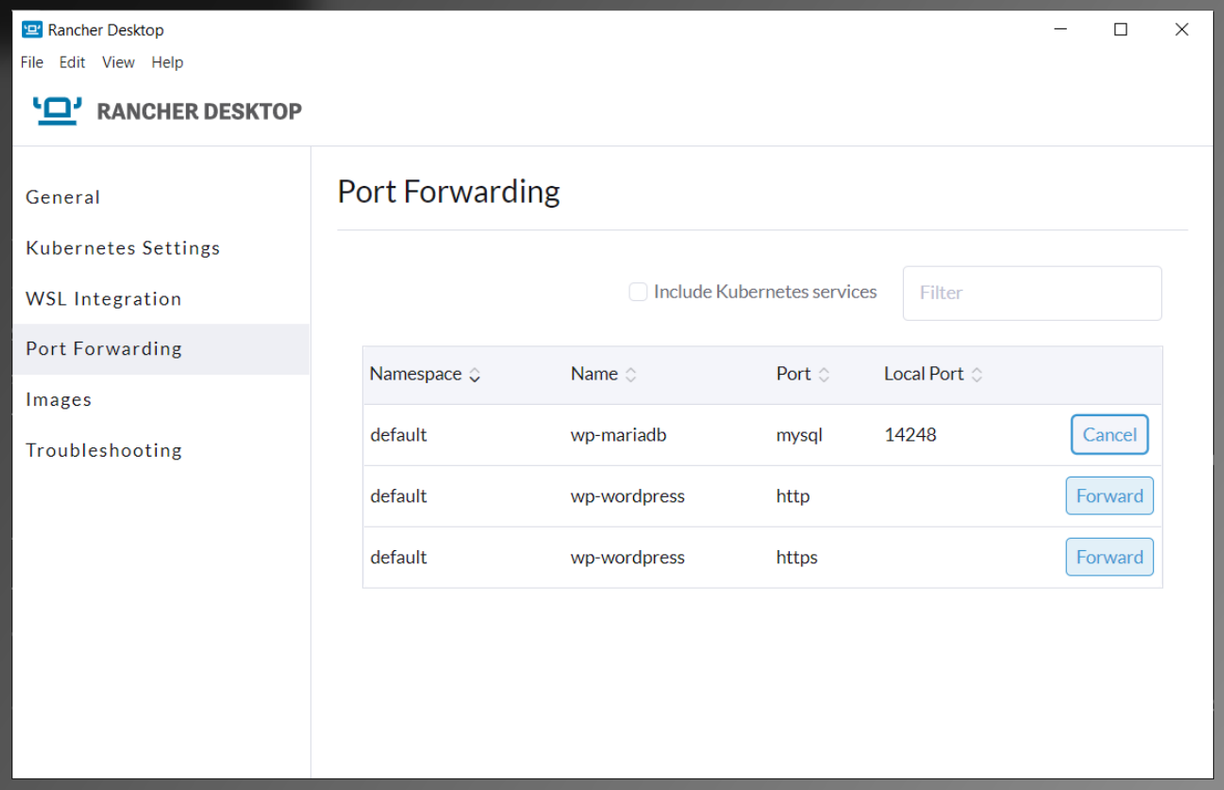
LENS

- 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

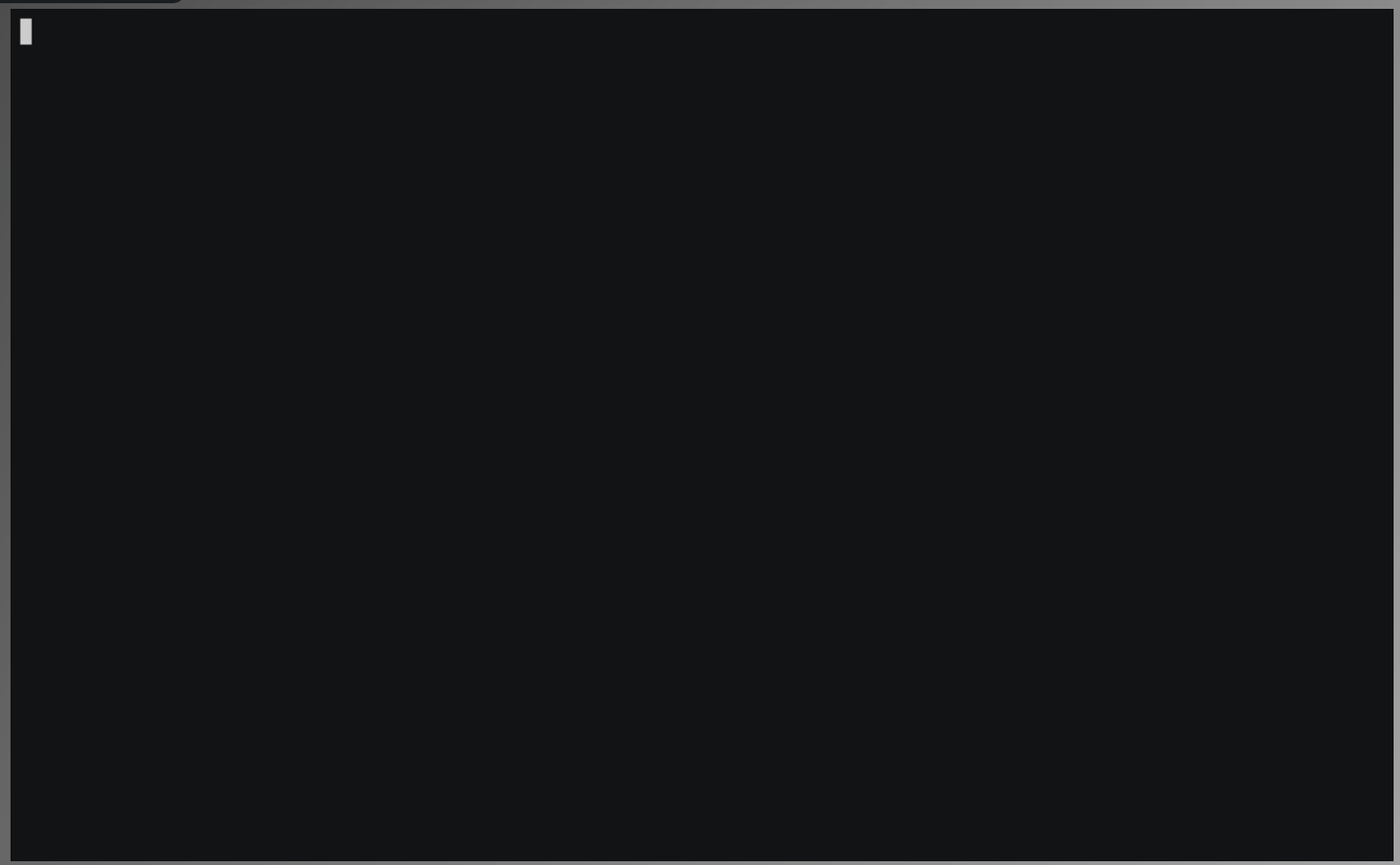
RANCHER DESKTOP



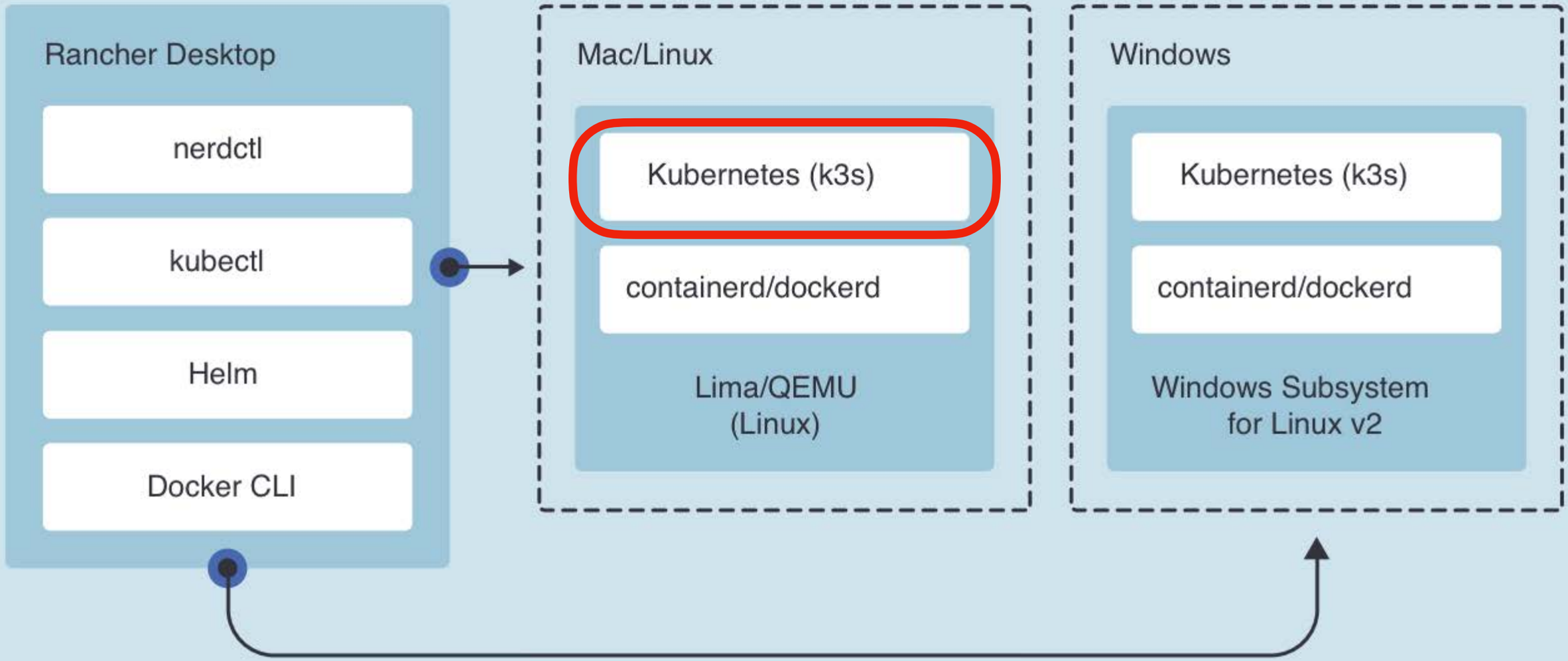


containerdctl

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



RANCHER DESKTOP



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

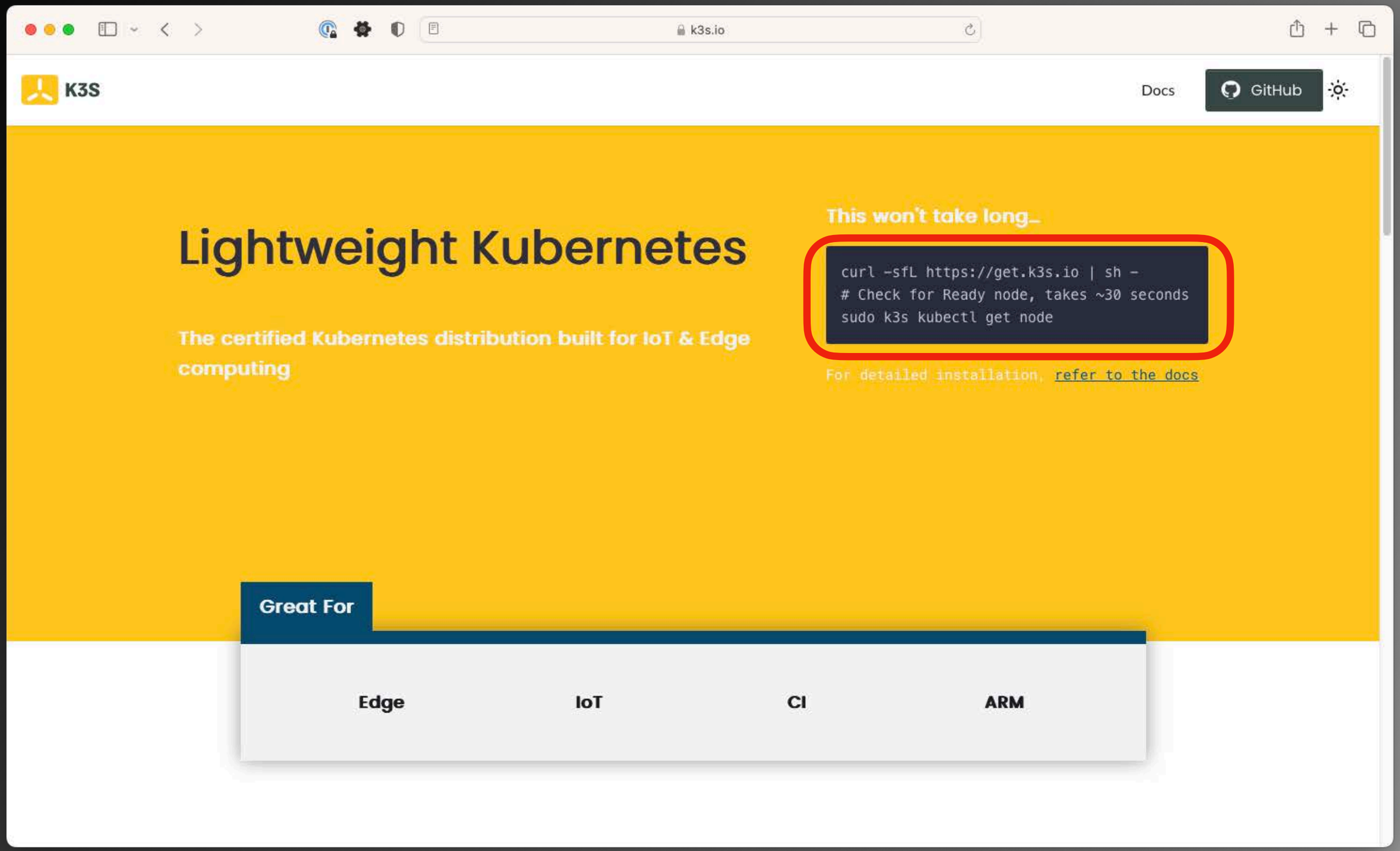
ADVANTAGES:

- 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.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 -sfL 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

RED HAT® OPENSHIFT Container Platform

=

Kubernetes (K8s)

Kubernetes is
Kubernetes is
Kubernetes

+

But what does this really mean?

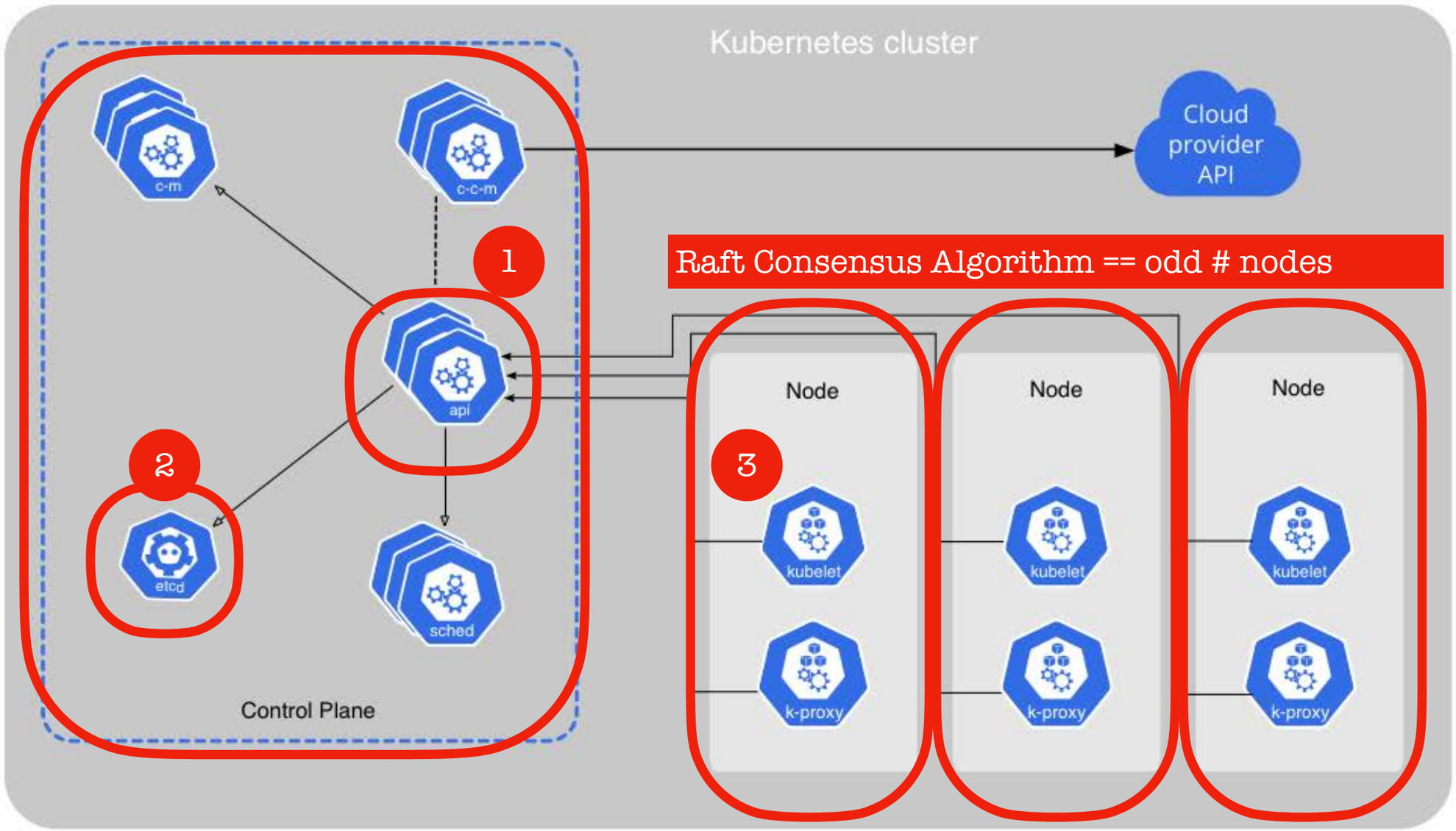
What does Kubernetes even DO for us?

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Traditional App Deployment

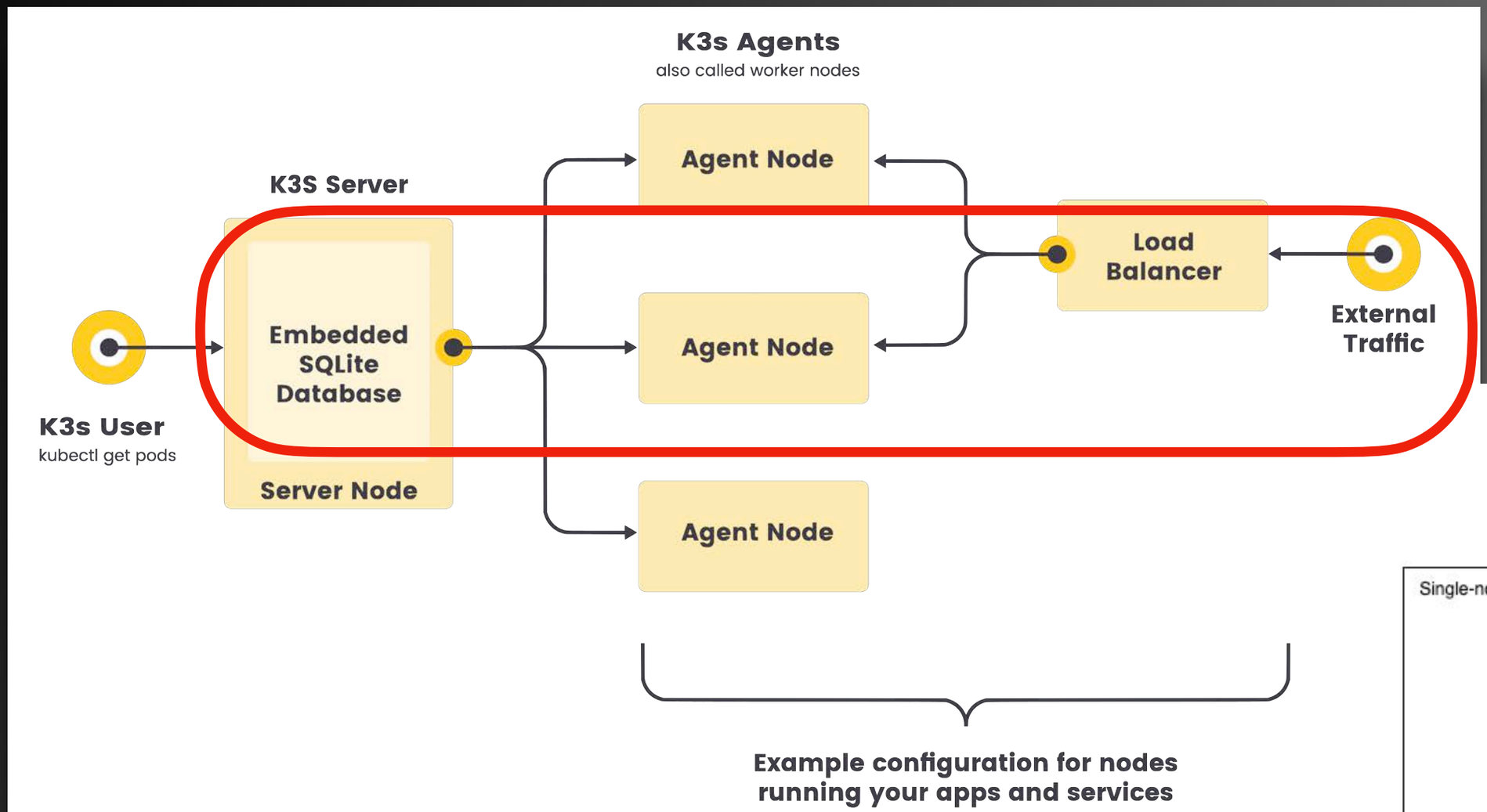
- Build a server (e.g., a "LAMP" stack with Linux, Apache, MySQL, Perl/PHP/Python)
- Install application code
- Configure all the bits (“artisanal”)
- Ignore scaling issues, 10K problem, etc.

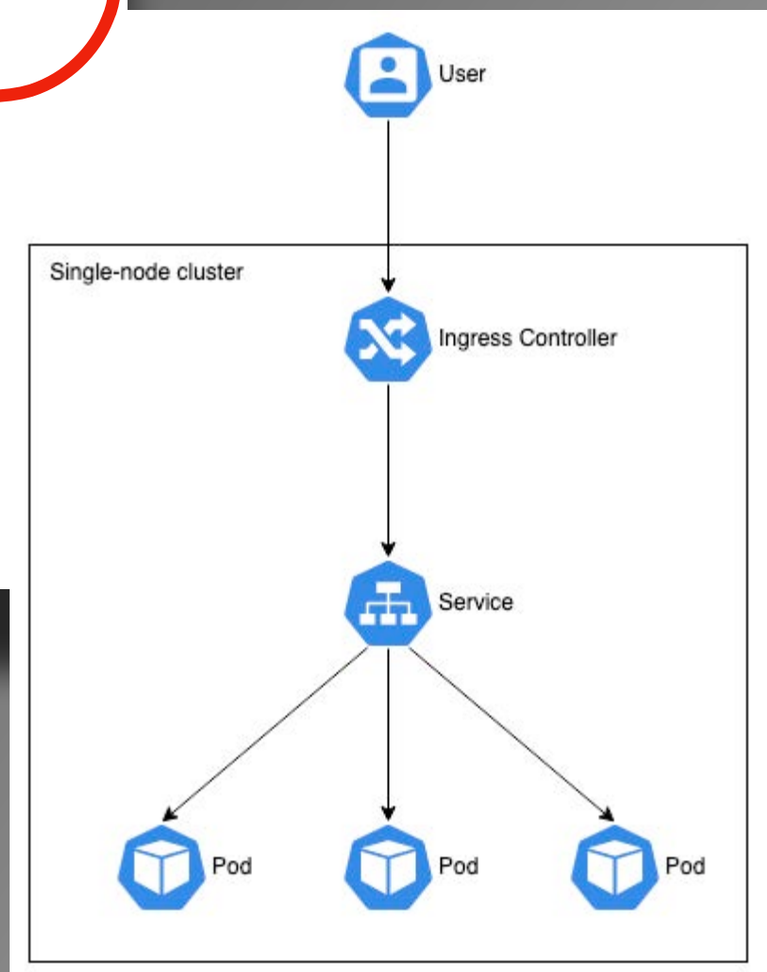
Full-blown Kubernetes (K8s) Architecture



Normally min. 4 nodes

K3s - Single Node Cluster





Minimum can be a single node for everything

K8s Default Setup

6443/TCP

Namespace

kube-system

80/TCP
443/TCP

ingress cntlr

53/TCP
9153/TCP

kube-dns

443/TCP

metrics-server

All pods assigned 10.43.x.x

default

443/TCP

Kubernetes API

kube-public

kube-node-lease

K3s Default Setup

6443/TCP

Namespace

kube-system

80/TCP
443/TCP
Traefik

53/TCP
9153/TCP
CoreDNS

443/TCP
metrics-server

All K3s pods assigned 10.43.x.x

default

443/TCP
Kubernetes API

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

K3S

K3s Default Setup

6443/TCP

Namespace

kube-system

80/TCP
443/TCP

Traefik

All K3s pods assigned 10.43.x.x

default

443/TCP

Kubernetes API

K3S

K3s Default Setup

6443/TCP

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

8000/TCP

Django/gunicorn

container

5432/TCP

PostgreSQL

container

5672/TCP

Celery (or Redis,...)

K3S

K3s Default Setup

6443/TCP

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

K3s Default Setup

appFQDN:80

Internet

6443/TCP

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")

80/TCP

80/TCP → 8000/TCP

container: Django/gunicorn

5432/TCP → 5432/TCP

container: PostgreSQL

5672/TCP → 5672/TCP

container: Celery (or Redis,...)

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

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`**.

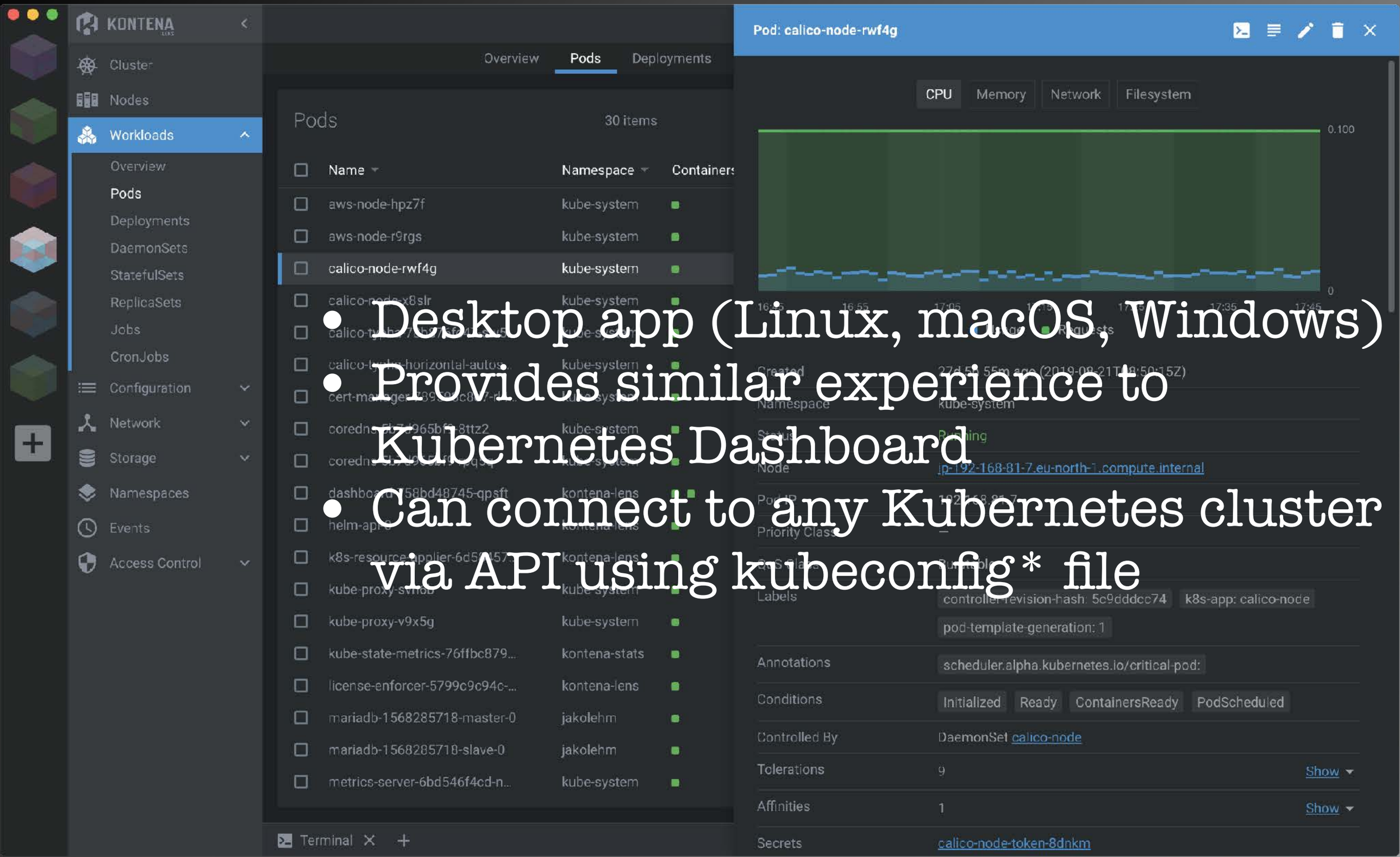
Kubectl

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: LS0tLS1CRUdJTiBFQyBQUklWQVRFIEt...
- name: fseesink/api-cloudapps-unc-edu:443
  user:
    token: sha256~lR6K25TO0CuswoJNtKEG03Fshy-9C028vcMZxmFwKP0
- name: rancher-desktop
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZ...
    client-key-data: LS0tLS1CRUdJTiBFQyBQUklWQVRFIEt...
```



LENS

- 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: LS0tLS1CRUdJTiBDRVJU
    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

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

So How are Kubernetes Clusters Managed?

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Application Manifests

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

YAML

- Namespace (optional)
- Deployment
- Service
- Ingress

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

```yaml
---
#################################################################
# 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

```yaml
---
##################################################################
# 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

```yaml
---
#####################################################################
# 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
```

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

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

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

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

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

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-5qhrg                        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
```

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

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.

```yaml
---
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: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUd3VENDQmFtZ0F3SUJBZ0lRZHNFRE...
  tls.key: LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCk1JSUV2Z0lCQURBTkJna3Foa2lHOXcwQk...
```

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/

CNCF Cloud Native Landscape

1.0

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

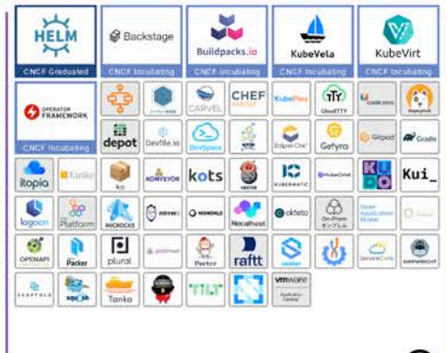
Greyed logos are not open source

App Definition and Development

Database

Streaming & Messaging

Application Definition & Image Build



Continuous Integration & Delivery

Orchestration & Management

Scheduling & Orchestration

Coordination & Service Discovery

Remote Procedure Call

Service Proxy

API Gateway

Service Mesh

Runtime

Cloud Native Storage

Container Runtime

Cloud Native Network

Provisioning

Automation & Configuration

Container Registry

Security & Compliance

Key Management

Special

Kubernetes Certified Service Provider

Kubernetes Training Partner

Certified CNFs

Platform

Certified Kubernetes - Distribution

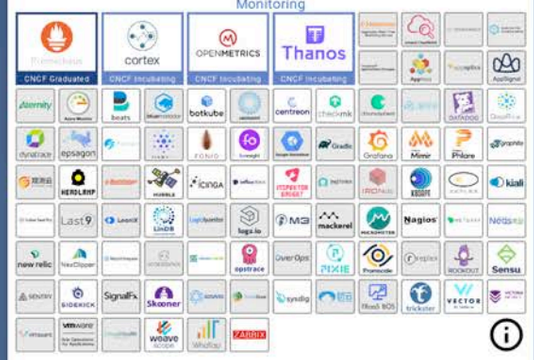
Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Observability and Analysis

Monitoring



Logging

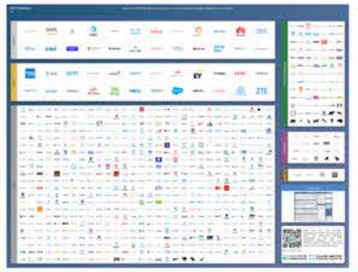
Tracing

Chaos Engineering

Continuous Optimization

Serverless

Members



CD Foundation Landscape

CLOUD NATIVE LANDSCAPE

CLOUD NATIVE COMPUTING FOUNDATION

Redpoint Amplify

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.

l.cncf.io

Thank You

https://frank.seesink.com/presentations/Internet2TechEx-Fall2023/

Frank Seesink
frank@seesink.com
frank@unc.edu

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL