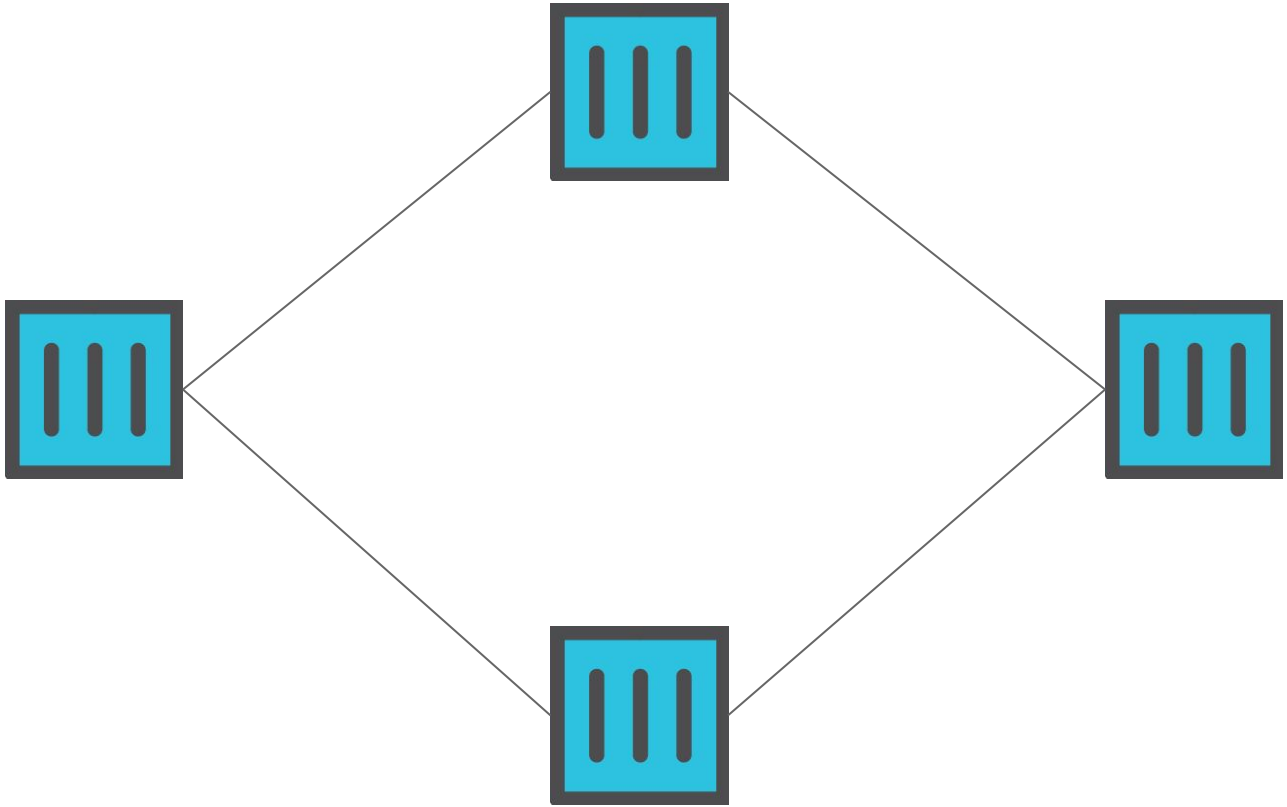
ThoughtWorks®

KUBERNETES

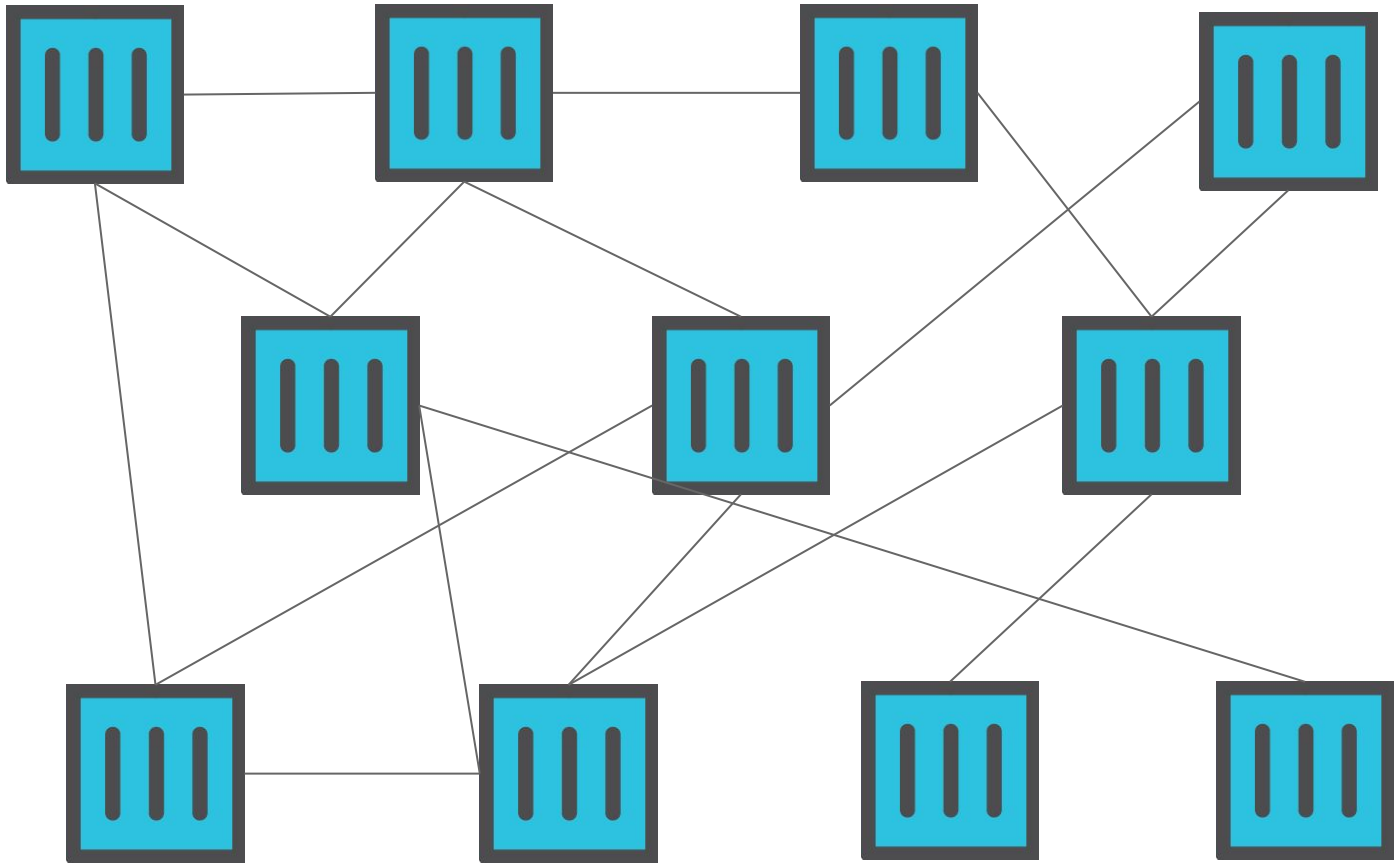
Rise of the Containers Workshop

World of Containers

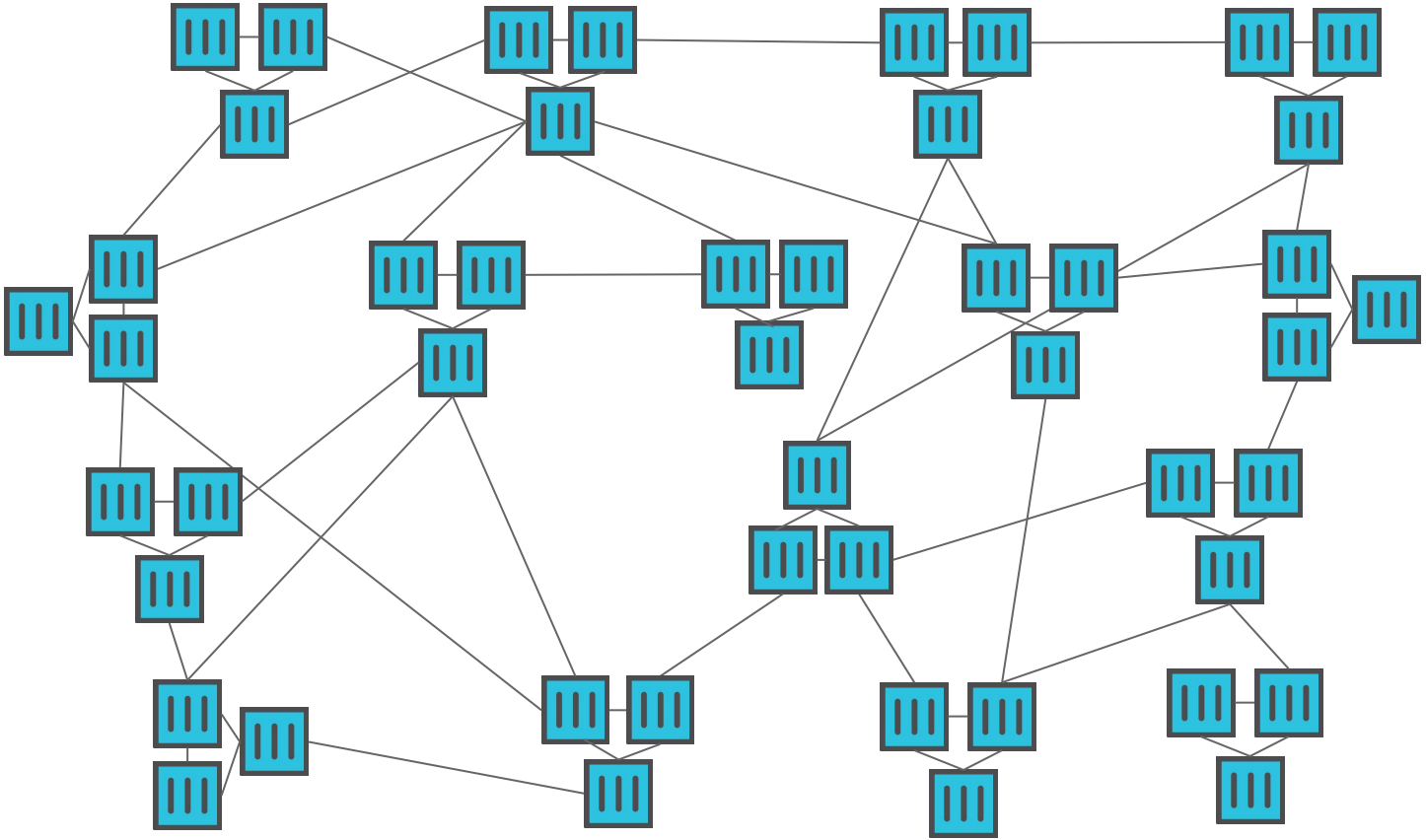


Sounds Simple?

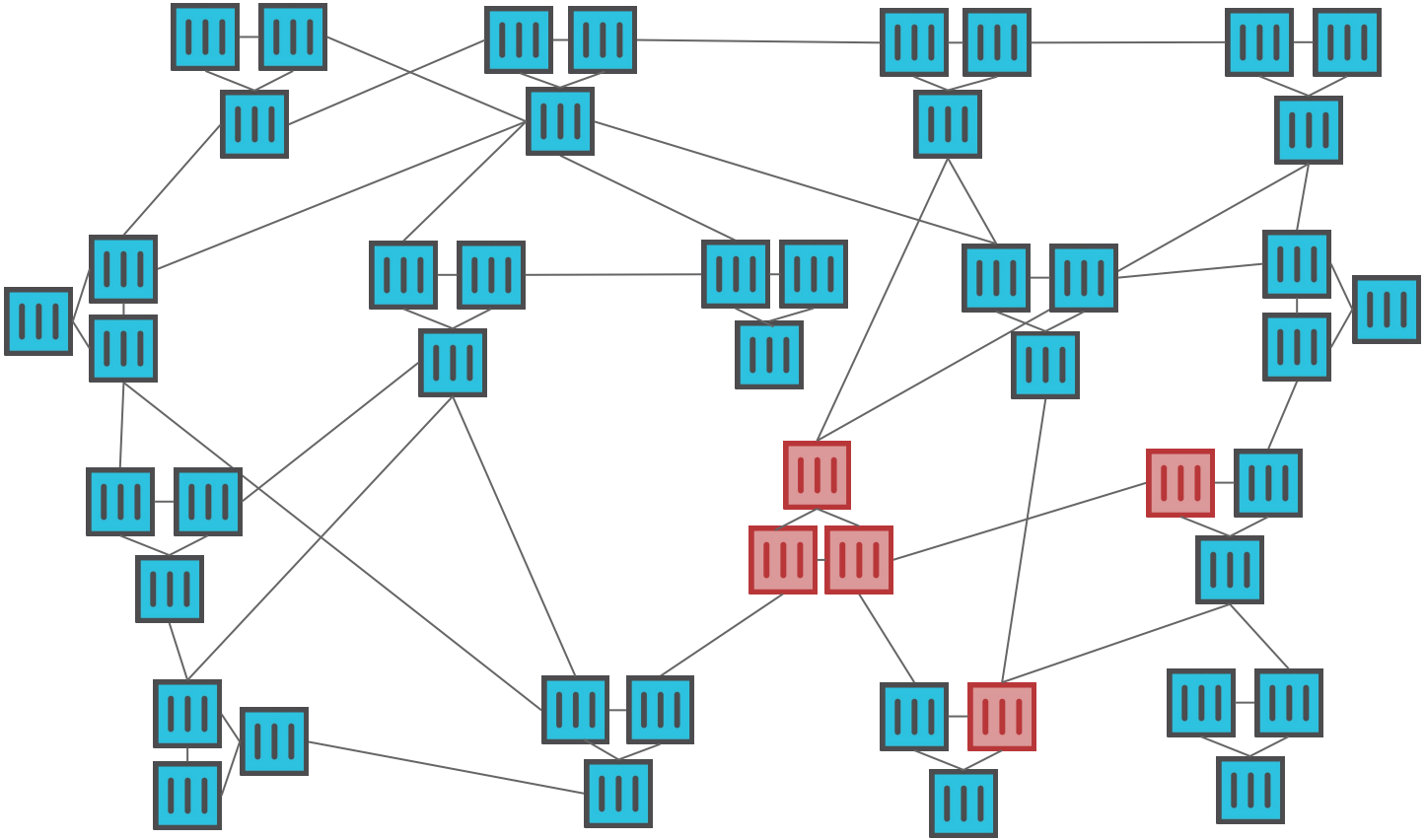
World of Containers



World of Containers



World of Containers



Maybe…Not So Simple

Solution

kubernetes

What is it?

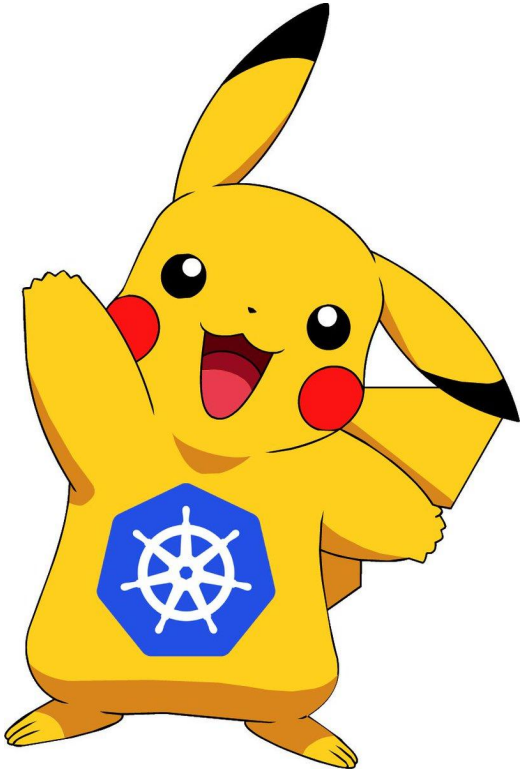
Kubernetes is a software system that allows you to easily deploy and manage containerized applications on top of it.

Brief History

- Greek for “helmsman” or pilot
- First announced in mid-2014, as an all-Google project
- In mid-2015, Google + Linux Foundation came together to form CNCF

Kubernetes Users


9:41 AM
100%
5
Sara

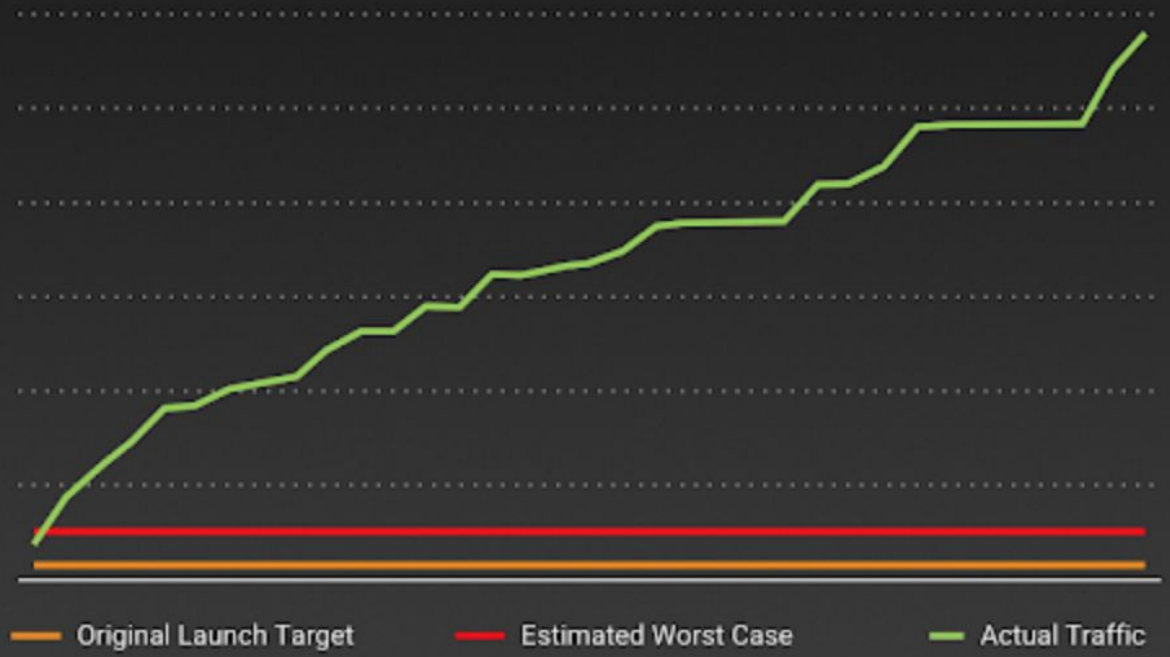


Cloud Datastore Transactions Per Second

1X
Target Traffic

5X
Worst Case Estimate

50X
Actual Traffic



Google Cloud

oscon OPEN SOURCE CONVENTION oscon 20 YEARS 2018

Most Impact Award

Ansible

Configuration

Works with hosts directly

Provision a system with the required config

On-demand system

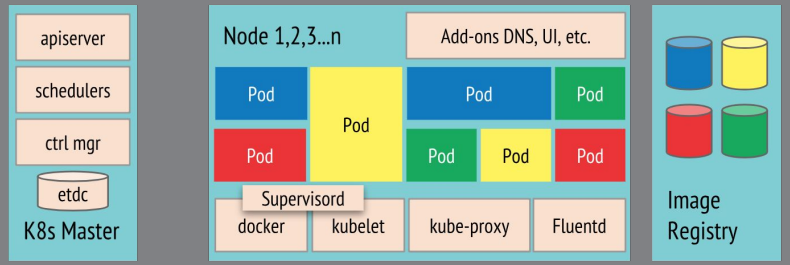
Kubernetes

Orchestration

Works with containers

Deploy ready-made images to infra

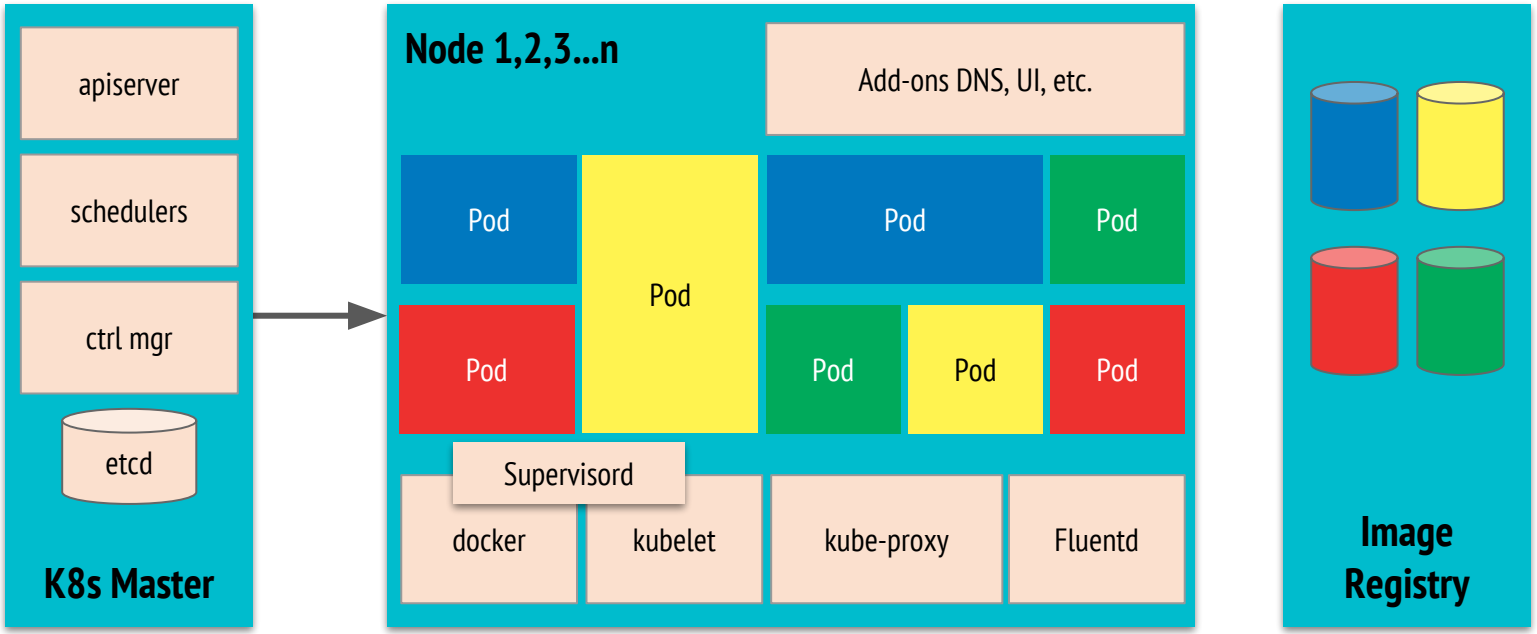
Live, realtime system



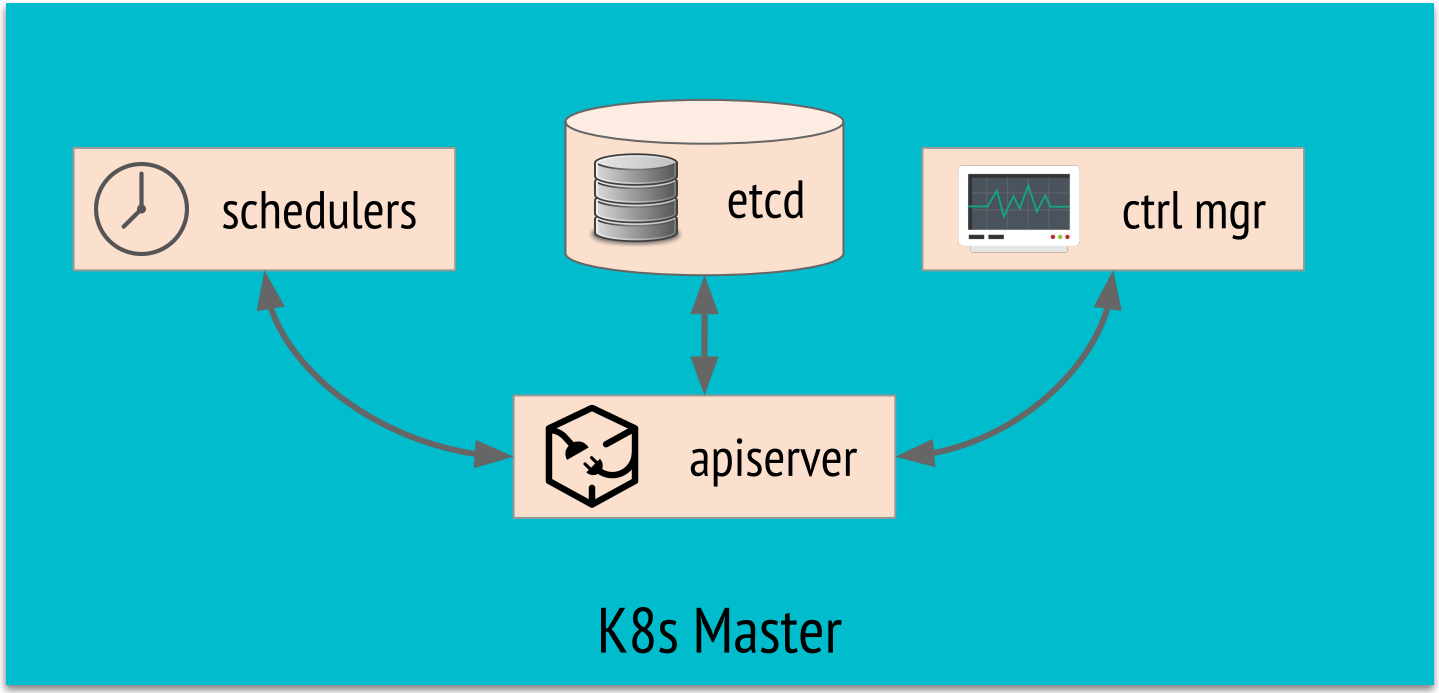
Kubernetes Architecture

Getting perspective at 30000 feet

High-level architecture

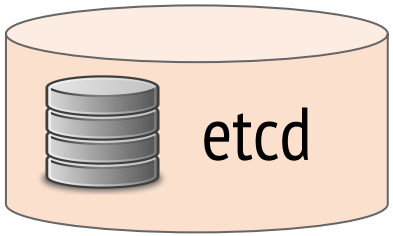


Master Components



etcd

- Distributed, key-value store
- Used to store all cluster data
- The only stateful component in kubernetes
- “Source of Truth”



kube-apiserver

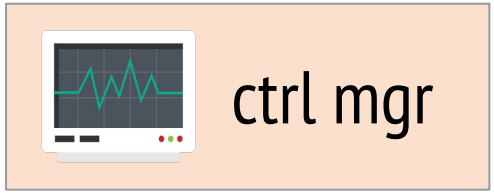
- Frontend to the “control plane”
- Exposes a REST API to interact with kubernetes
- Often mistaken for being the master

kube-scheduler

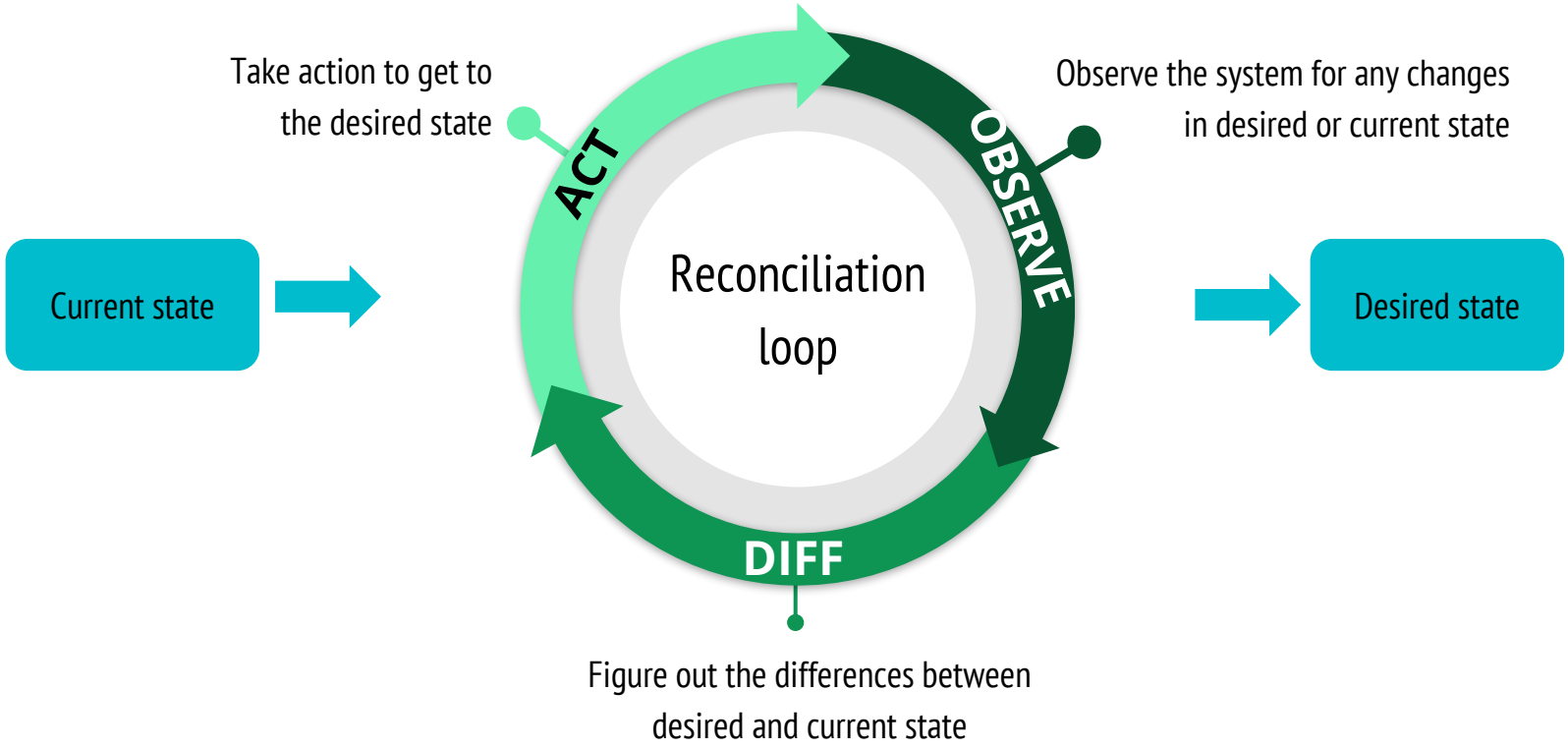
- Assigns a node for newly created pods
- Various options to choose from for affinity
- Continuously keeps watching store for new pods

kube-controller-manager

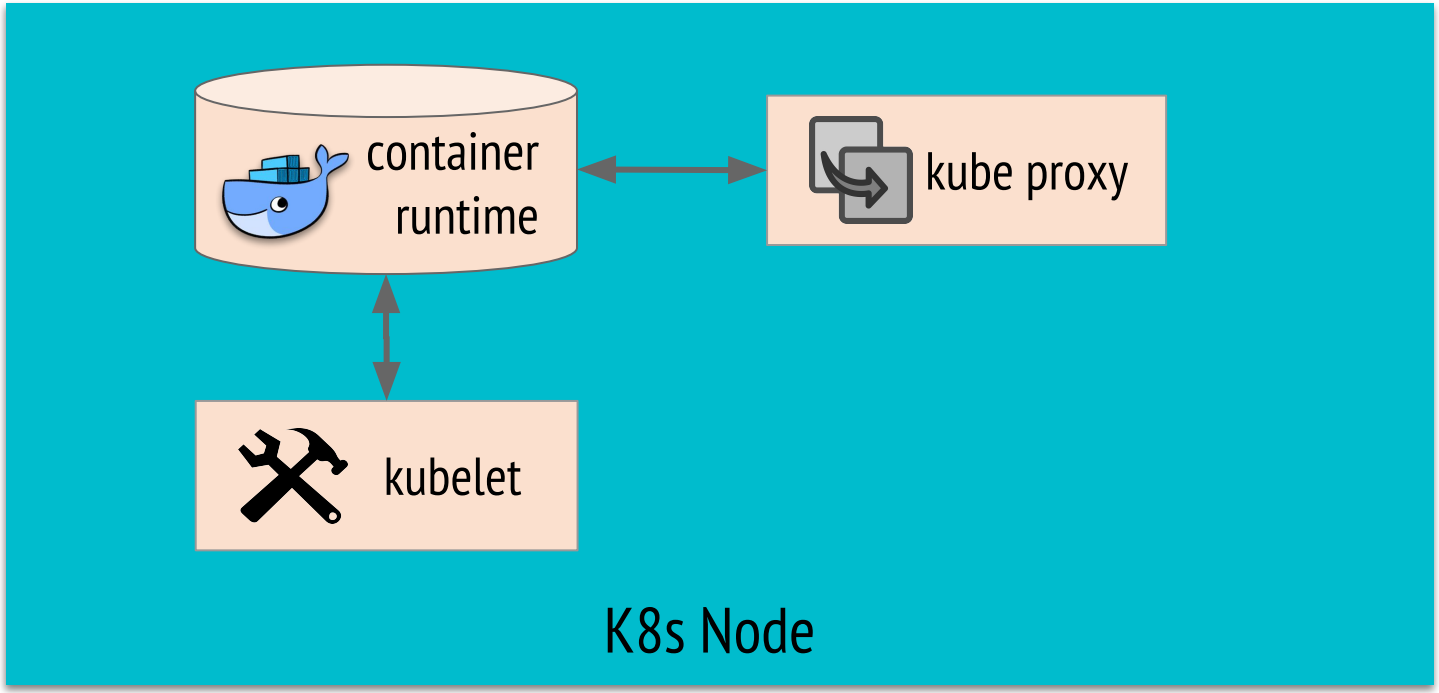
- Runs “controllers” in kubernetes
- Controllers are daemons that ensure the desired state is achieved in the cluster
- Examples of controller:
  - Node Controller
  - Replication Controller
  - Endpoints Controller



Reconciliation

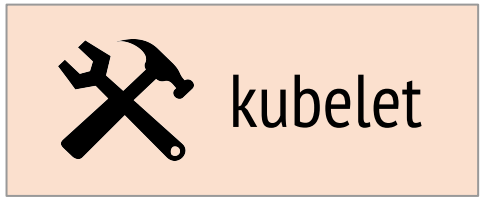


Node Components



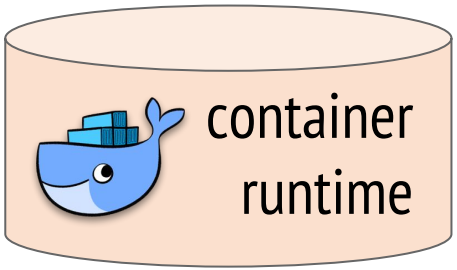
kubelet

- Main agent on the node, often mistaken for the node itself
- Keeps watching the apiserver for work
- Instantiates pods
- Reports to master



Container Runtime

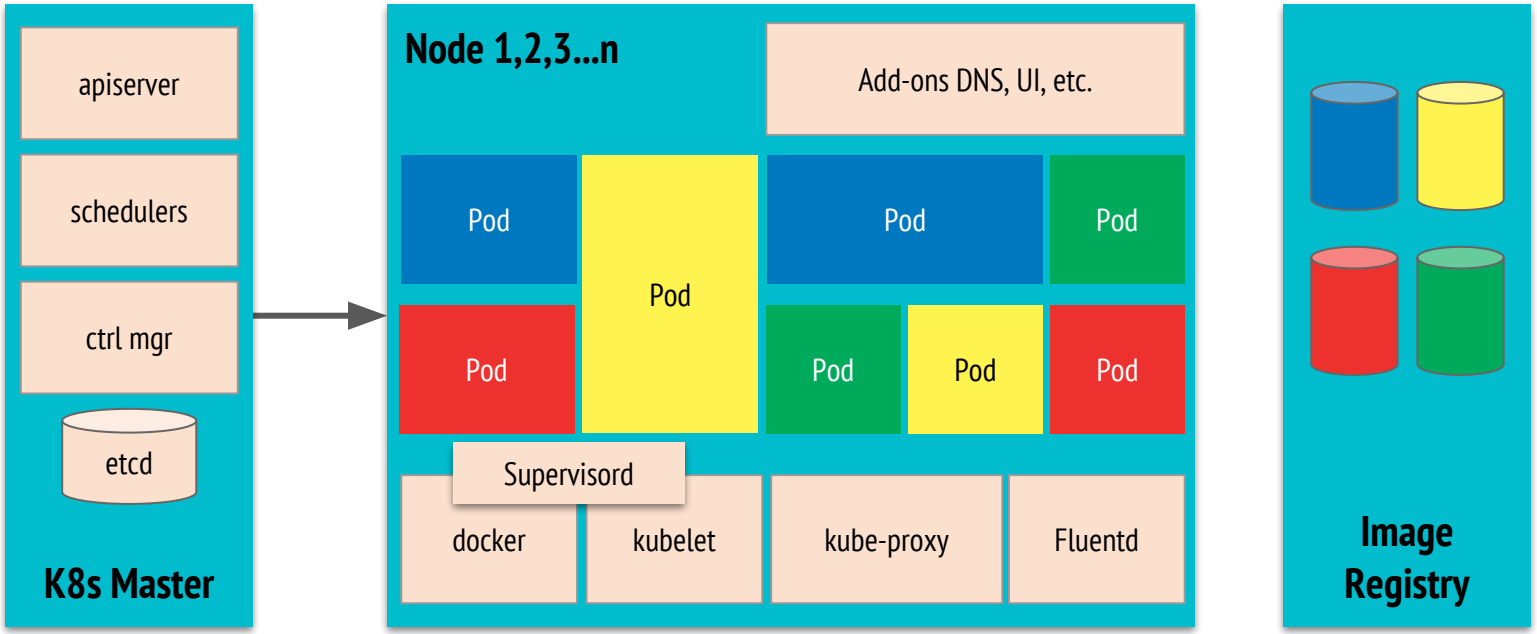
- Deals with the container abstraction
- Pull images, start/stop containers, etc
- Works with any OCI compliant container engine
- Usually docker, but also supports rkt



kube-proxy

- Deals with networking within the node
- Assigns an IP to each “pod”
- Is primarily used to maintain the “service” abstraction

High-level architecture



minikube

Hands on Minikube

Let's kick things off with **`$ minikube start`**

kubectl the command line

```
kubectl [command] [TYPE] [NAME] [flags]
```

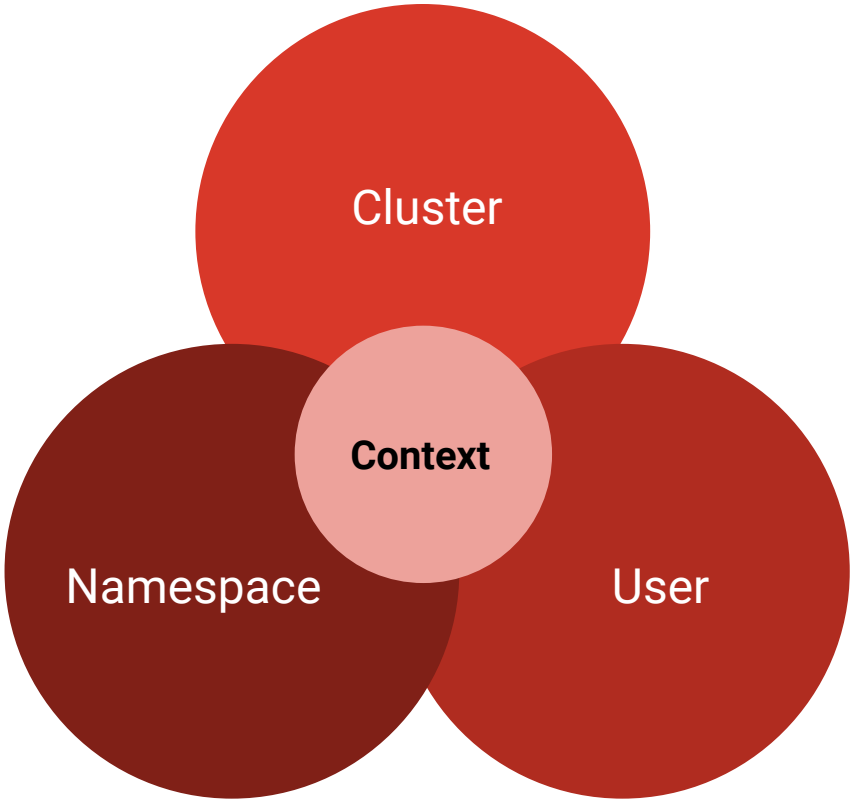
Basic commands to inspect k8s

```
$ kubectl version

$ kubectl cluster-info

$ kubectl config view
```

kubectl config



kubectl

config context

```
$ kubectl config get-contexts

$ kubectl config current-context

$ kubectl config use-context
```

kubectl *namespaces*

```
$ kubectl get namespaces

$ kubectl get pods --namespace
kube-system

$ kubectl describe pod <podName>
```

kubectl

running applications

```
$ kubectl run hello-world \
    --image=gcr.io/google-samples/node-hello:1.0 \
    --port=8080

$ kubectl get pods
```

THANK YOU

For questions or suggestions:

Ankita Luthra

ankital@thoughtworks.com

kubectl

access applications

```
$ kubectl expose deployment
hello-world \
--type=NodePort \
--name=hello-world-service

$ kubectl describe services \
hello-world-service

$ curl http://<minikube-ip>:<nodePort>
```

Quick exercise!

Run the **metadata** service using **kubectl**

NOTE:

- Point local docker client to minikube's docker:
  ```
  $ eval $(minikube docker-env)
  ```
- Build the metadata image again before attempting to run it