Docker From The Ground Up Part 4
Kubernetes 101

A whistle-stop tour of Kubernetes architecture and control/data planes.

Docker Birmingham Meetup 12/12/2018

kubernetes

Thanks to our Sponsor!

Black**Cat** /

Thanks to Katacoda

For providing a great learning environment/demos

Powered by Katacoda

$ whoami

Shaun McLernon

shaun@codesome.tech

Independent software engineer with 20+ years in development roles, and 5 years in ”devops” space.

Github: @smclernon

Docker Birmingham Slack: @shaun

kubernetes

Recap

CONTAINERIZE
ALL THE THINGS

Requirements for Dev -> Production

- Multi-machine
- Discovery and Naming
- Scaling
- Multiple users
- Failure tolerance and recovery
- Monitoring
- Logging
- High availability
- Deployment lifecycle
- Load balancing
- etc, etc

Alternatives

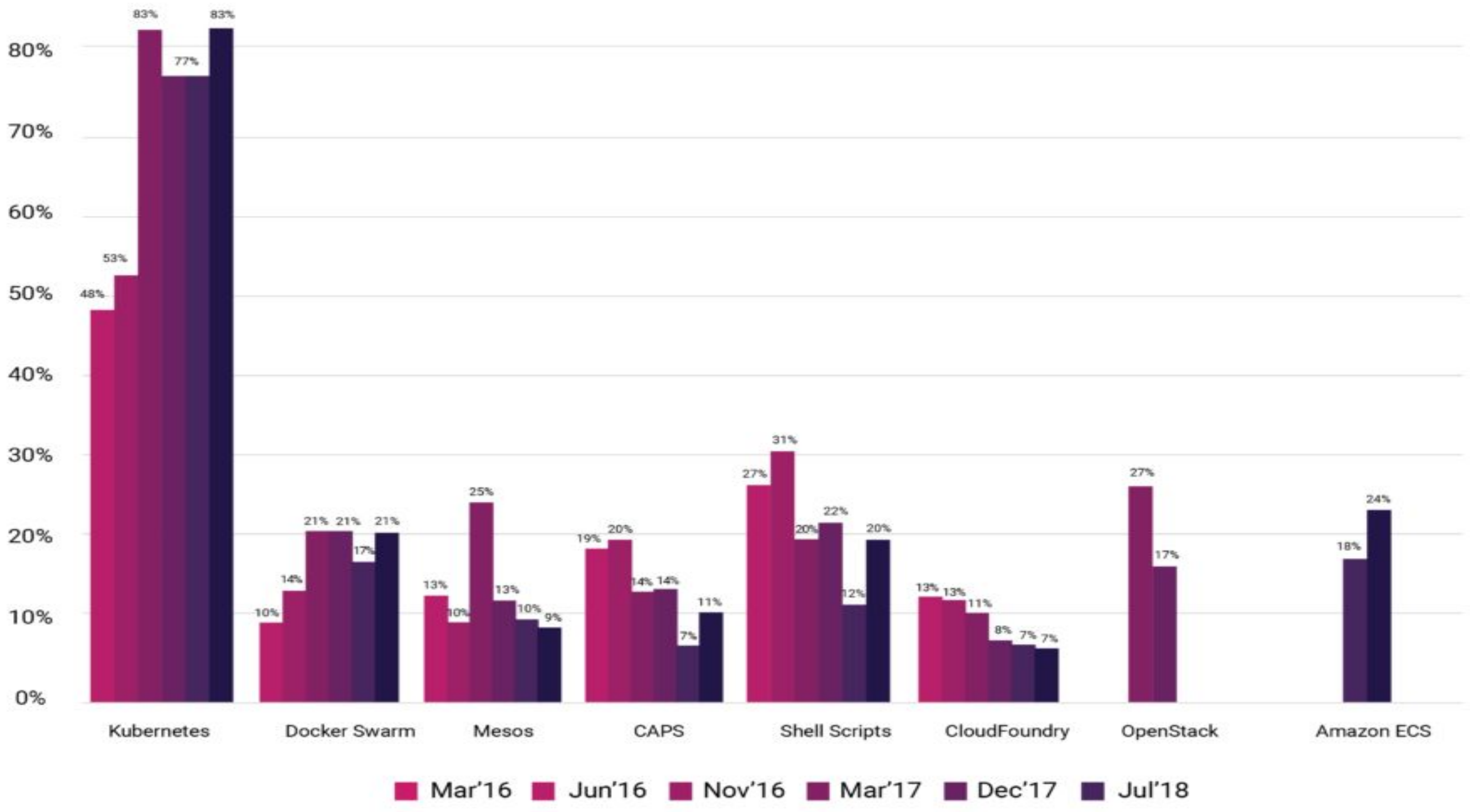


Image ref: https://www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-in-production-has-grown-over-200-percent/

Poll: Developers/Operators spread?

Kubernetes
K8s

Greek for "Helmsman" < person who steers a ship

K8s is an abbreviation derived by replacing the 8 letters “ubernete” with “8”.

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

What does that really mean?

Kubernetes is a "Container Orchestrator"

- Scheduling: put containers on machines
  - by resource needs (CPU, memory)
  - by affinity requirements (put X near Y)
  - by labels put X on a "GPU" machines
- Replication: run N copies
- Discovery: find peers and services in other containers
- Recovery: automatically from failure
  - Health checking
  - Application failures
  - Machine failures
- Inspection: tell me what is happening
  - Basic monitoring
  - Logging

Where did it come from?

- Created by Joe Beda, Craig McLuckie (Heptio) and Brendan Burns (Microsoft)
- Based on ideas proven at Google for a decade and a half of experience that Google has with running production workloads at scale. *
- Project was open-sourced in 2014.
- Google has since donated to the Cloud Native Computing Foundation (CNCF).

* Large-scale cluster management at Google with Borg

Design Principles

- Declarative > imperative
- Control loops
- Simple > complex
- Modularity
- Legacy compatible
- Network-centric
- Labels > hierarchy
- Cattle > pets
- Open > closed

Kubernetes Architecture

Architecture overview

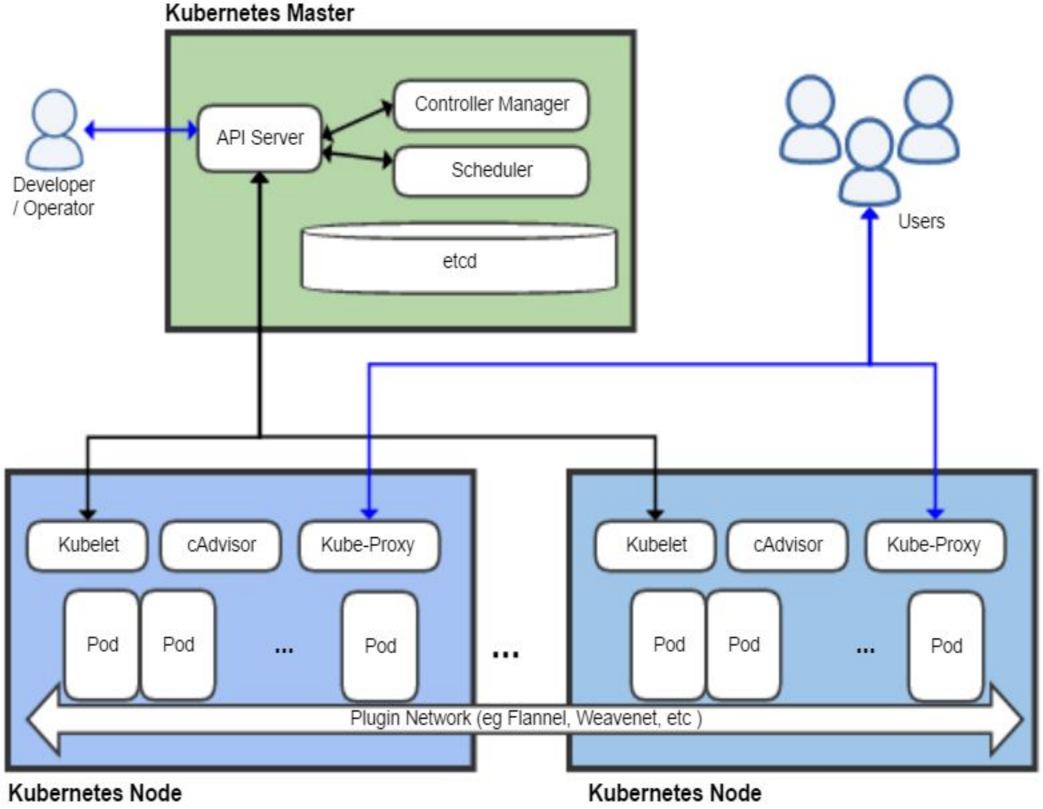
The machines that make up a Kubernetes cluster are called **nodes**.

Nodes in a Kubernetes cluster may be physical (bare metal), or virtual (vm's).

Two types of nodes:

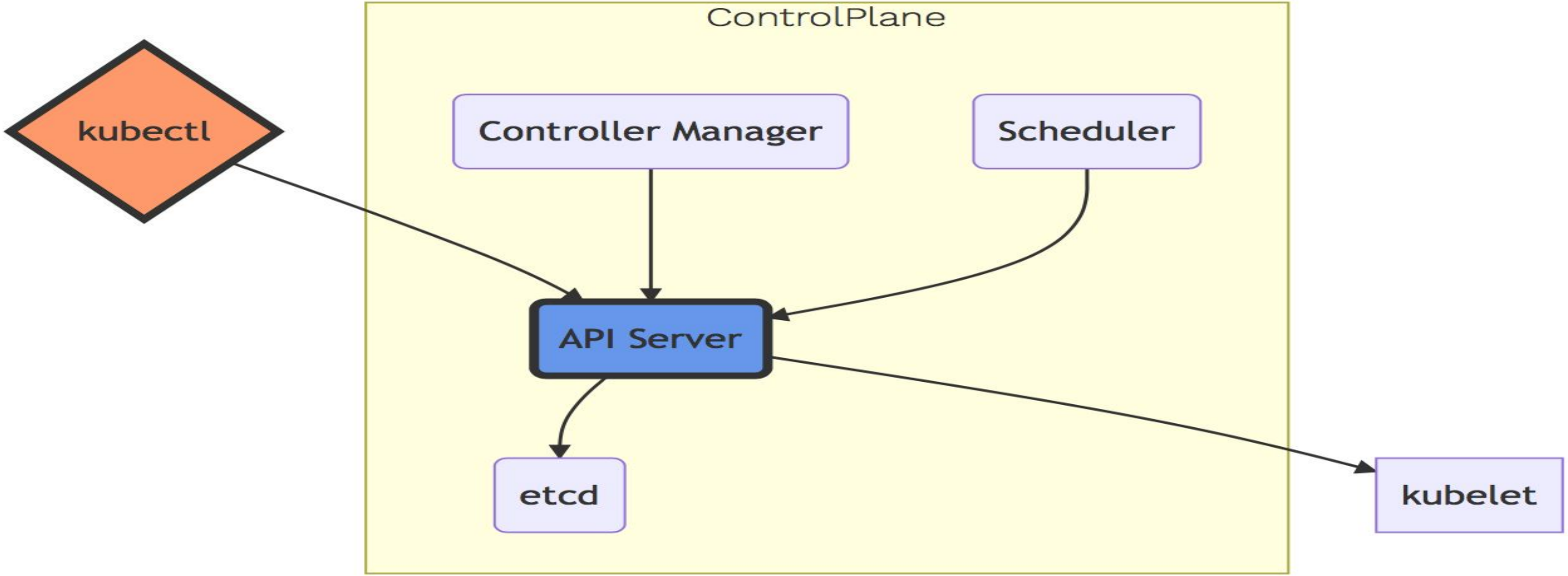
Masters, which make up the **Control Plane**, acts as the "brains" of the cluster.

Workers, which make up the **Data Plane**, runs the actual container images (via pods).



Master components (Control plane)

Master components



- One or more API Servers: Entry point for REST / kubectl
- etcd: Distributed key/value store
- Controller-manager: Always evaluating current vs desired state
- Scheduler: Schedules pods to worker nodes

kubernetes

API server

The kube API server exposes the Kubernetes REST API. It can easily scale horizontally as it is stateless and stores all data in the etcd cluster.

All clients, including nodes, users and other applications interact with Kubernetes strictly through the API Server.

etcd

Etcd is a highly reliable distributed datastore. Kubernetes uses it to store the entire cluster state.

In small, transient cluster a single instance of etcd can run on the same node with all the other master components.

However, for more substantial clusters it is typical to have a 3-node or even 5-node etcd cluster for redundancy and high availability.

Controller manager

The controller-manager is a collection of various managers rolled up into one binary. It is the primary daemon that manages all core component control loops.

All these managers monitor the cluster state via the API and steers the cluster towards the desired state.

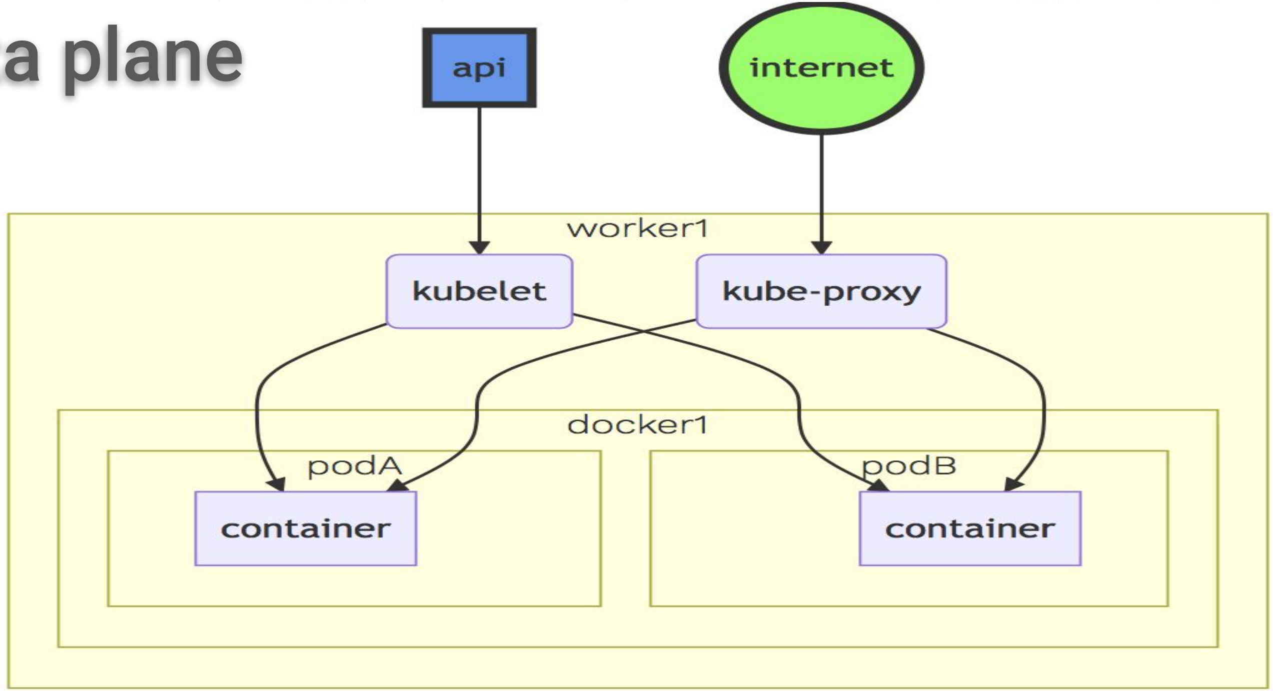
Scheduler

The kube-scheduler is responsible for evaluating the workload requirements and attempts to place it on a matching resource.

These requirements can include such things as general hardware reqs, affinity, anti-affinity, and other custom resource requirements.

Node components (Data plane)

Data plane



- Made up of worker nodes
- kubelet: Acts as a conduit between the API server and the node
- kube-proxy: Manages IP translation and routing

kubelet

Acts as the node agent responsible for managing pod lifecycle on its host.

It oversees the communication with the master components and manage the running pods.

- Download pod secrets from the API server
- Mount volumes
- Run the pod’s container
- Report the status of the node and each pod
- Run container liveness probes

kube-proxy

Manages the network rules on each node and performs connection forwarding or load balancing for Kubernetes cluster services.

Container runtime

A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.

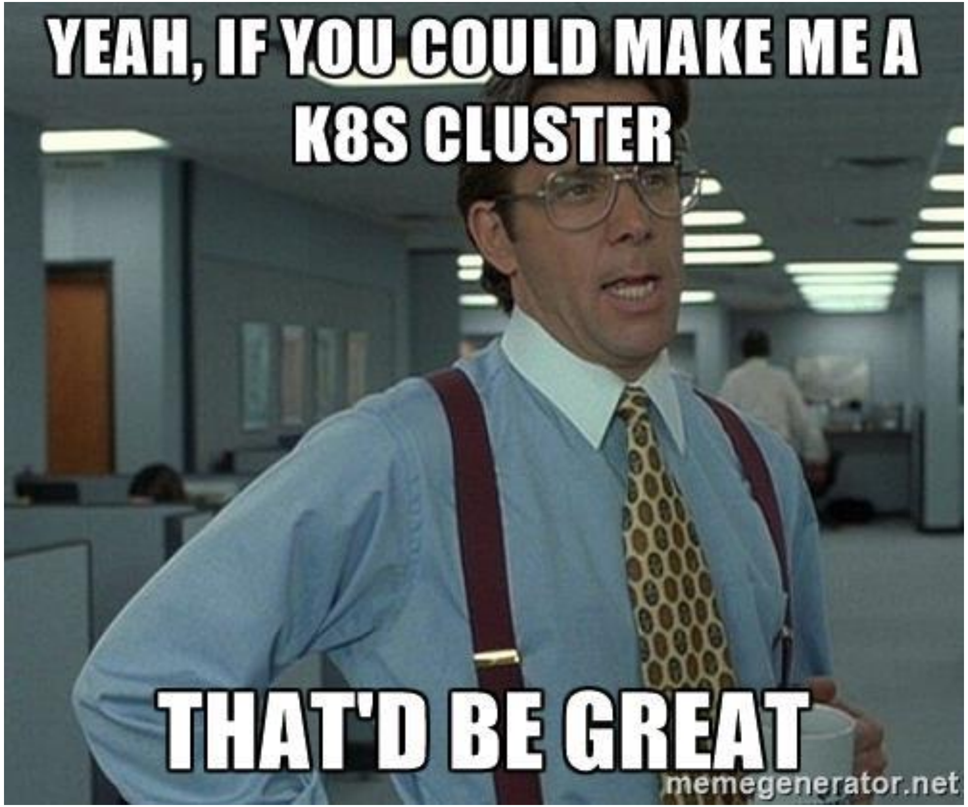
Kubernetes originally supported Docker as a container runtime engine. However, that is no longer the case;

- Containerd (docker)
- CRI-O
- Rkt
- etc

Cluster DNS

Provides Cluster Wide DNS for Kubernetes Services.

- kube-dns (default pre 1.11)
- CoreDNS (current default 1.13)



https://www.katacoda.com/smclernon/scenarios/kubernetes-101

Kubernetes objects

Namespaces

Namespaces are a logical cluster or environment, and are the primary method of partitioning a cluster or scoping access.

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: prod
  labels:
    app: CorpWebApp
```

```
$ kubectl get ns --show-labels
NAME           STATUS    AGE       LABELS
default        Active    11h       <none>
kube-public    Active    11h       <none>
kube-system    Active    11h       <none>
prod           Active    6s        app=CorpWebApp
```

Default namespaces

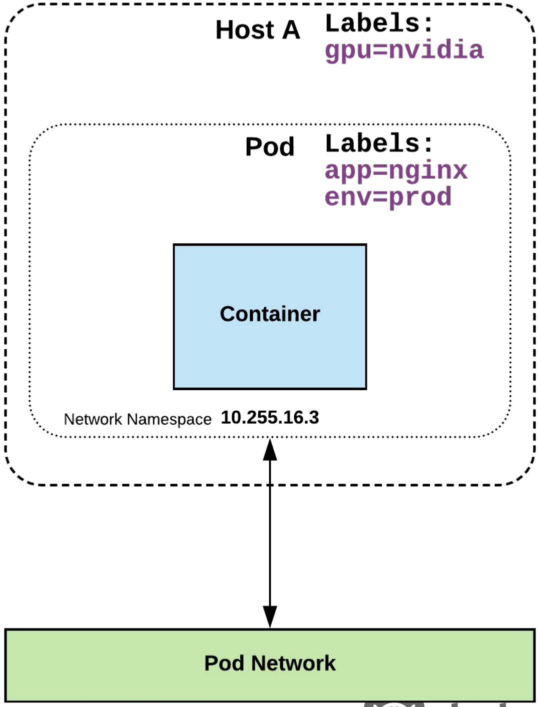
- **default**: The default namespace for any object without a namespace.
- **kube-system**: Acts as the home for objects and resources created by Kubernetes itself.
- **kube-public**: A special namespace; readable by all users that is reserved for cluster bootstrapping and configuration.

```
$ kubectl get ns --show-labels
NAME            STATUS    AGE        LABELS
default         Active    11h        <none>
kube-public     Active    11h        <none>
kube-system     Active    11h        <none>
```

kubernetes

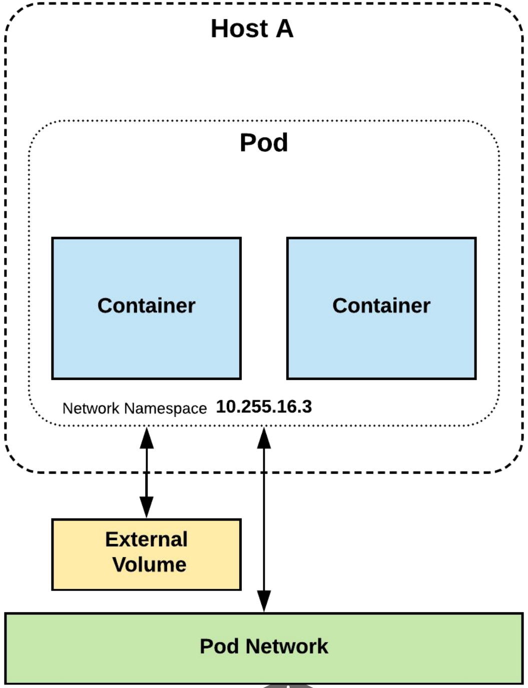
Labels

- key-value pairs that are used to identify, describe and group together related sets of objects or resources.
- **NOT** characteristic of uniqueness.
- Have a strict syntax with a slightly limited character set.



Pods

- **Atomic unit** or smallest "*unit of work*"of Kubernetes.
- Foundational building block of Kubernetes Workloads.
- Pods are one or more containers that share volumes, a network namespace, and are a part of a **single context**.



kubernetes

Pods are mortal

- They are born and when they die, they are not resurrected.
- ReplicaSets in particular create and destroy Pods dynamically (e.g. when scaling out or in).
- While each Pod gets its own IP address, even those IP addresses cannot be relied upon to be stable over time.

Services

- **Unified method of accessing** the exposed workloads of Pods.
- **Durable resource** (unlike Pods)
  - static cluster-unique IP
  - static namespaced DNS name

```
<service name>.<namespace>.svc.cluster.local
```

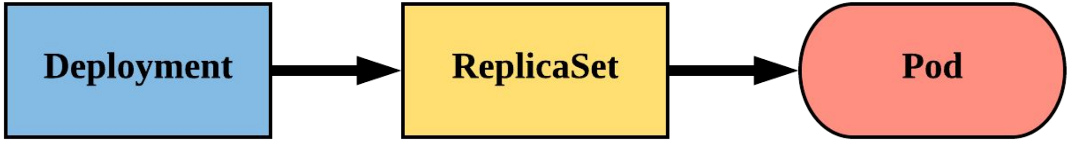
kubernetes

Services (contd)

- Target Pods using **equality based selectors**.
- Uses **kube-proxy** to provide simple load-balancing.
- **kube-proxy** acts as a daemon that creates **local entries** in the host’s iptables for every service.

Deployments

- Declarative method of managing Pods via **ReplicaSets.**
- Provide rollback functionality and update control.
- Updates are managed through the **pod-template-hash** label.
- Each iteration creates a unique label that is assigned to both the **ReplicaSet** and subsequent Pods.



kubernetes

Deployment

- revisionHistoryLimit: The number of previous iterations of the Deployment to retain.
- strategy: Describes the method of updating the Pods based on the type. Valid options are Recreate or RollingUpdate.
    - Recreate: All existing Pods are killed before the new ones are created.
    - RollingUpdate: Cycles through updating the Pods according to the parameters: maxSurge and maxUnavailable.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-example
spec:
  replicas: 3
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    <pod template>
```

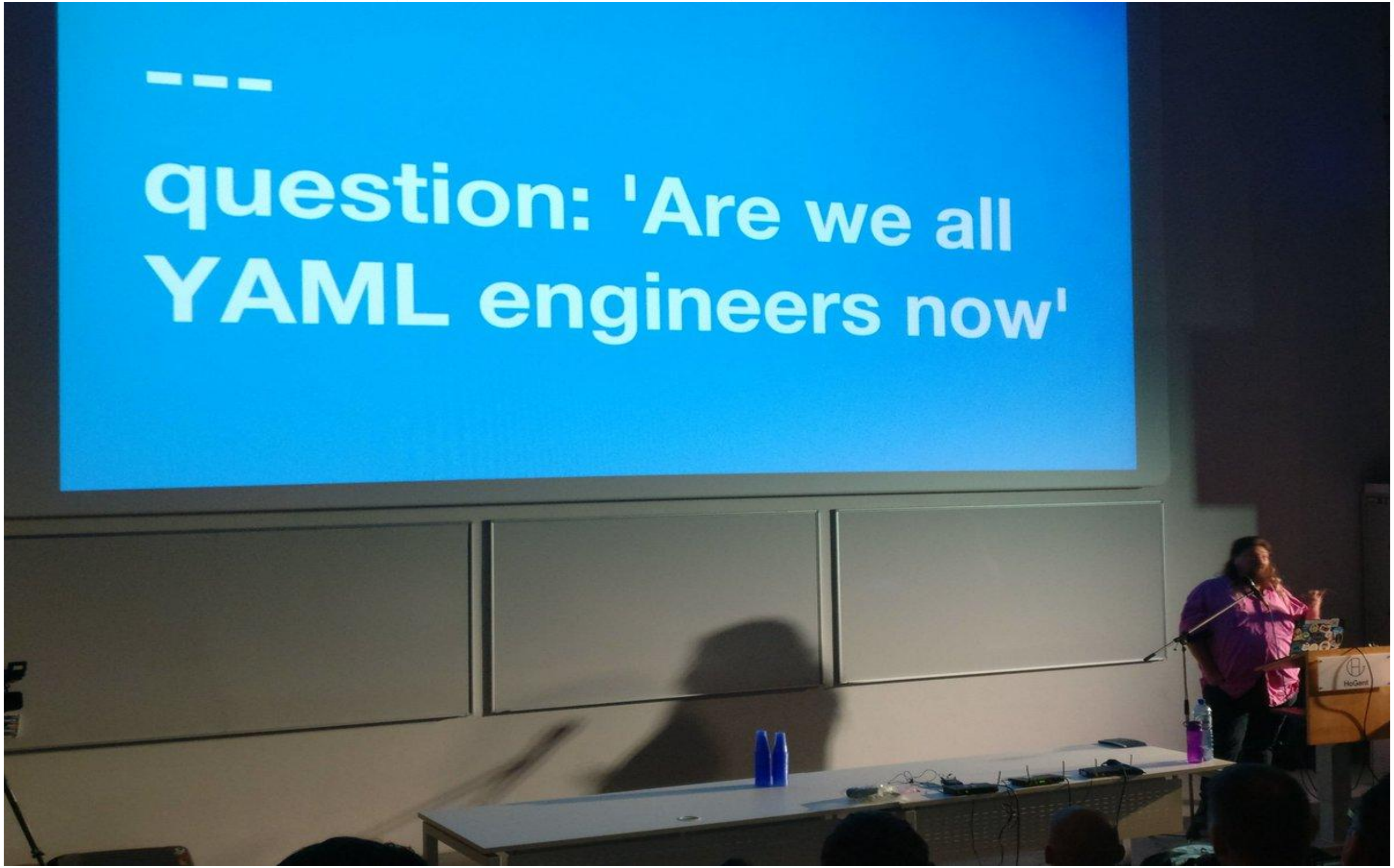
ENOUGH TALK
SHOW ME A DEMO!!!
memegenerator.net

And more

- Ingress
  - L7 load balancing
- Jobs
  - Run to completion
- Autoscaling
  - Automatically adjust replica count
- DaemonSets
  - Run something on every node (or subset)
- StatefulSet
  - Support for long term stateful distributed systems

So much more

- Role Based Access Control (RBAC)
    - Control what users have access to what objects
- Multiple Schedulers
- Flexible Scheduling Constraints
    - Affinity, anti-affinity, taints, tolerations
- Automatic Cluster Scaling
    - K8s publishes signals that allow external services to scale the cluster automatically.
- Cloud Provider Integration
    - GCP, AWS, Azure, OpenStack, vSphere
- Network Policy
    - Network ingress policy



```
---
question: 'Are we all
YAML engineers now'
```

Gotchas

If you are managing your own cluster, then it is not without its perils.

Even with all the features that Kubernetes provides, it still requires lots of monitoring and pre-emptive capacity planning.

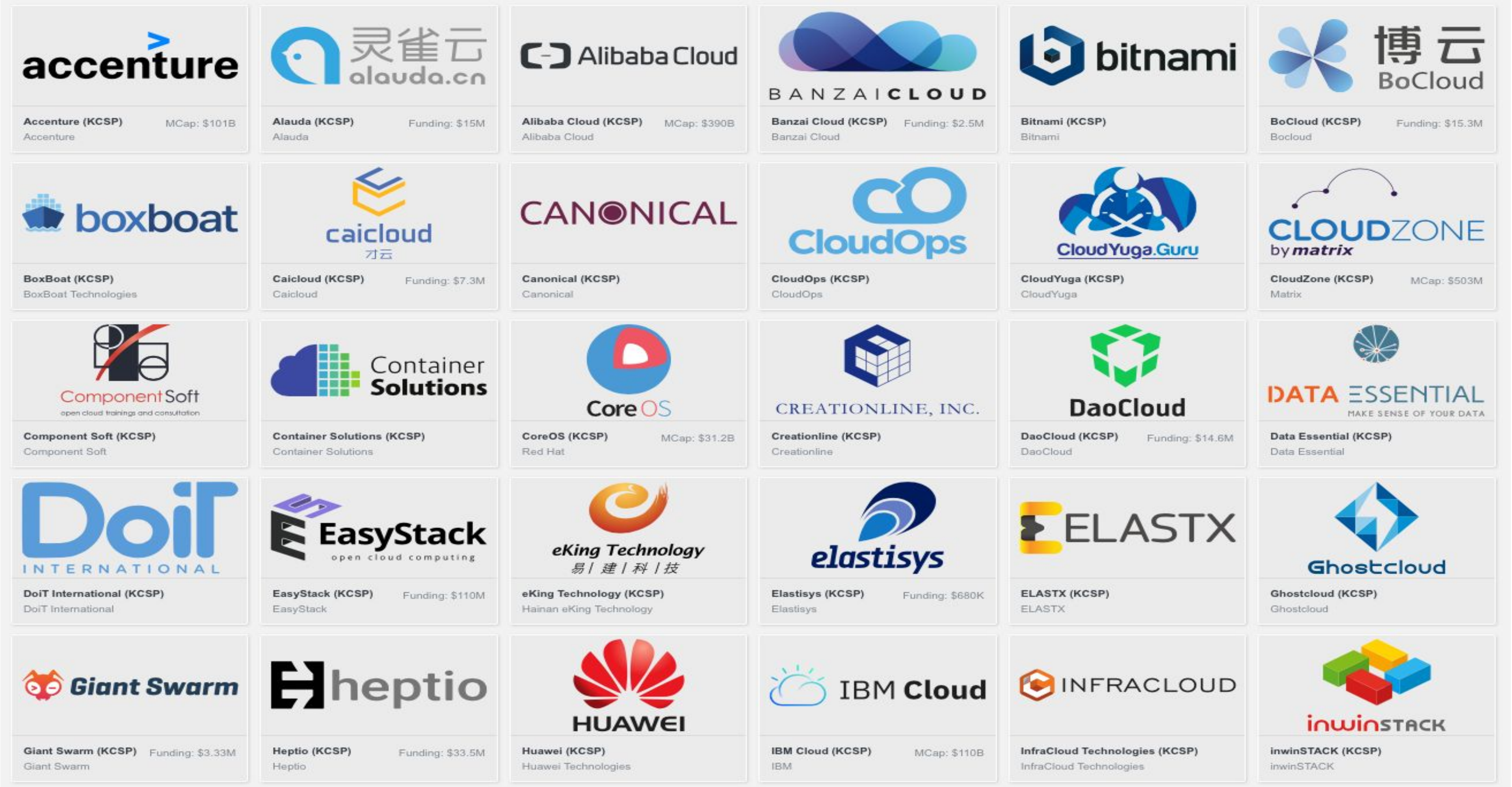
Unbalanced clusters

Unmonitored clusters will go on fire (not literally)

Without autoscaling in place, clusters can crash

Kubernetes Certified Service Provider

At time of writing there are 60+ certified service providers - see full list at https://www.cncf.io/certification/kcsp/



Accenture (KCSP) MCap: $101B
Accenture

Alauda (KCSP) Funding: $15M
Alauda

Alibaba Cloud (KCSP) MCap: $390B
Alibaba Cloud

Banzai Cloud (KCSP) Funding: $2.5M
Banzai Cloud

Bitnami (KCSP)
Bitnami

BoCloud (KCSP) Funding: $15.3M
Bocloud

BoxBoat (KCSP)
BoxBoat Technologies

Caicloud (KCSP) Funding: $7.3M
Caicloud

Canonical (KCSP)
Canonical

CloudOps (KCSP)
CloudOps

CloudYuga (KCSP)
CloudYuga

CloudZone (KCSP) MCap: $503M
Matrix

Component Soft (KCSP)
Component Soft

Container Solutions (KCSP)
Container Solutions

CoreOS (KCSP) MCap: $31.2B
Red Hat

Creationline (KCSP)
Creationline

DaoCloud (KCSP) Funding: $14.6M
DaoCloud

Data Essential (KCSP)
Data Essential

DoiT International (KCSP)
DoiT International

EasyStack (KCSP) Funding: $110M
EasyStack

eKing Technology (KCSP)
Hainan eKing Technology

Elastisys (KCSP) Funding: $680K
Elastisys

ELASTX (KCSP)
ELASTX

Ghostcloud (KCSP)
Ghostcloud

Giant Swarm (KCSP) Funding: $3.33M
Giant Swarm

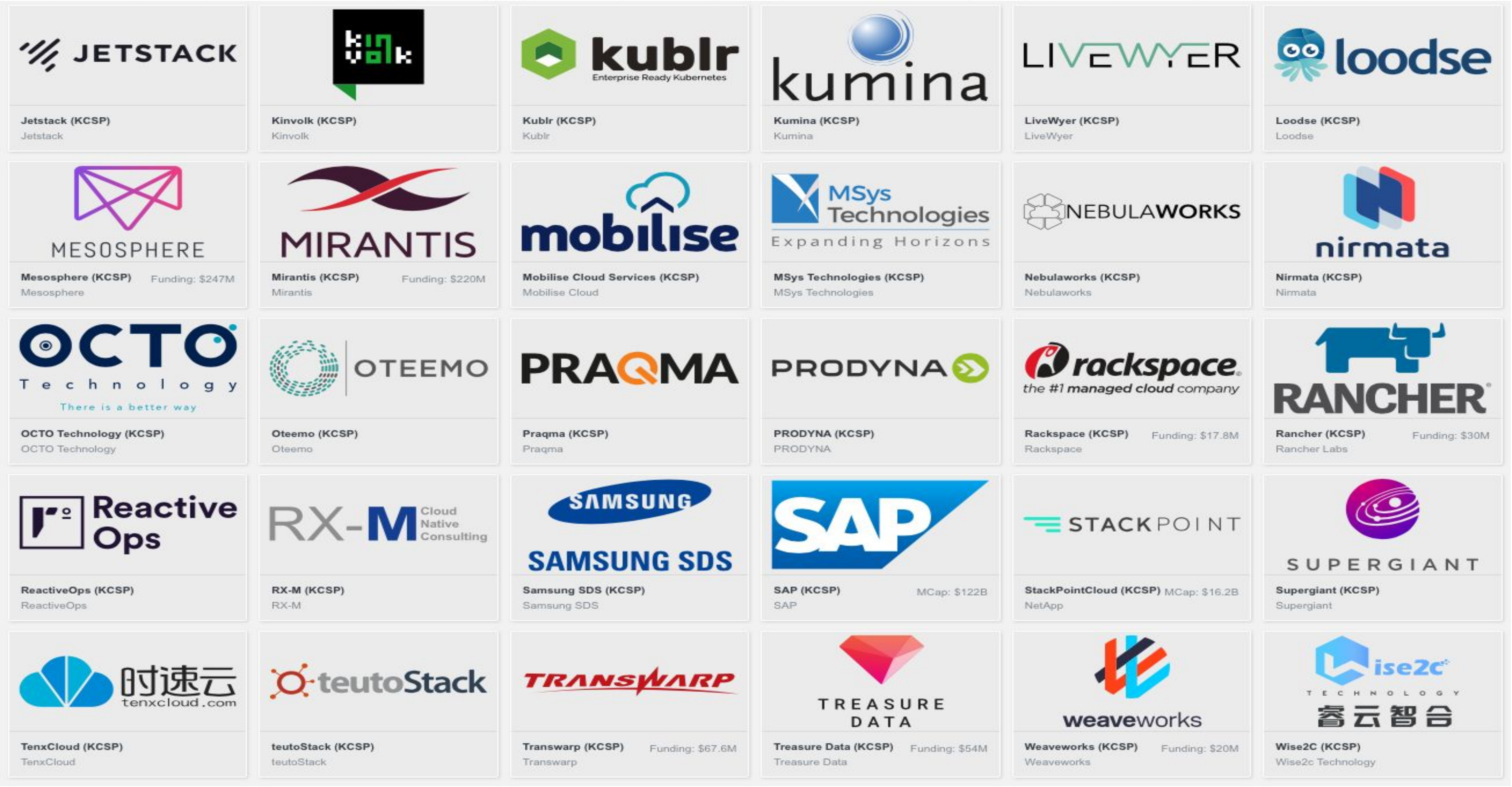
Heptio (KCSP) Funding: $33.5M
Heptio

Huawei (KCSP)
Huawei Technologies

IBM Cloud (KCSP) MCap: $110B
IBM

InfraCloud Technologies (KCSP)
InfraCloud Technologies

inwinSTACK (KCSP)
inwinSTACK



Jetstack (KCSP)
Jetstack

Kinvolk (KCSP)
Kinvolk

Kublr (KCSP)
Kublr

Kumina (KCSP)
Kumina

LiveWyer (KCSP)
LiveWyer

Loodse (KCSP)
Loodse

Mesosphere (KCSP) Funding: $247M
Mesosphere

Mirantis (KCSP) Funding: $220M
Mirantis

Mobilise Cloud Services (KCSP)
Mobilise Cloud

MSys Technologies (KCSP)
MSys Technologies

Nebulaworks (KCSP)
Nebulaworks

Nirmata (KCSP)
Nirmata

OCTO Technology (KCSP)
OCTO Technology

Oteemo (KCSP)
Oteemo

Praqma (KCSP)
Praqma

PRODYNA (KCSP)
PRODYNA

Rackspace (KCSP) Funding: $17.8M
Rackspace

Rancher (KCSP) Funding: $30M
Rancher Labs

ReactiveOps (KCSP)
ReactiveOps

RX-M (KCSP)
RX-M

Samsung SDS (KCSP)
Samsung SDS

SAP (KCSP) MCap: $122B
SAP

StackPointCloud (KCSP) MCap: $16.2B
NetApp

Supergiant (KCSP)
Supergiant

TenxCloud (KCSP)
TenxCloud

teutoStack (KCSP)
teutoStack

Transwarp (KCSP) Funding: $67.6M
Transwarp

Treasure Data (KCSP) Funding: $54M
Treasure Data

Weaveworks (KCSP) Funding: $20M
Weaveworks

Wise2C (KCSP)
Wise2c Technology

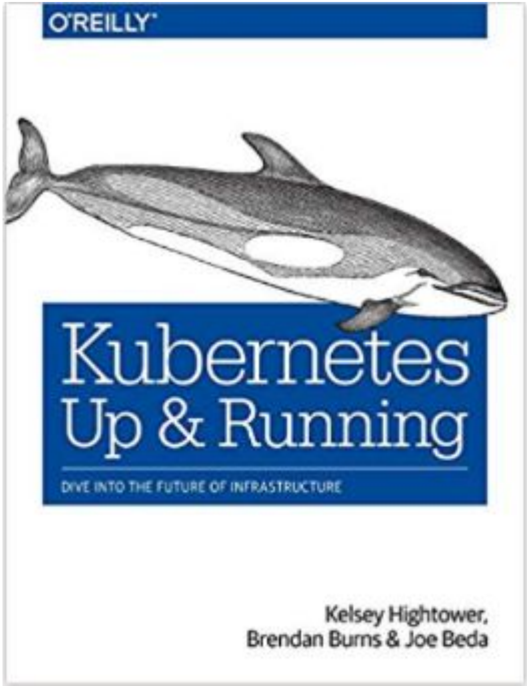
Additional resources

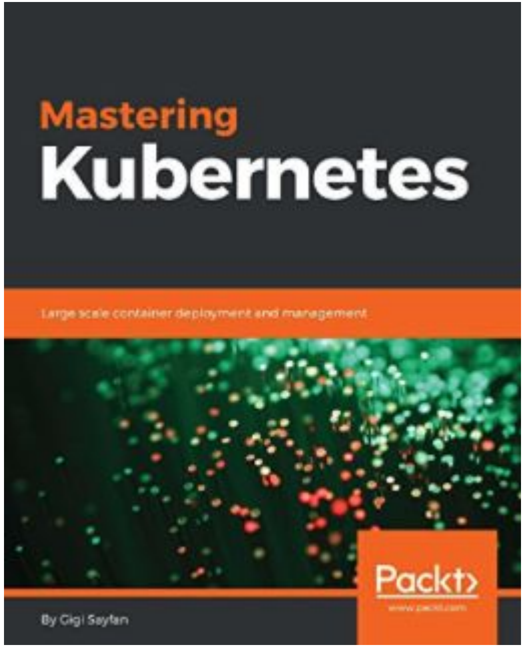
kubernetes

Links

- **Kubernetes documentation**
  https://kubernetes.io/docs/home/
- **Interactive Kubernetes Tutorials**
  https://www.katacoda.com/courses/kubernetes
- **Learn Kubernetes the Hard Way by Kelsey Hightower**
  https://github.com/kelseyhightower/kubernetes-the-hard-way
- **Awesome Kubernetes**
  https://github.com/ramitsurana/awesome-kubernetes

Some useful books...





Kubernetes cluster setup

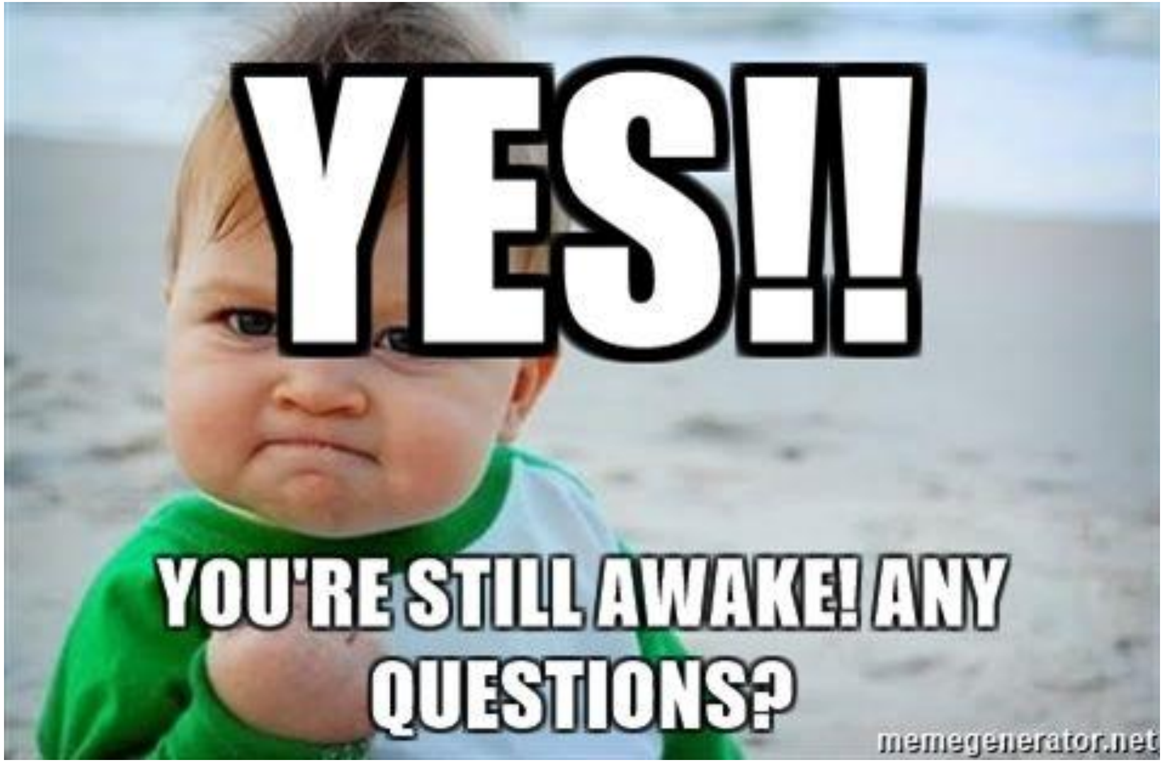
There are many tools available to help bootstrap and configure a self-managed/commercial Kubernetes cluster.

Local development/learning;

- Minikube
- Docker for Mac
- Docker for Windows
- Kubeadm
- katacoda.com/learn

Commercial offerings;

- GKE - https://cloud.google.com/kubernetes-engine/
- EKS - https://aws.amazon.com/eks/
- AKS - https://azure.microsoft.com/en-gb/services/kubernetes-service/
- IKS - https://www.ibm.com/cloud/container-service
- PKS - https://pivotal.io/platform/pivotal-container-service
- Kops


YES!!
YOU'RE STILL AWAKE! ANY QUESTIONS?
memegenerator.net