CLOUD INFRASTRUCTURE WITH KUBERNETES

WHO, HOW, WHY AND WHEN?

Ben Avellone // CEO // Coastware Technologies

COASTWARE TECHNOLOGIES

- Founded in 2016, business-class software applications and comprehensive IT services
- Adopted Kubenetes in early 2017, now managing all of Coastware's web services
- Limited time & resources to spend on infrastructure

WHAT IS KUBERNETES?

"Pilot/Helmsman" in Greek

Initially built inside Google based on **Borg and Omega** - Google's internal systems that power search, video and advertising services

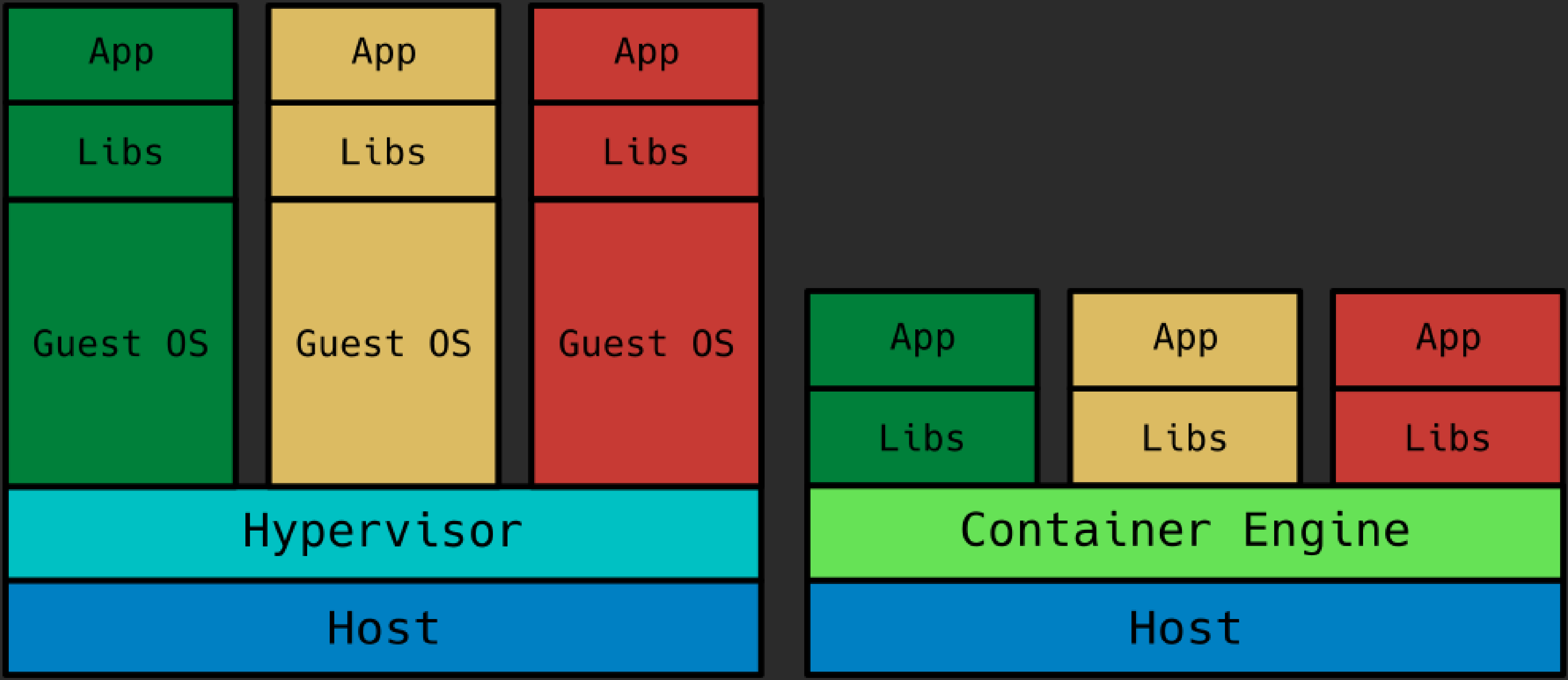
Set of **tools, concepts and APIs** designed to deploy, scale and maintain containers running on a cluster of machines

BUILDING BLOCKS: CONTAINERS

- Contains all libraries and code needed by application
- Less overhead and lower start-up time than VMs
- Easy to compose and replace

VMS

CONTAINERS



App | App | App

Libs | Libs | Libs

Guest OS | Guest OS | Guest OS

Hypervisor

Host

App | App | App

Libs | Libs | Libs

Container Engine

Host

PROBLEMS WHERE KUBERNETES CAN HELP

Developer / Programmer

- Applications and machines crash; the more you have the more often it will happen
- Distributed systems are powerful and can solve many problems but are difficult to manage
- Existing infrastructure is limited in features or guarantees

PROBLEMS WHERE KUBERNETES CAN HELP

Ops / System Administrator

- Scaling infrastructure can be a slow, error-prone and manual process
- Staff need to be on-call 24/7 to respond to inevitable crisis at 3am
- Existing infrastructure is brittle and cumbersome

PROBLEMS WHERE KUBERNETES CAN HELP

Manager / Team Lead / Executive

- Hiring and training staff on homegrown infrastructure is difficult and expensive
- More engineers and technicians are needed to manage ever-growing infrastructure
- Brittle infrastructure increases risk of mistakes, errors and oversights

WHY KUBERNETES?

WHY KUBERNETES?

Strong community: 1800+ contributors, used by F500 companies and startups alike

WHY KUBERNETES?

Strong community: 1800+ contributors, used by F500 companies and startups alike

Runs everywhere: Public cloud, private cloud, bare metal, laptop

WHY KUBERNETES?

Strong community: 1800+ contributors, used by F500 companies and startups alike

Runs everywhere: Public cloud, private cloud, bare metal, laptop

Batteries included: Lots of functionality out-of-the-box

BUILDING BLOCKS: DEPLOYMENT

Our application is packaged up into a container
How do we run, update and scale it?

BUILDING BLOCKS: DEPLOYMENT

Our application is packaged up into a container
How do we run, update and scale it?

- Manual - Humans

BUILDING BLOCKS: DEPLOYMENT

Our application is packaged up into a container
How do we run, update and scale it?

- Manual - Humans
- Automated - Scripts

BUILDING BLOCKS: DEPLOYMENT

Our application is packaged up into a container
How do we run, update and scale it?

- Manual - Humans
- Automated - Scripts
- Automated - Orchestrator

CONTAINER MANAGMENT:

HUMANS

Pros: simple - minimal tooling, configuration and setup required

Cons: not automated, error-prone, doesn't scale, problems require manual troubleshooting

CONTAINER MANAGMENT: SCRIPTS

Pros: integrates with existing infrastructure, reproducible, auditable

Cons: manual placement of containers on machines, becomes unmanageable for larger systems

CONTAINER MANAGMENT:
ORCHESTRATION

Pros: automated, self-healing, scalable, portable

Cons: some overhead, learning curve, new tooling

BUILDING BLOCKS: ORCHESTRATORS

BUILDING BLOCKS: ORCHESTRATORS

Scheduling: match containers to machines

- by resource needs (CPU, Memory)
- by affinity requirements (put X near Y)
- by labels (put X on a "test" machine)

BUILDING BLOCKS: ORCHESTRATORS

Scheduling: match containers to machines

- by resource needs (CPU, Memory)
- by affinity requirements (put X near Y)
- by labels (put X on a "test" machine)

Replication: run N copies

BUILDING BLOCKS: ORCHESTRATORS

Scheduling: match containers to machines

- by resource needs (CPU, Memory)
- by affinity requirements (put X near Y)
- by labels (put X on a "test" machine)

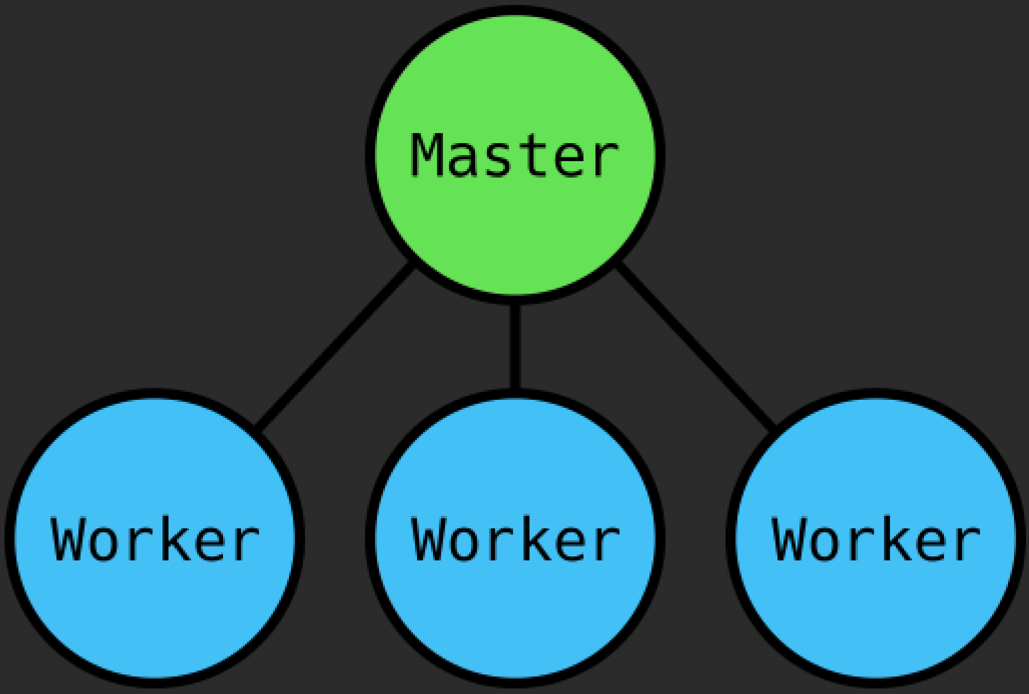
Replication: run N copies

Recovery: handle machine failures

KUBERNETES ARCHITECTURE

NODES

Machine running Kubernetes



CLUSTER

Group of cooperating nodes

PODS

Group of cooperating containers

```
kind: Pod
metadata:
  name: my-website-1
  labels:
    app: my-website
spec:
  containers:
  - name: webserver
    image: my-website-v1.0
    ports:
    - containerPort: 80
```

DEPLOYMENT

Manages the lifecycle of a group of pods

```yaml
kind: Deployment
metadata:
  name: my-website
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-website
  template:
    metadata:
      labels:
        app: my-website
    spec:
      containers:
      - name: webserver
        image: my-website-v1.0
        ports:
        - containerPort: 80
```

SERVICE

A service forwards traffic to a group of pods

```
kind: Service
metadata:
  name: my-website
spec:
  selector:
    app: my-website
  ports:
  - name: http
    port: 8000
    targetPort: 80
```