Kubernetes 101

Andres Guisado and Taliesin Sisson

CONTINO

Whoami

Taliesin Sisson

- talieson.sisson@contino.io
- http://github.com/taliesins

Andres Guisado

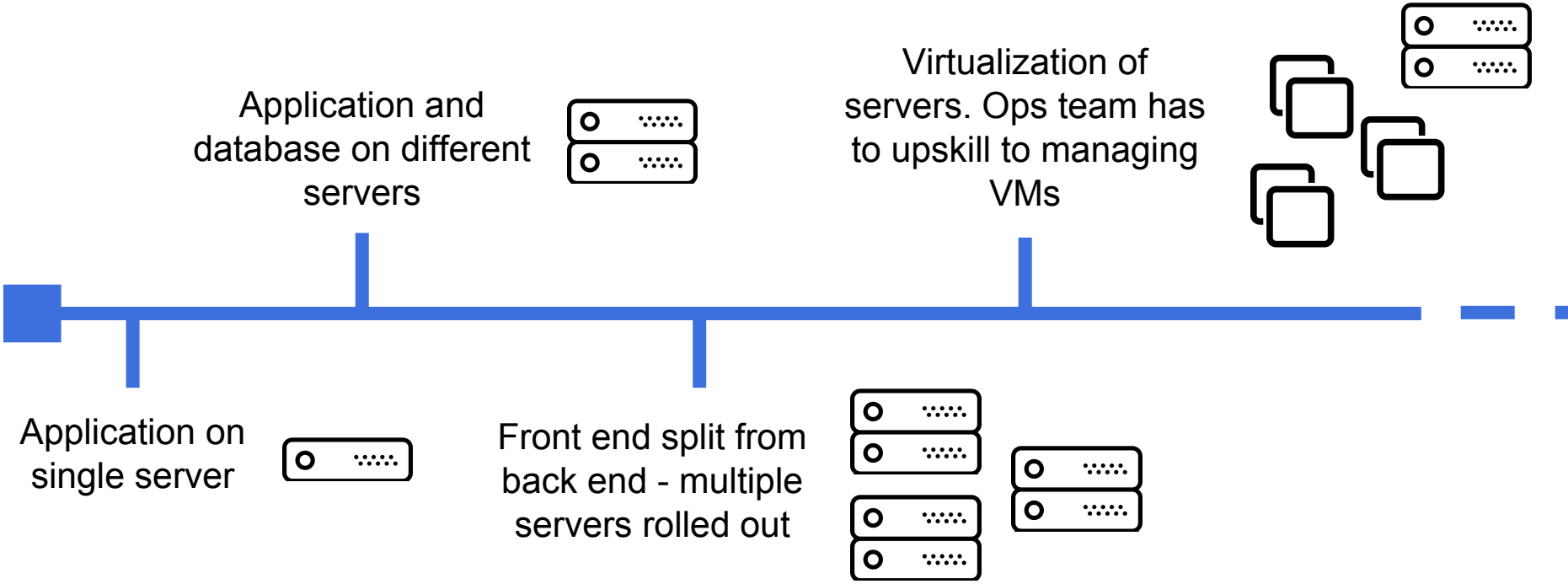
- andres.guisado@contino.io
- CKA
- @andresguisado

Agenda

1. Evolution to container orchestrator (5 min)
2. Kubernetes? (3 min)
3. Kubernetes architecture & concepts (20 min)
4. Hello World Kubernetes - demo (10 min)
5. What’s next? (2 min)
6. Q&A (15 min)

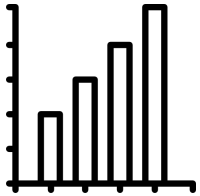
Evolution to container orchestrator

CONTINO

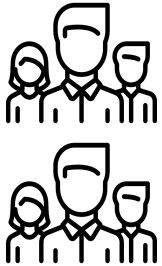

Application and database on different servers
Virtualization of servers. Ops team has to upskill to managing VMs
Application on single server
Front end split from back end - multiple servers rolled out

Health checks, smoke tests and centralized logging added to apps

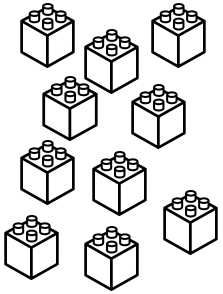
KPI/metrics and distributed transaction visualization



New dev team added to increase rollout pace



Explosion in microservices as each dev team responsible for multiple microservices



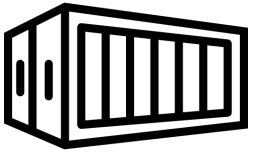
Service discovery reduces most of the configuration and some infrastructure overhead

Running in the cloud. Dev and Ops are pretty much one team now

Lets run container orchestrator/CaaS

Lets move to cloud

Applications that can't be replaced by SaaS get moved into containers



Kubernetes?

CONTINO

What is Kubernetes?

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

Goal

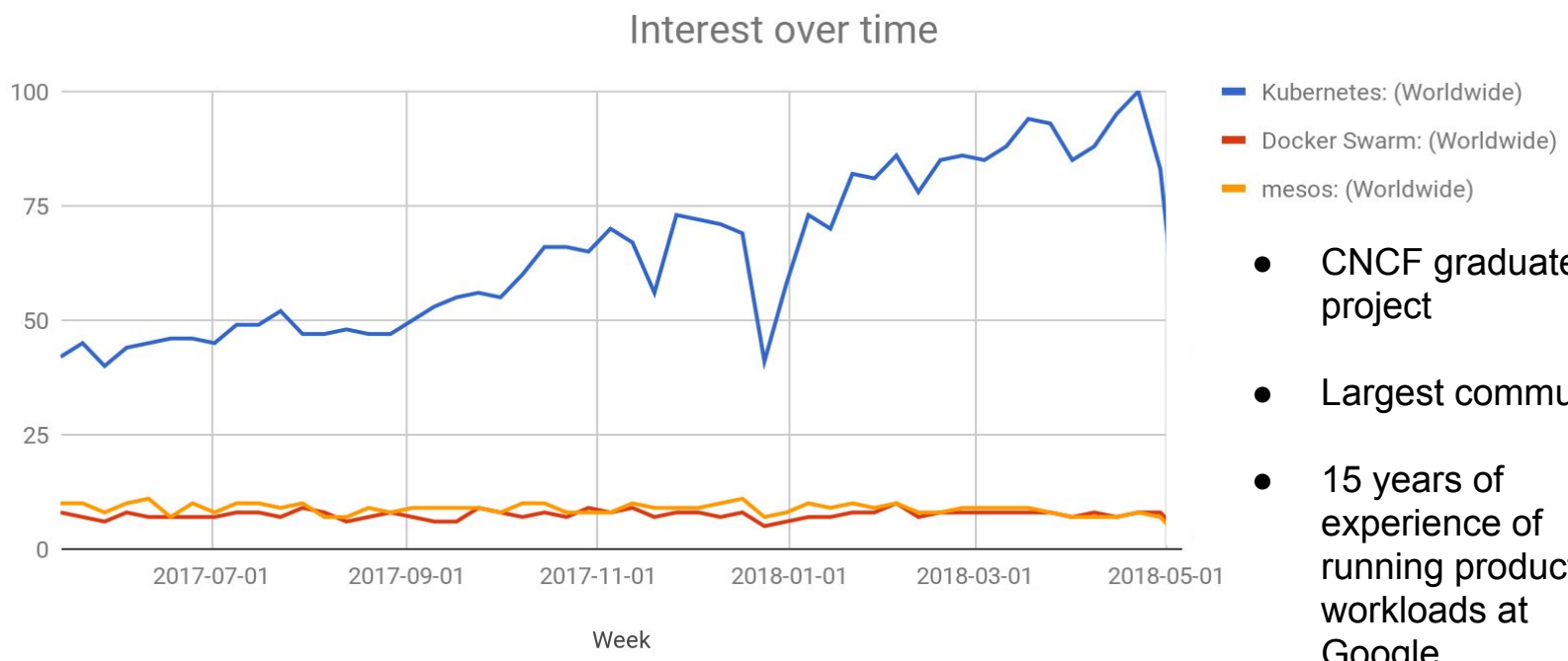
*"Kubernetes was built to radically change the way that applications are built and deployed in the cloud. Fundamentally, it was designed to give **developers** more **velocity**, **efficiency**, and **agility**"*

Kelsey Hightower, Brendan Burns & Joe Beda
-Kubernetes Up and Running Book

History

- Kubernetes is heavily influenced by Google's Borg system
- Original Google code name was Project Seven for Star Trek
- Released in 2015 when Google partnered with Linux foundation to form CNCF
- Often called K8s which is a Numeronym
  - K[ubernete]s → K[8]s → K8s (pronounced "Kates")
- Kubernetes - Greek for helmsman or pilot

Why pick Kubernetes



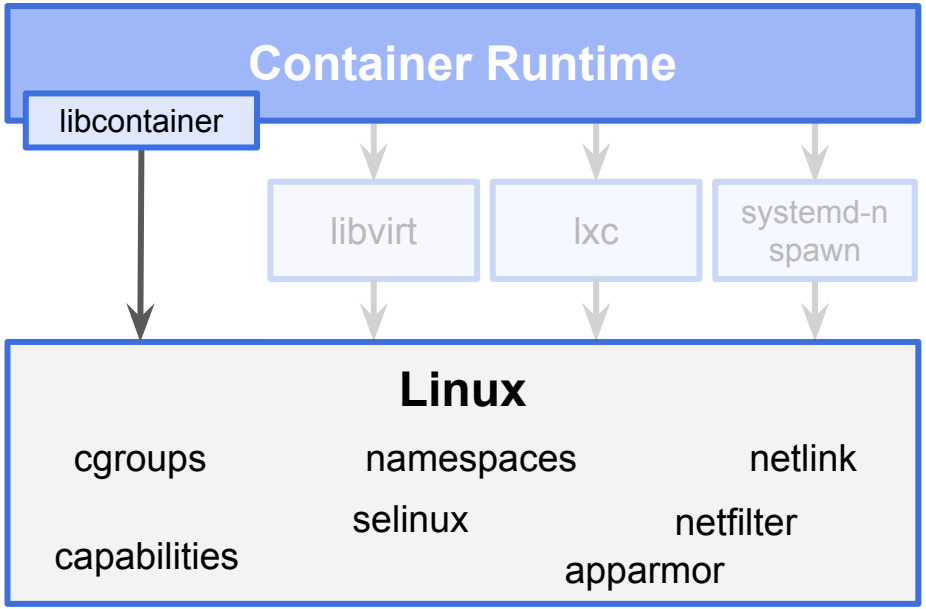
- CNCF graduated project
- Largest community
- 15 years of experience of running production workloads at Google

Kubernetes architecture & concepts

CONTINO

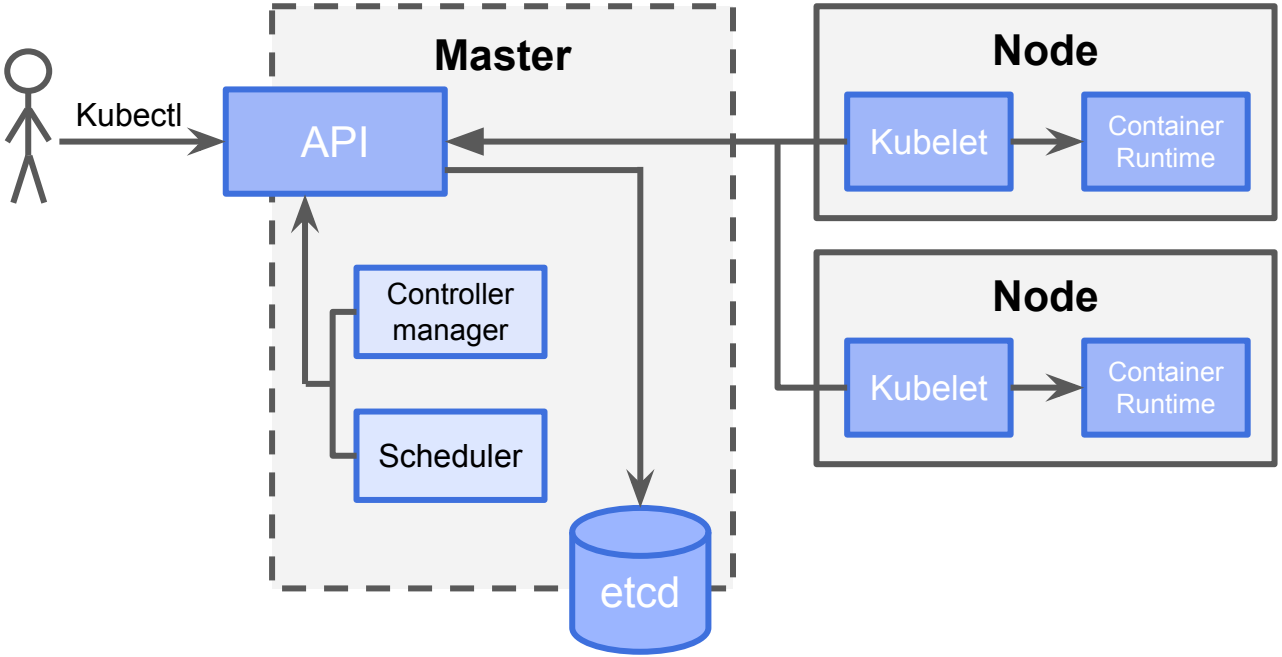
Container Runtime

- Container Runtime
  - Libcontainer is **runc**
- Linux Kernel features
  - Capabilities
  - Cgroups
  - Namespaces



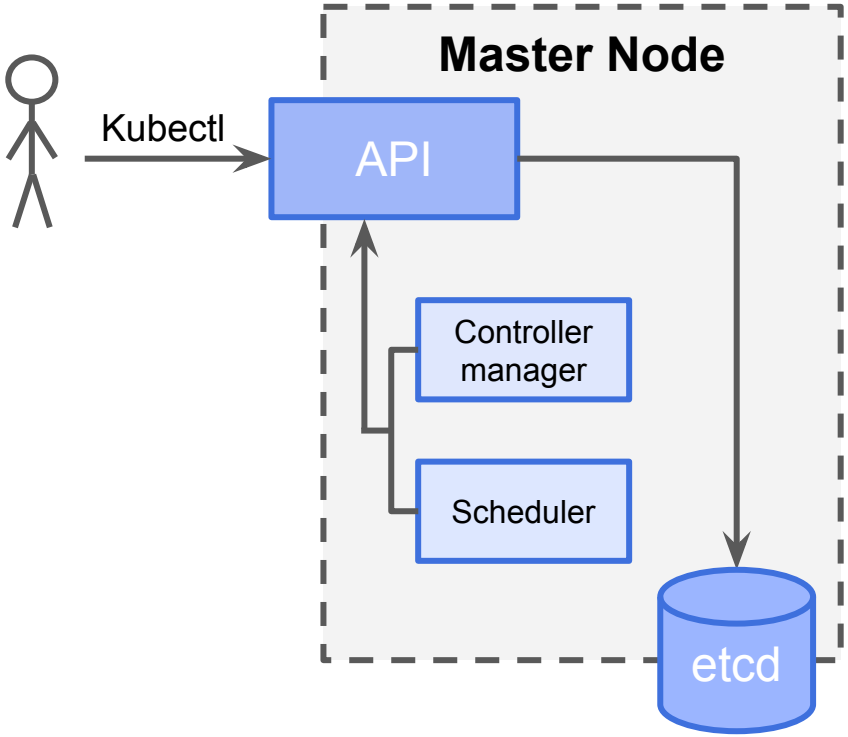
30,000ft view

- User:
  - Kubectl (CLI tool)
  - UI: Dashboard
- Master:
  - API Server
  - Etcd
  - Scheduler
  - Controller Manager
- Nodes:
  - Kubelet
  - Container Runtime



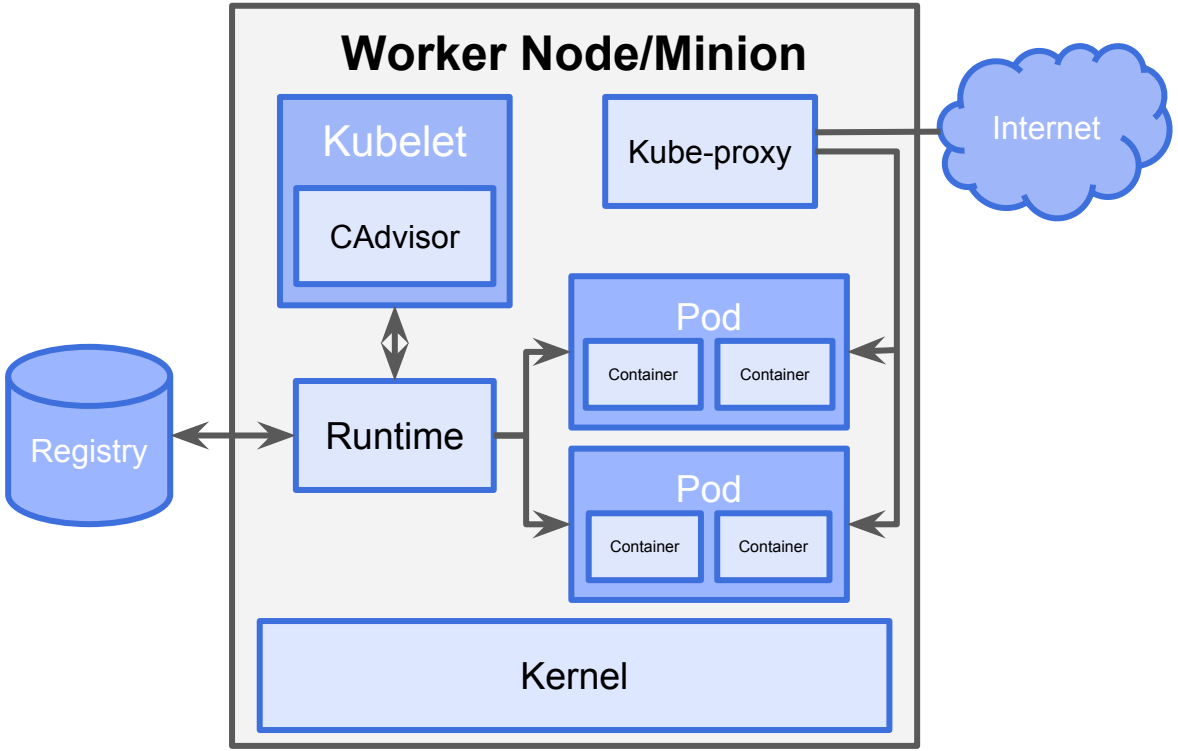
Master

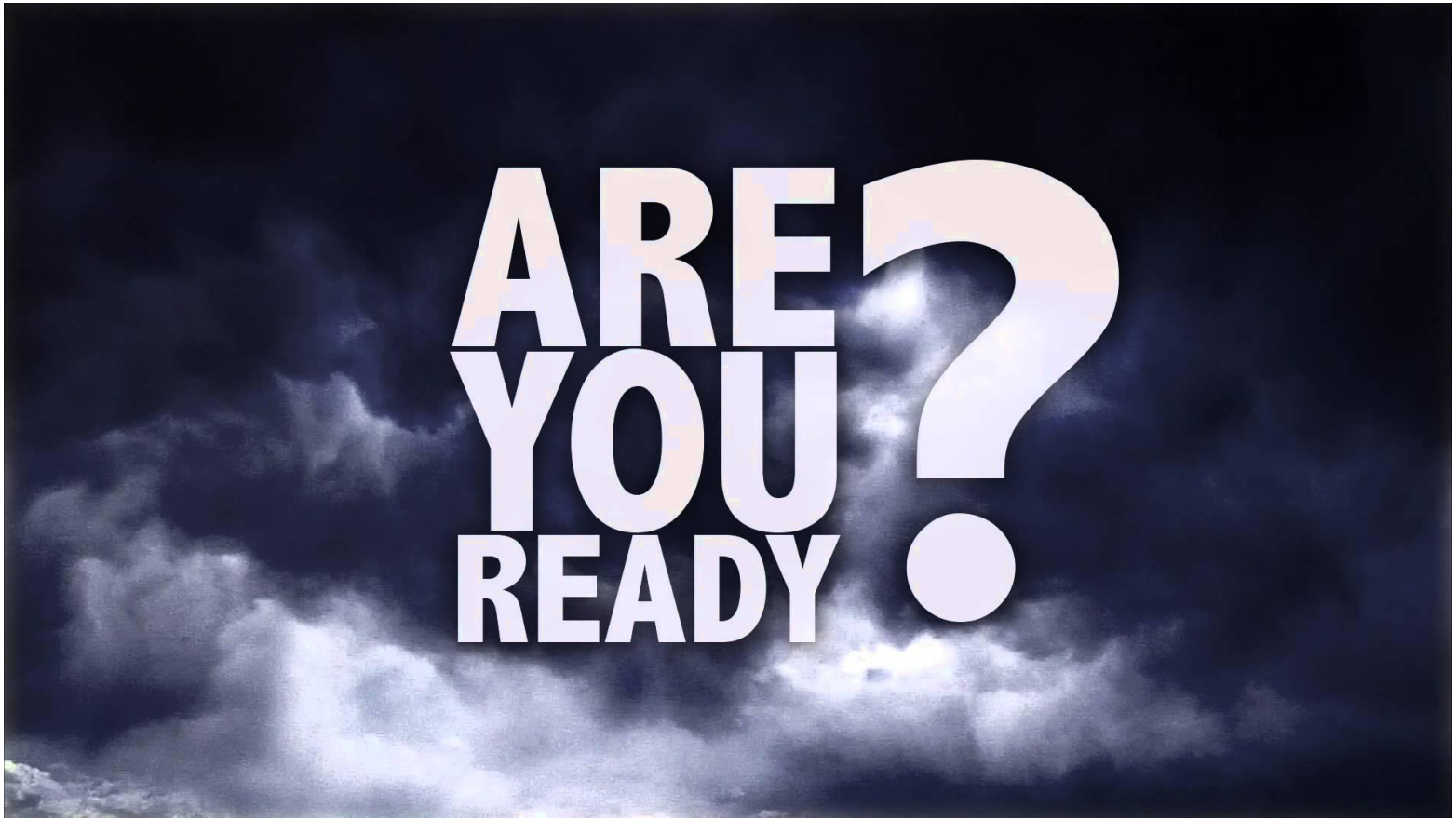
- API data store: Etcd (Cluster State)
- Controller Manager :
  - Node Controller
  - Deployment Controller
  - ReplicaSet Controller
  - Replication Controller
  - Endpoints Controller
  - Service Account & Token Controller
  - ..
- Scheduler - Bind pod to Node
- Well documented API
  https://kubernetes.io/docs/reference/
- Uses OpenAPI and Swagger



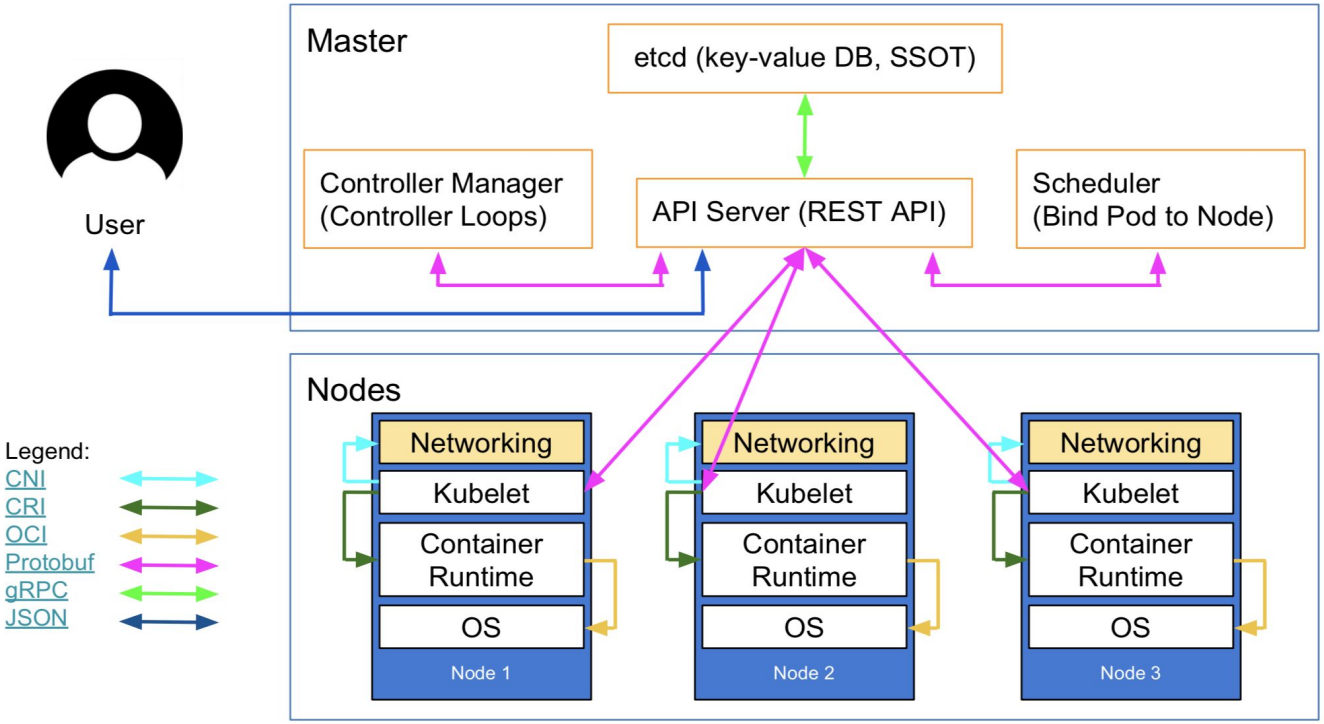
Nodes

- Kubelet:
  - cAdvisor (metrics, logs...)
- Container Runtime:
- Pod
  - Container (one or more)
- Kube-proxy:
  - Used to reach services and allow communication between Nodes.



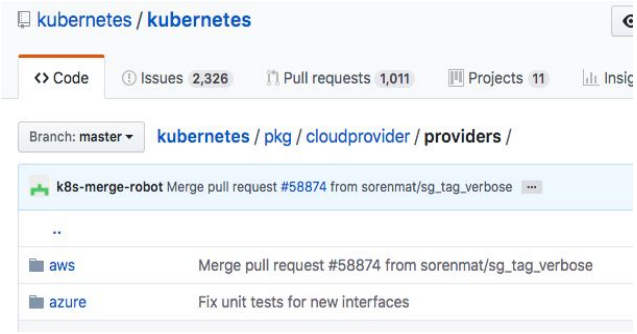

ARE
YOU
READY
?

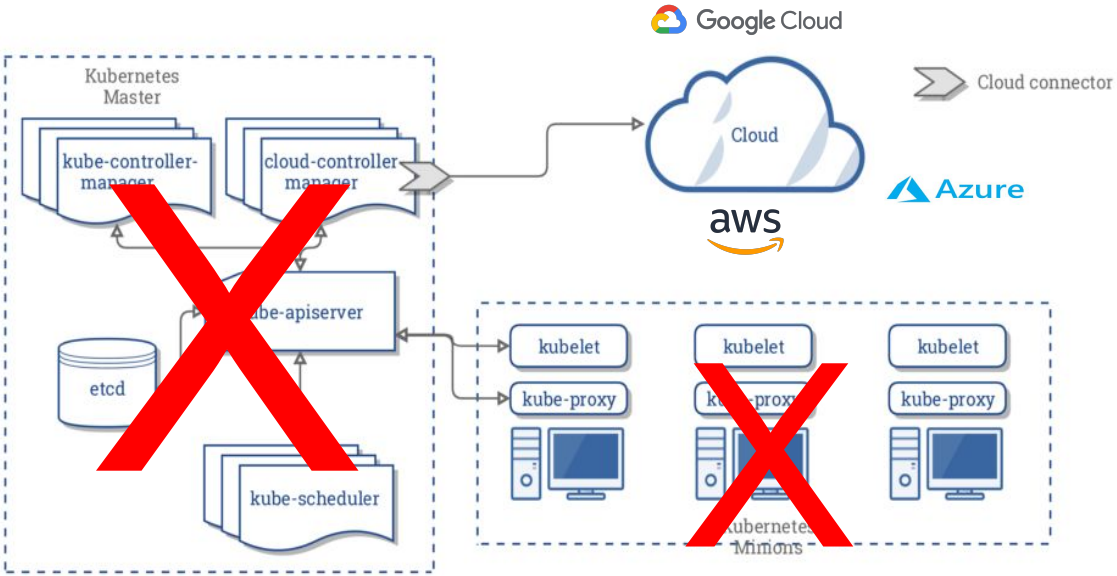
Communication

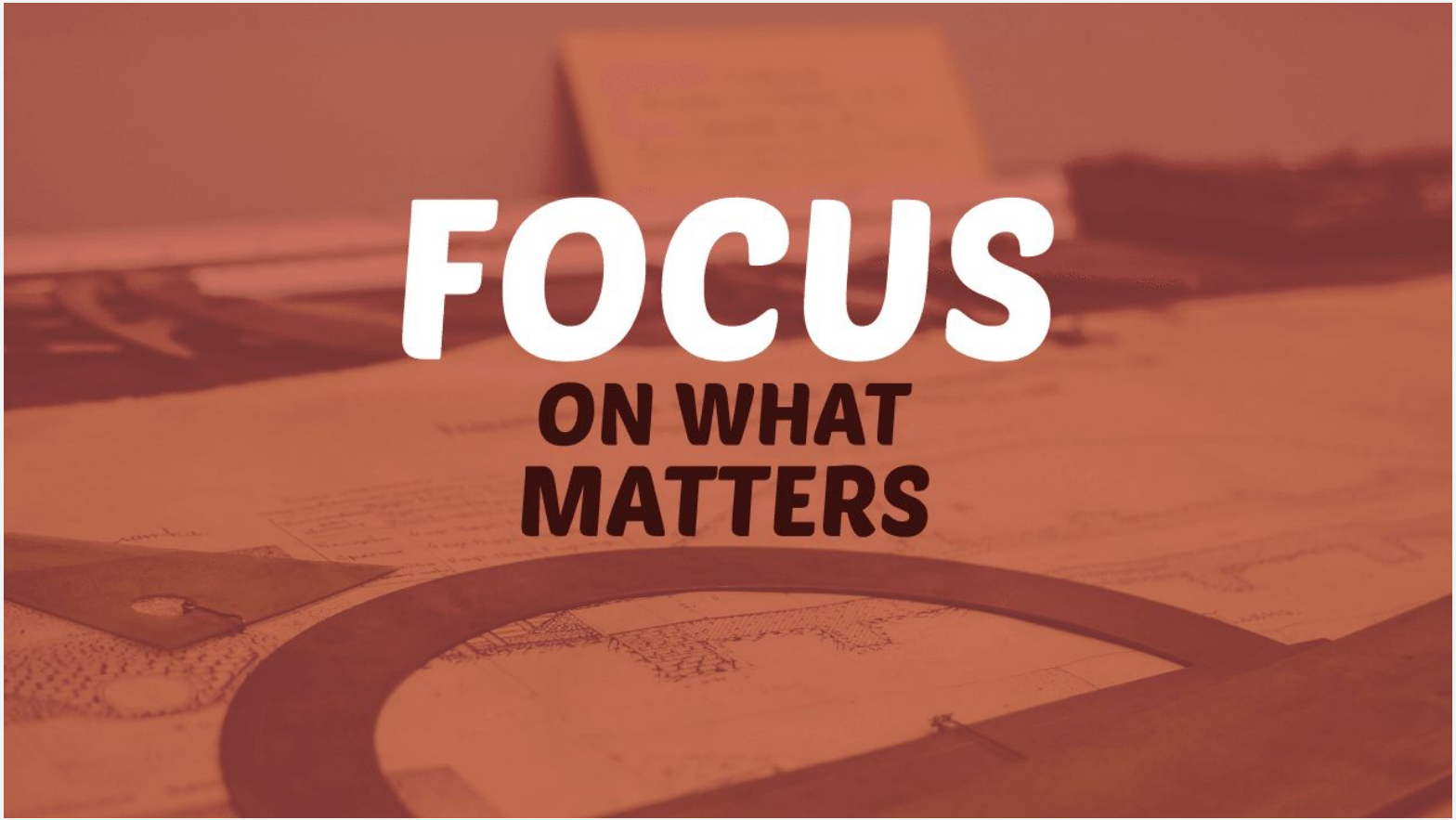


Run K8s for me!

- Just focus on Deployments:
  - Not maintain Master or Nodes
- Cloud Controller Manager (CCM)
  - Code in Kubernetes Git repo:

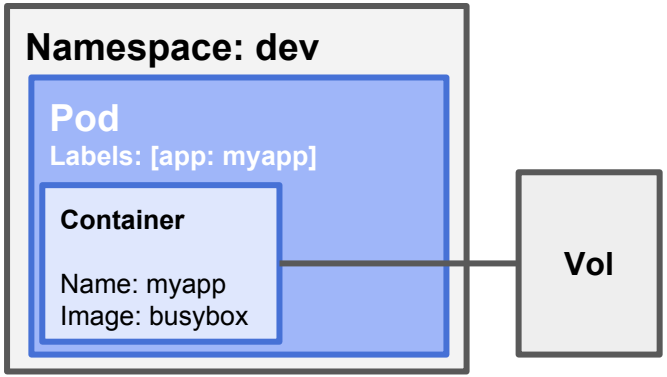





FOCUS
ON WHAT
MATTERS

Basic Concepts

- Pods
- Labels
- Namespaces(kube-system and default)
- Volumes (empty, configmap, secrets...)




```yaml
apiVersion: v1
kind: Pod
metadata:
 name: myapp
 namespace:dev
 labels:
   app: myapp
spec:
 containers:
 - name: myapp-container
   image: busybox
   command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
    volumeMounts:
   - mountPath: /myvol
     name: myvol
 volumes:
 - name: myvol
   emptyDir: {}
```

```
$Kubectl apply -f mypod.yaml
```

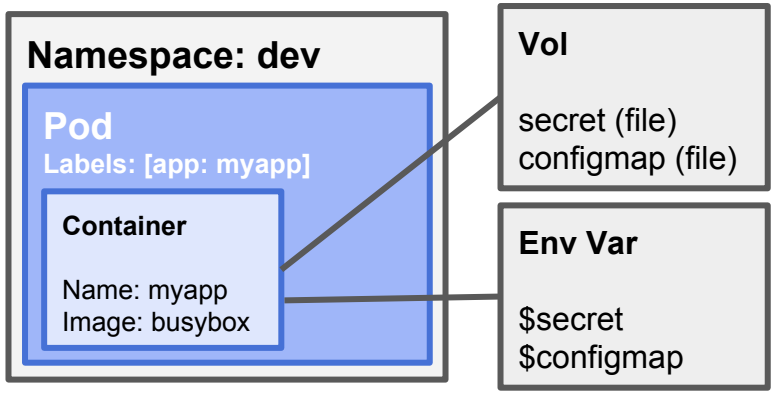
Configmaps and Secrets

- Secrets

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
```

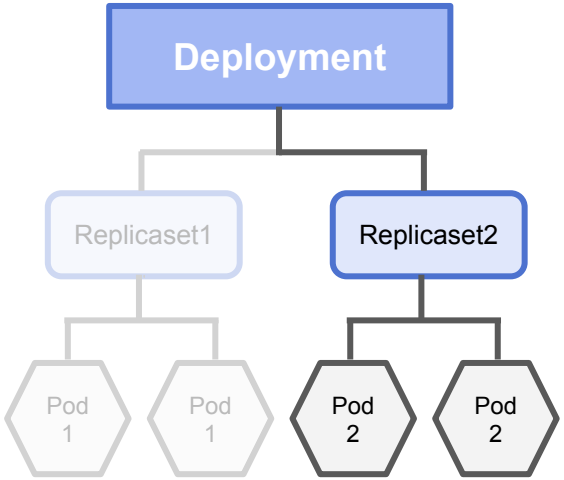
- ConfigMaps

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
data:
  SPECIAL_LEVEL: very
  SPECIAL_TYPE: charm
```



Deployments Concepts

- Deployments
  - scaling
  - Rolling
- ReplicaSet
  - desired state
- Strategies:
  - Recreate
  - RollingUpdate (default)
  - Blue/Green
  - Canary
  - A/B Testing



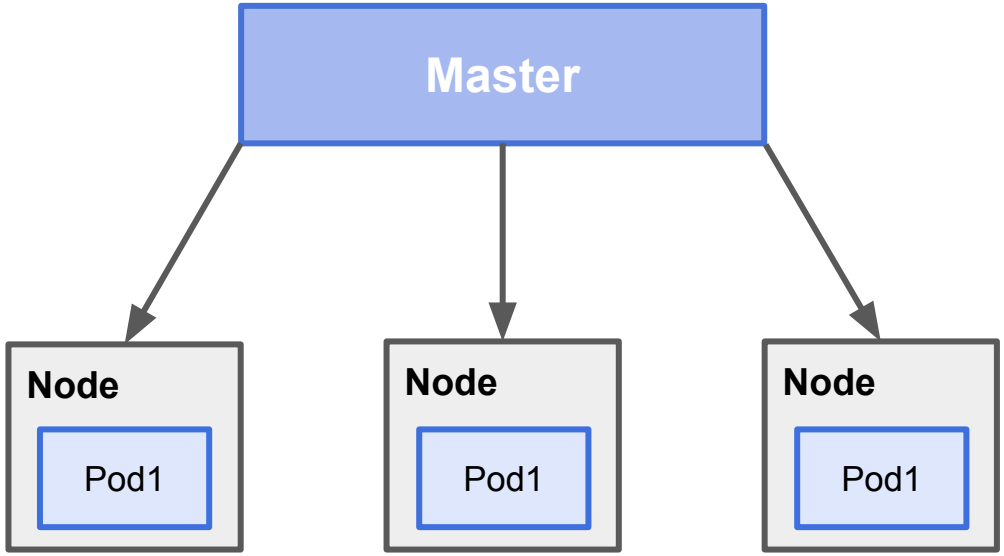

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

StatefulSet

- Deployments which requires persistence.
- Pretty similar specification than for Deployments
- Provides:
  - Stable, unique network identifiers [0,N)
  - Stable, persistent storage/volume
  - Ordered, graceful deployment and scaling (0..N-1)
  - Ordered, graceful deletion and termination (N-1..0)
- For stateless needs use Deployments or ReplicaSet

DaemonSets

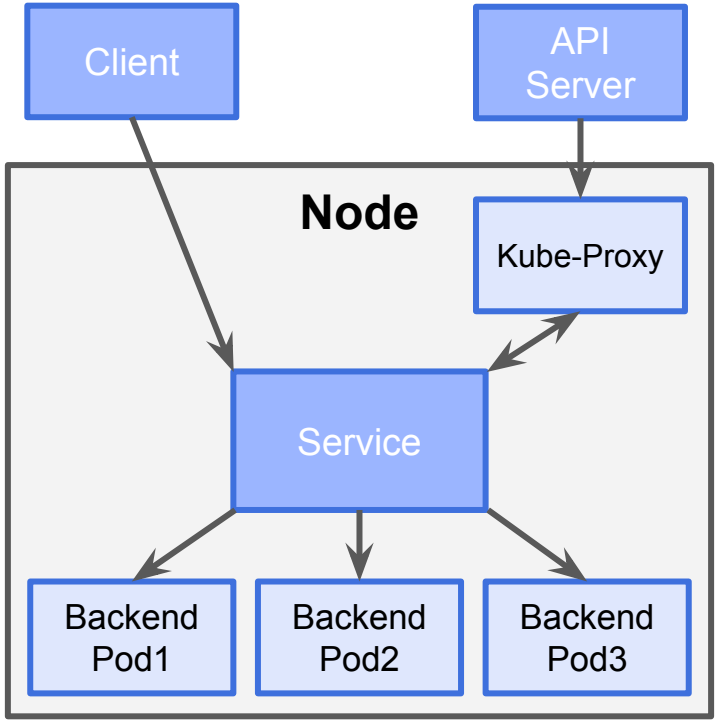
- At least one pod running on each node



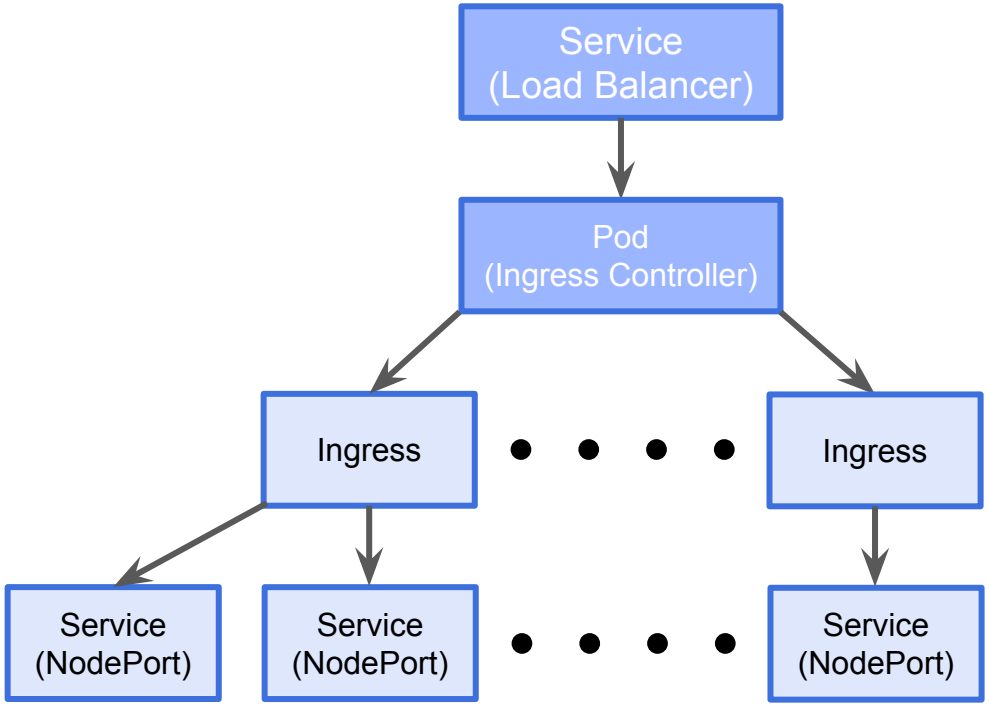
Services

- Services Types
  - ClusterIP
  - NodePort
  - Load Balancer
- Provide:
  - Find pods by label selector
  - Load Balancing
  - Service Discovery

```yaml
kind: Service
apiVersion: v1
metadata:
 name: my-service
spec:
 type: LoadBalancer
 selector:
   app: MyApp
 ports:
 - protocol: TCP
   port: 80
   targetPort: 9376
```

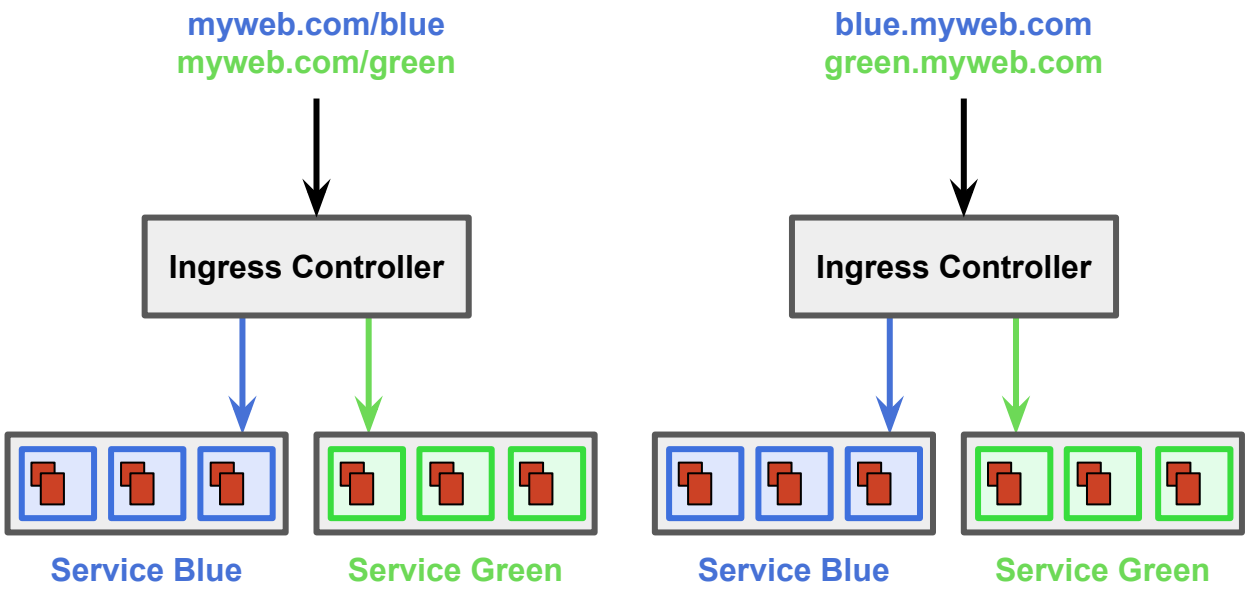


Ingress Controller



- **Pod** running on nodes
- **Service** - Load balancer
- Provides:
  - Load Balancer
  - Proxy Reverse
- Join point:
  - Ingress object

Ingress

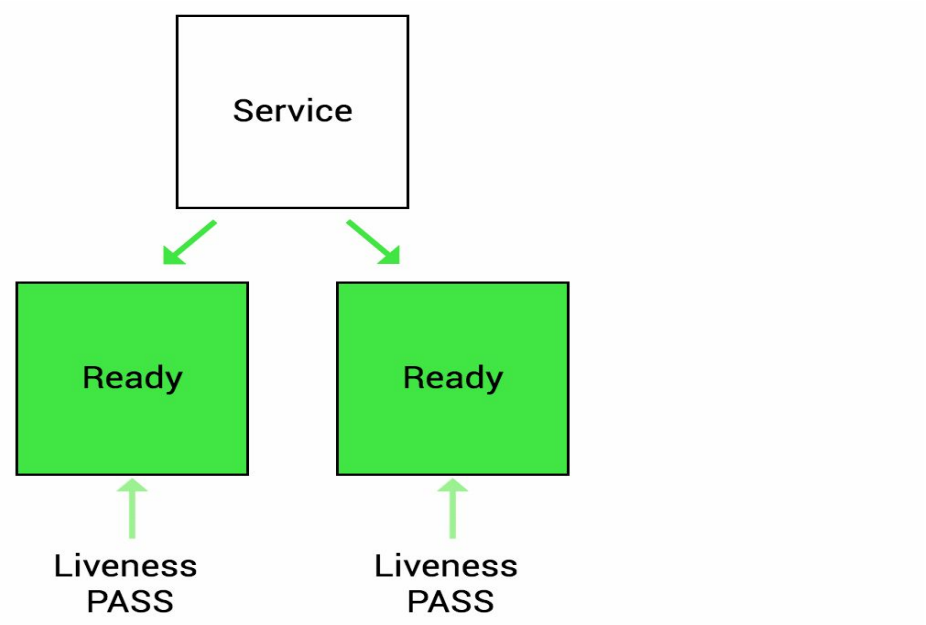



```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: test
spec:
 rules:
 - host: foo.bar.com
   http:
     paths:
     - backend:
         serviceName: s1
         servicePort: 80
 - host: bar.foo.com
   http:
     paths:
     - backend:
         serviceName: s2
         servicePort: 80
```

Health checks: Liveness probe

- **Liveness** probe
  - Command
  - TCP
  - HTTP

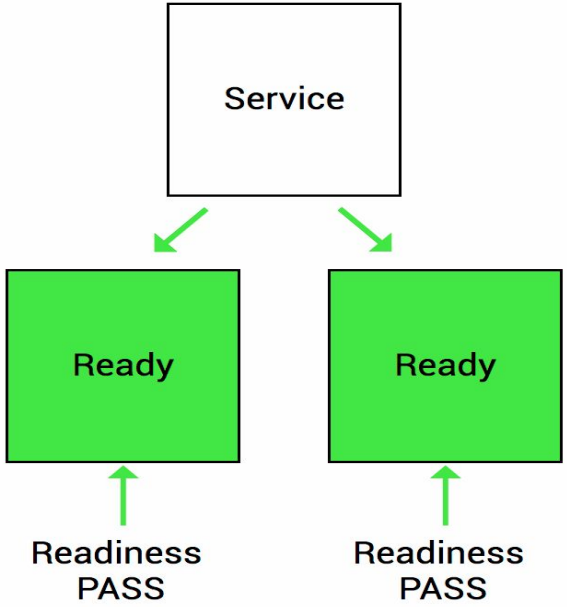
```yaml
livenessProbe:
    httpGet:
      path: /healthz
      port: 8080
      httpHeaders:
      - name: X-Custom-Header
        value: Awesome
    initialDelaySeconds: 3
    periodSeconds: 3
```



Health checks: Readiness probe

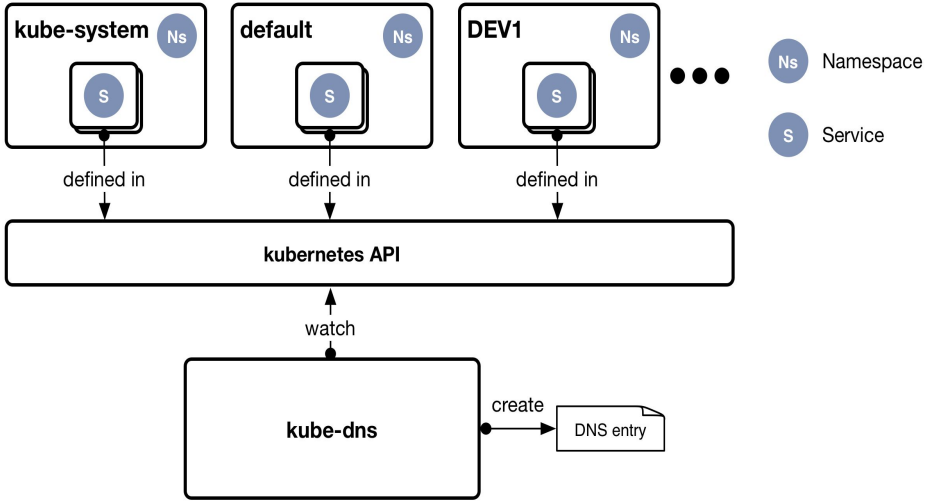
- **Readiness** probe
  - Command
  - TCP
  - HTTP

```yaml
readinessProbe:
  exec:
    command:
    - cat
    - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
```



Service Discovery

- **Kube-dns**: Pod running on nodes
- Watching API for Pod and Services events
- You can **reach your deployments** by the service name thanks to kube-dns
- Default domain: cluster.local
  - **my-svc.my-namespace.svc.**cluster.local
  - **pod-ip.my-namespace.pod.**cluster.local

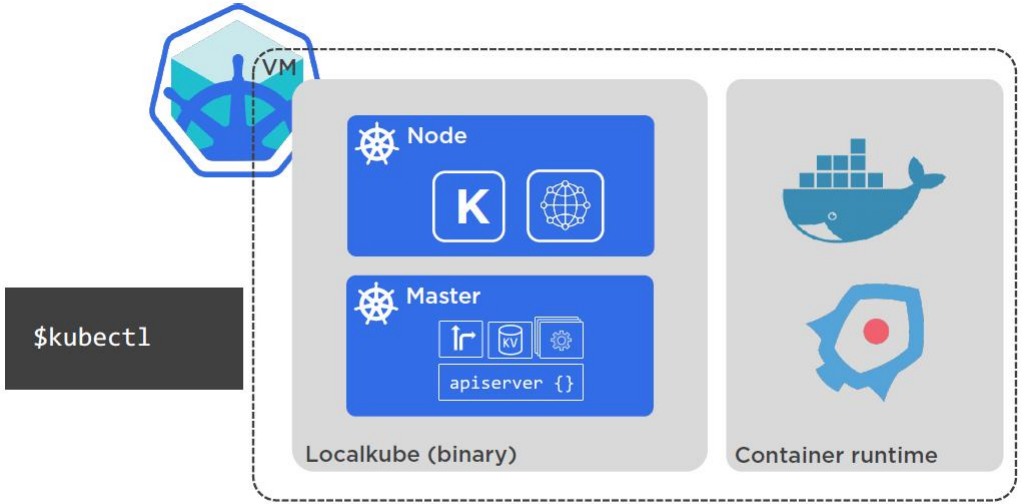


Hello World Kubernetes (Demo)

CONTINO

Run K8s on your local environment

- Docker with Kubernetes
  - Hypervisor Configured
  - Boot2Docker.iso
- Minikube
  - Hypervisor Configured
  - Minikube.iso
  - Kubectl
  - Kubeadm
  - Kubelet
  - Kubernetes containers



Demo Time

http://github.com/contino/kubernetes-101

- Create Pod & Service
- Health check failing container recovery
- Max memory exceeded failing container recovery
- Deploy a faulty container & have no downtime & then deploy a fixed container
- Override container env variables
- Override container env variables and hookup volumes with config maps & secrets
- Change config maps & secrets at runtime

1 pod is ready. The other 2 are in progress still.

What’s Next?

CONTINO

Extending Kubernetes

- CRDs (API groups)
- Custom Admission Controllers
- Join us for next brown bag: **Programmable Infrastructure with Kubernetes**

What’s Next?

- **Helm:** Package Manager for Kubernetes
- **Service Mesh**: Istio - Layer for handling service-to-service communication.
- **Monitoring:** Prometheus
- **Logging:** EFK

Thank you !

CONTINO

QUESTIONS?

London

1 Fore Street,
Moorgate,
London,
EC2Y 9DT,
UK

—
london@contino.io

New York

404 5th Avenue,
New York, NY,
10018EC2Y 9DT,
UK

—
newyork@contino.io

Melbourne

Level 2,
Hub Southern Cross,
696 Bourke St,
Melbourne VIC 3000,
Australia

—
melbourne@contino.io

@ContinoHQ
@ContinoHQ
Contino

CONTINO

contino.io

info@contino.io