Containers and Docker

Lab 1

001-Lab-Setup

Lab Goals:

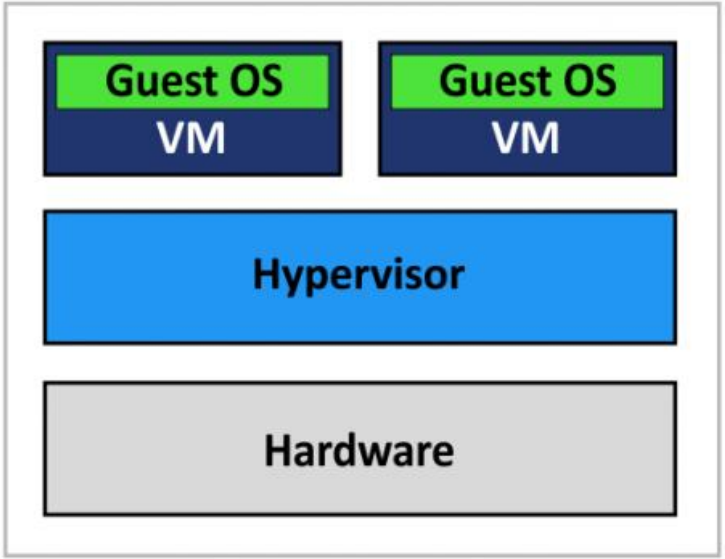
- Set up infrastructure
- Get acquainted with Cloud Shell / Minikube
- Explore the git repo

Before Containers

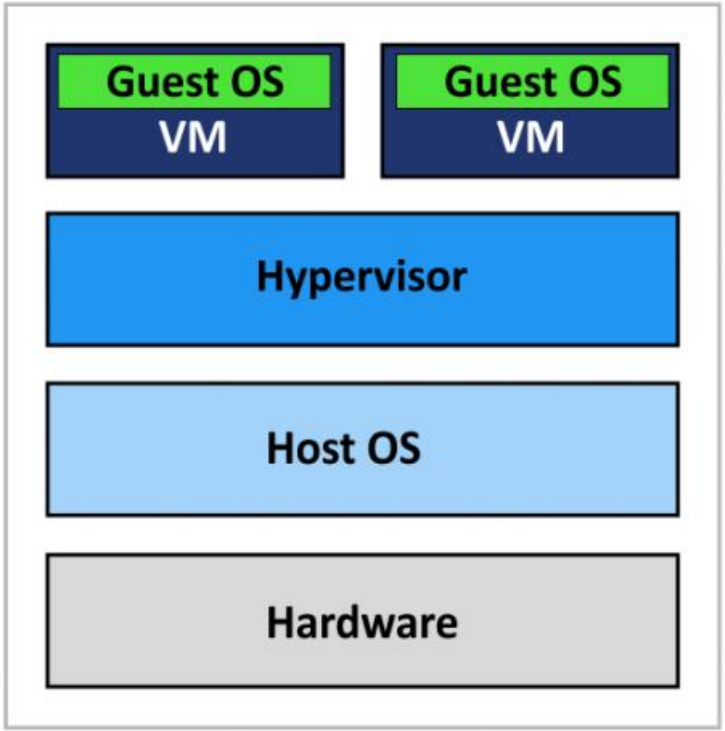
- 'Big' and 'fast' servers
- 1 server = 1 application
- Migration?
- Cost?
- Environment Replication?
- Scalable?

Virtual Machines (VMs)

- 1 server = multiple applications
- Hypervisor (VM Monitor) Architecture
  - Type 1 (Windows Hyper-V)
  - Type 2 (VMWare)
- Host hardware allocation across all apps
- Each VM contains:
  - Hardware (Virtual)
  - OS
  - Application

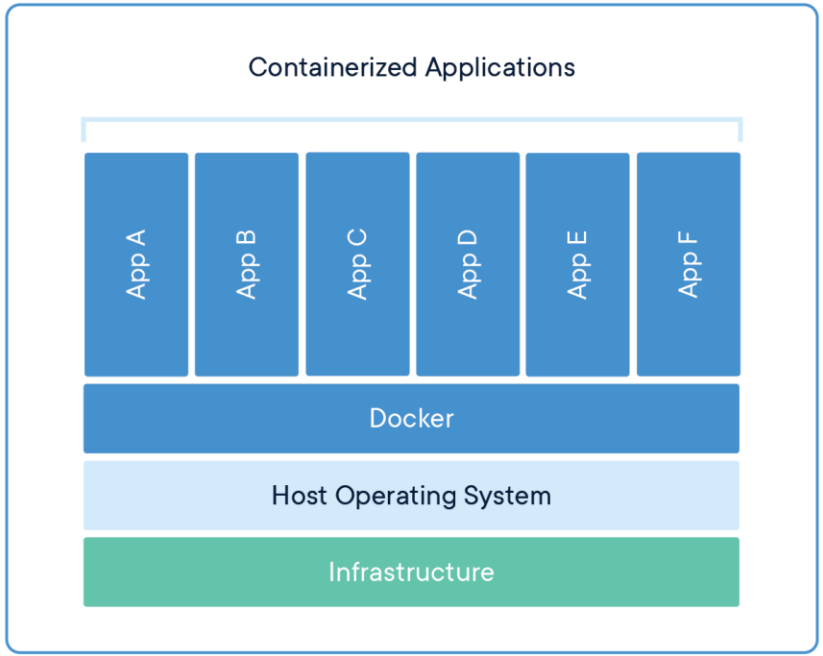


Type 1 Hypervisor
(Bare-Metal Architecture)



Type 2 Hypervisor
(Hosted Architecture)

Containers

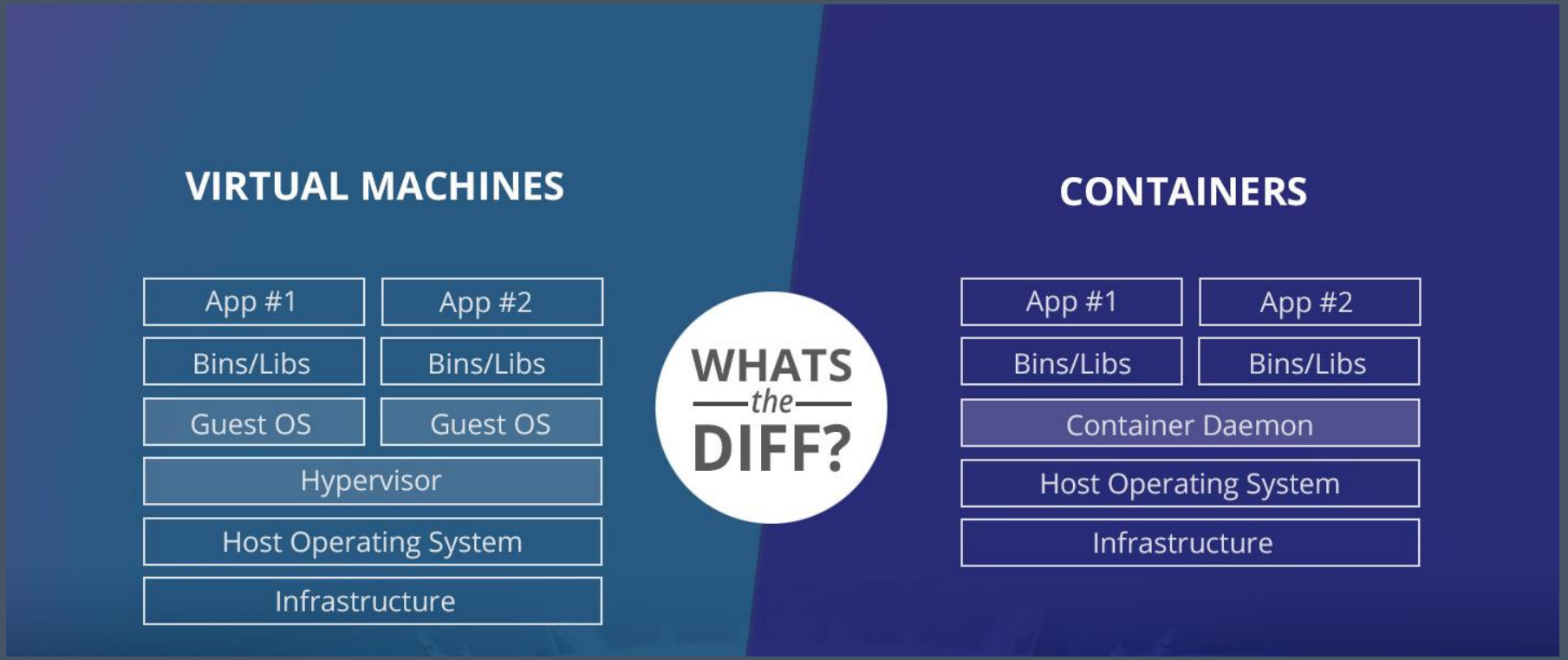


- Standardised unit of software
  - Application layer construction
- Own libraries and packages
- Shared OS

Containers

- Standardisation and productivity
- Compatibility and maintainability
- CD/CI

VMs VS Containers



Docker 101

Image

The basis of a Docker container. The content at rest.

Container

The image when it is 'running.' The standard unit for app service

Engine

The software that executes commands for containers. Networking and volumes are part of Engine. Can be clustered together.

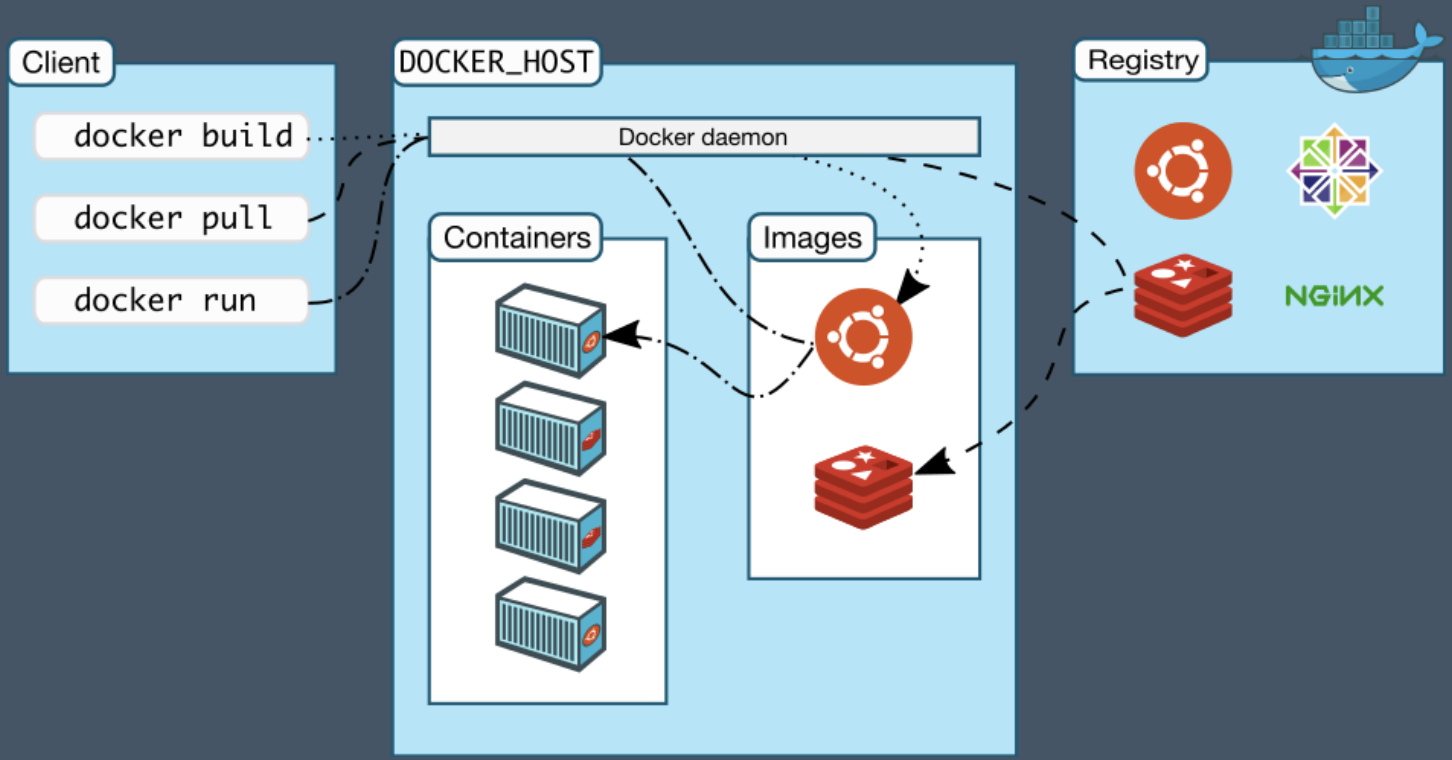
Registry

Stores, distributes and manages Docker images

Control Plane

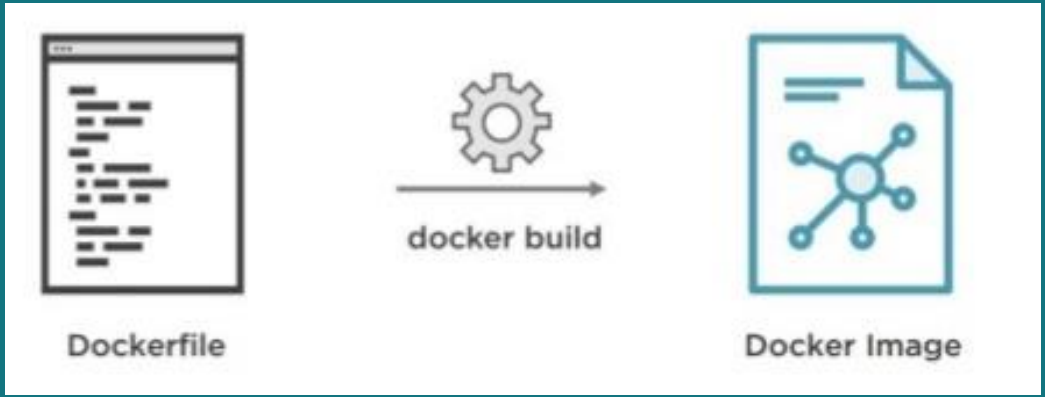
Management plane for container and cluster orchestration

Docker Engine



DockerFile

- Text document for building docker image
- Contains all CLI instructions for the build



Lab 2

002-Containerizing-An-Application

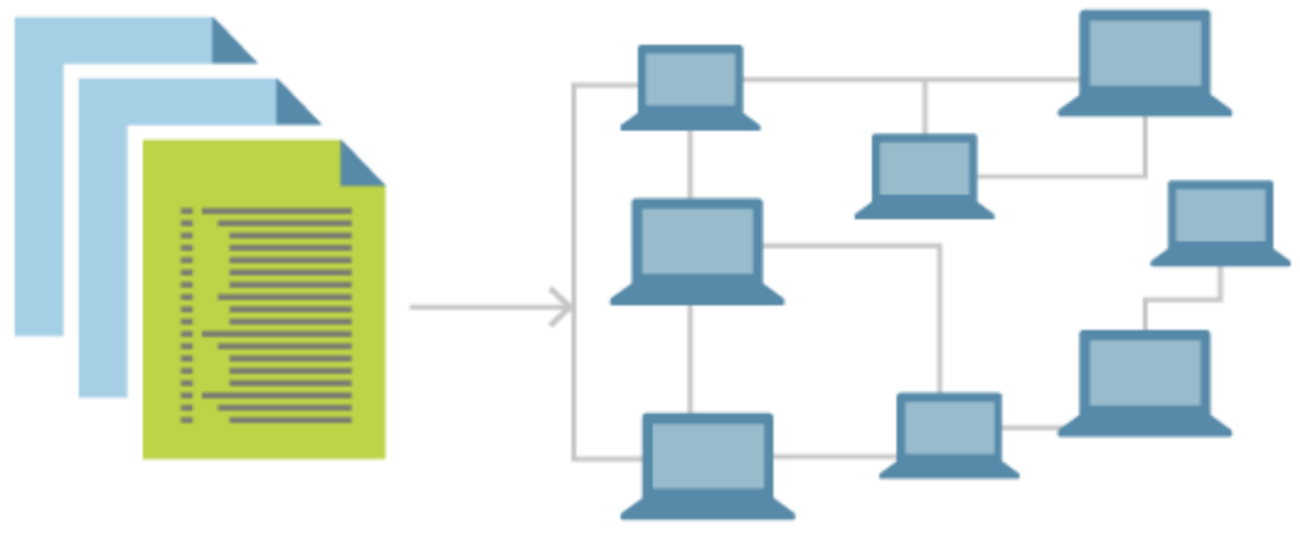
Lab Goals:

- Run Golang API locally
- Build Docker Image
- Run the Docker Container
- Use Environment Variables

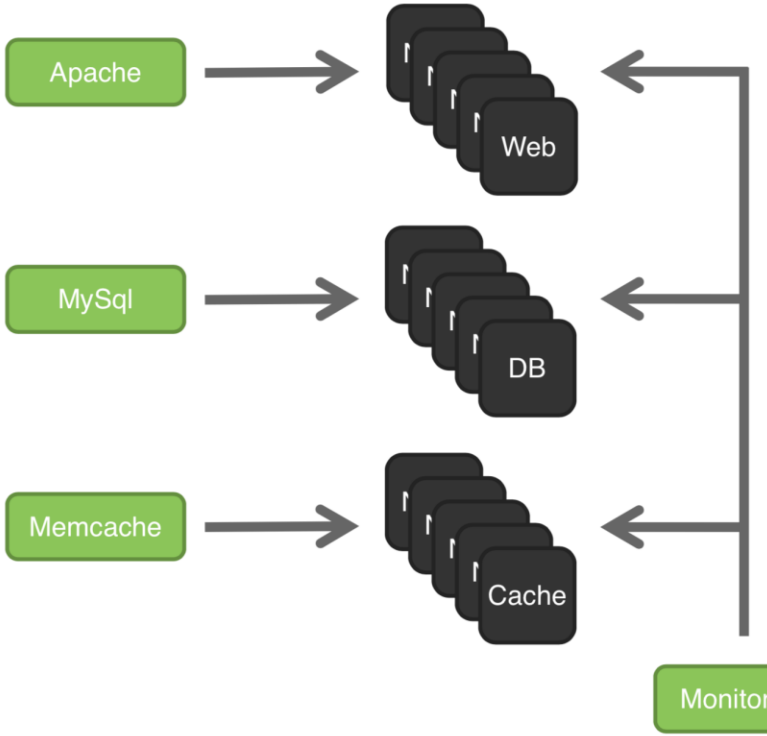
An introduction to Kubernetes (k8s)

Infrastructure as Code (intro to k8s)

Infrastructure as code (IaC) allows for infrastructure to be deployed using a high-level, descriptive language. IaC treats the entire infrastructure as if it is software. Because, it's all software...

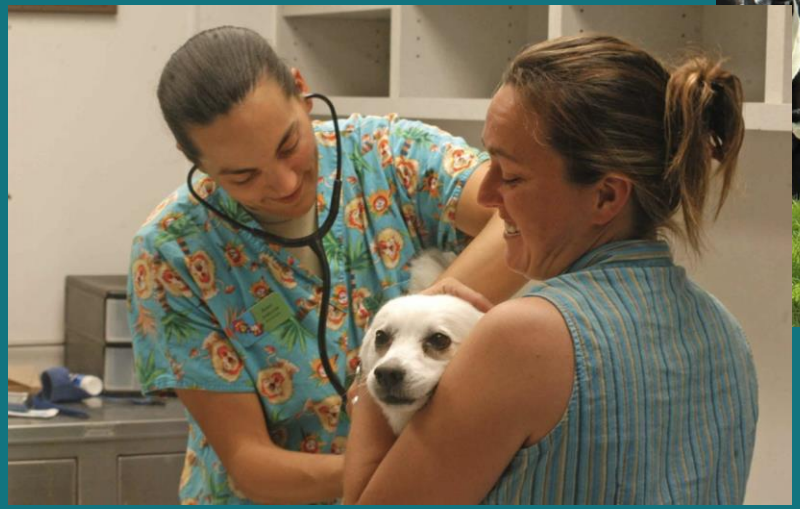


Grouping and Tagging



- Tagging your servers and containers applies the required set of automation
- A base set of for all servers
- Each server can have multiple tags
- Map tags to security requirements

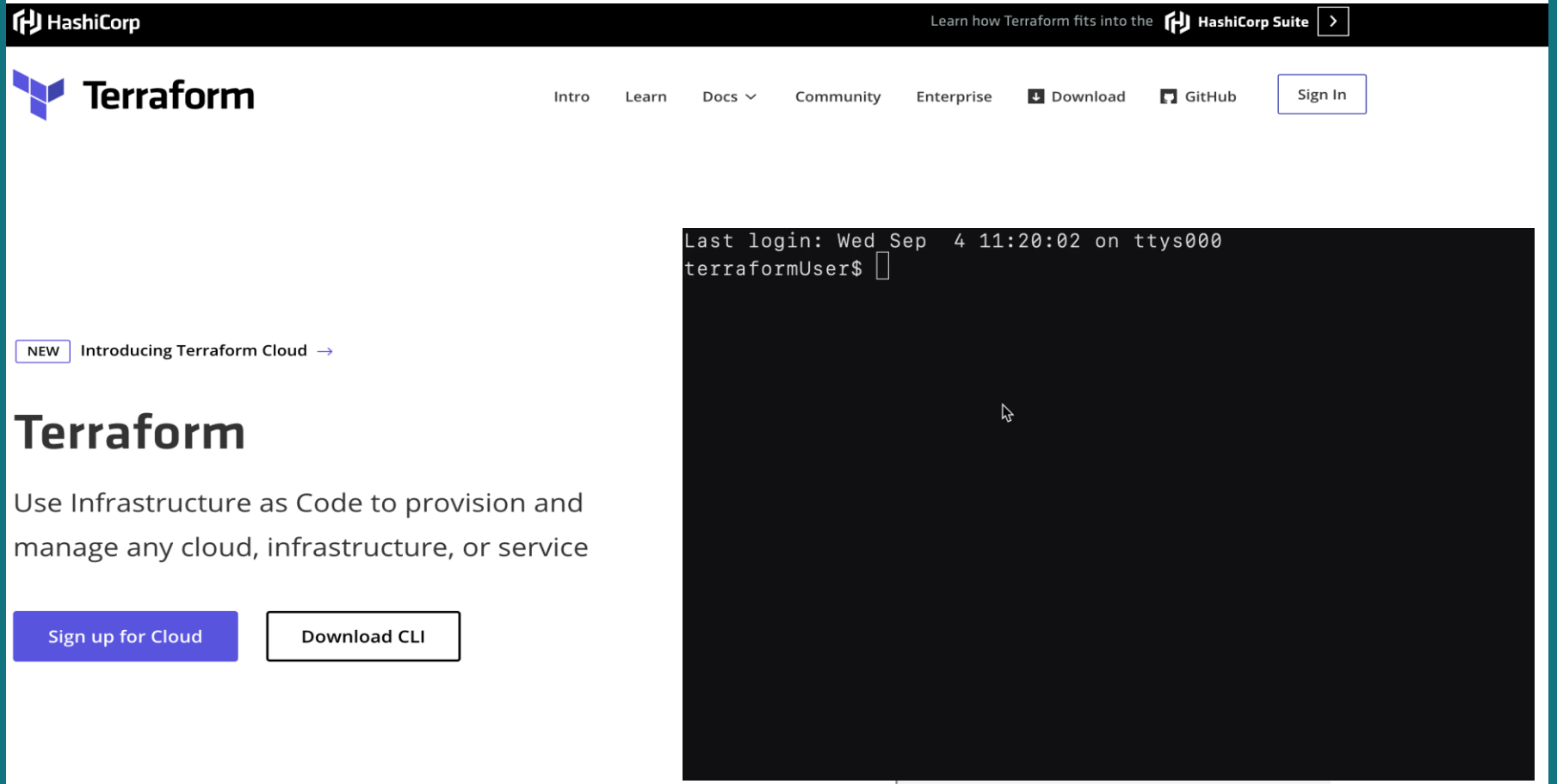
Cattle, not pets



Security wins !

- Security team now has insight into the entire system
- Infrastructure is auditable and version controlled, just like source code
- Patching can be applied programmatically with a high level of certainty
- Alerting can be built for changes to specific areas of the infrastructure
- A new firewall rule is created or deleted – Administrative user is created – New VPC rolled out
- Testing can occur much earlier in the pipeline

Infrastructure as Code - Terraform



Infrastructure as Code – K8s

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-site-ingress
  namespace: my-site-prod
  annotations:
    kubernetes.io/tls-acme: "true"
    kubernetes.io/ingress.class: "gce"
    kubernetes.io/ingress.global-static-ip-name: my-site-external-ip
spec:
  tls:
  - hosts:
    - api.my.site
    - my.site
    secretName: my-site-cert
  rules:
  - host: api.my.site
    http:
      paths:
      - path: /*
        backend:
          serviceName: app-api
          servicePort: 80
  - host: my.site
    http:
      paths:
      - path: /*
        backend:
          serviceName: my-site-prod
          servicePort: 80
```

K8s Intro

Kubernetes is an open-source platform built to automate **deployment**, **scaling** and **orchestration** of containers.

K8S is **portable**. Clusters can be deployed on a public/private cloud, on prem, and even on your laptop.

K8S is **customizable**. It is modular and extensible to fit a variety of use-cases.

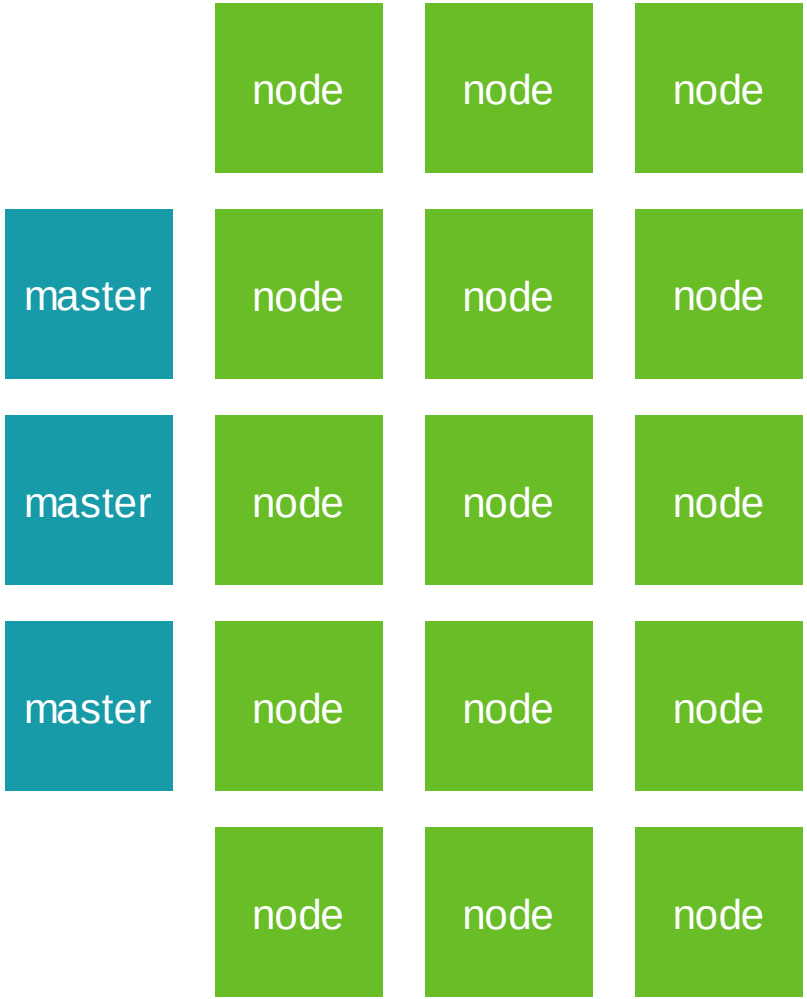
K8s Intro

K8S is **scalable**. It provides self-healing, auto scaling, and replication out of the box.

virtual machines that Kubernetes manages

cluster

cluster

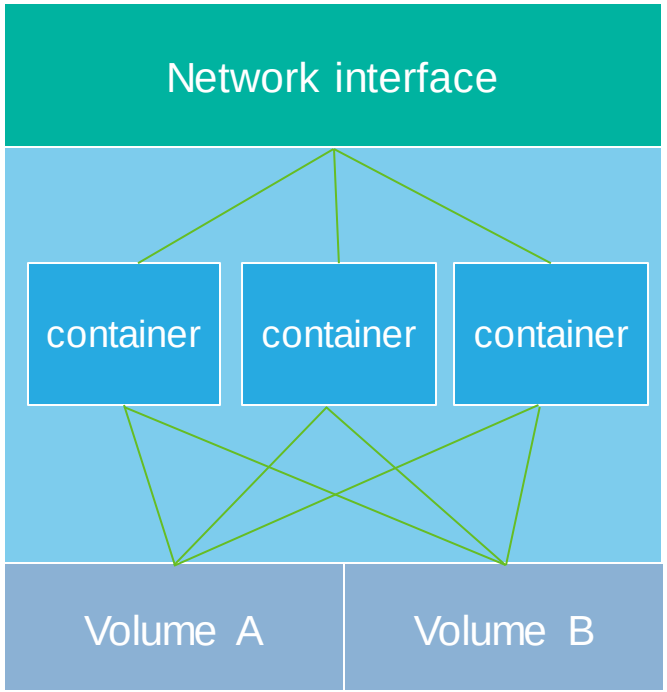

node
node
node
master
node
node
node
master
node
node
node
master
node
node
node
node
node
node

cluster

pod

group of containers sharing storage and network

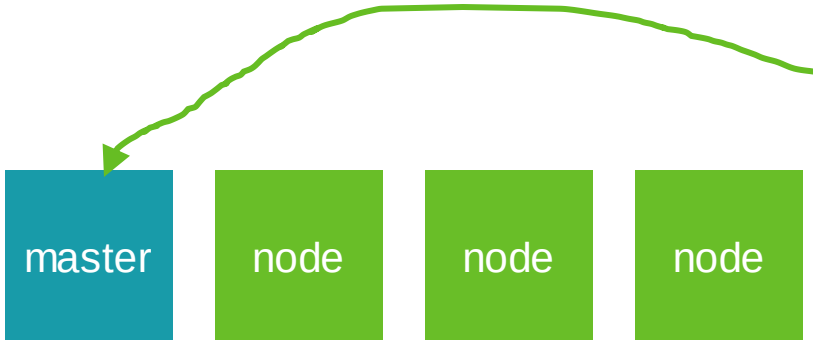
pod


Network interface
container
container
container
Volume A
Volume B

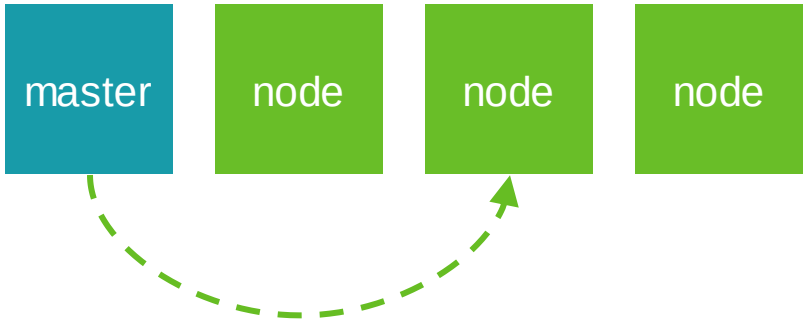
pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis-rails
spec:
  containers:
  - name: key-value
    image: redis
    ports:
    - containerPort: 6379
  - name: rails-frontend
    image: rails
    ports:
    - containerPort: 3000
```

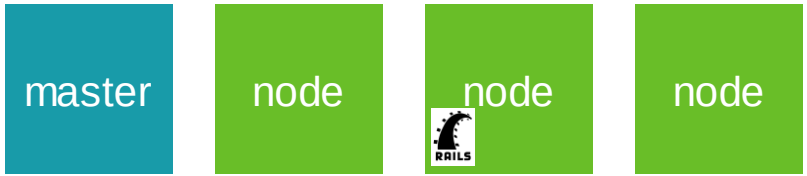
pod.yaml



pod.yaml



pod.yaml



pod.yaml

deployment

ensure N pods are up and running

deployment

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: rails-deployment
  labels:
    app: rails
spec:
  replicas: 4
  selector:
  matchLabels:
    app: rails
  template:
    metadata:
      labels:
        app: rails
    spec:
      containers:
      - name: key-value
        image: redis
        ports:
        - containerPort: 6379
      - name: rails-frontend
        image: rails
        ports:
        - containerPort: 3000
```

deploy.yaml

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: rails-deployment
  labels:
    app: rails
spec:
  replicas: 4
  selector:
  matchLabels:
    app: rails
  template:
    metadata:
      labels:
        app: rails
    spec:
      containers:
      - name: key-value
        image: redis
        ports:
        - containerPort: 6379
      - name: rails-frontend
        image: rails
        ports:
        - containerPort: 3000
```

deploy.yaml

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: rails-deployment
  labels:
    app: rails
spec:
  replicas: 4
  selector:
  matchLabels:
    app: rails
  template:
    metadata:
      labels:
        app: rails
    spec:
      containers:
      - name: key-value
        image: redis
        ports:
        - containerPort: 6379
      - name: rails-frontend
        image: rails
        ports:
        - containerPort: 3000
```

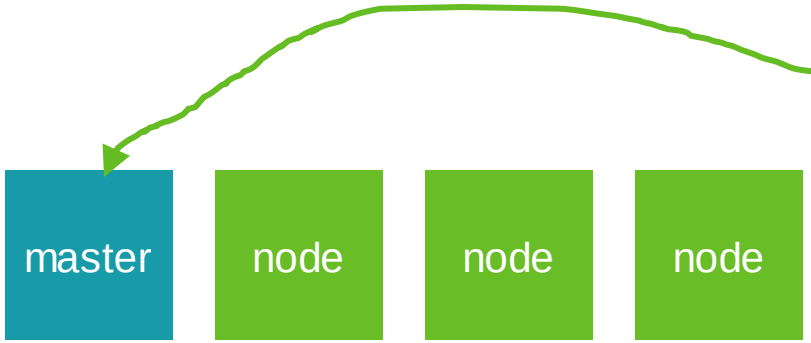
deploy.yaml

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: rails-deployment
  labels:
    app: rails
spec:
  replicas: 4
  selector:
  matchLabels:
    app: rails
  template:
    metadata:
      labels:
        app: rails
    spec:
      containers:
      - name: key-value
        image: redis
        ports:
        - containerPort: 6379
      - name: rails-frontend
        image: rails
        ports:
        - containerPort: 3000
```

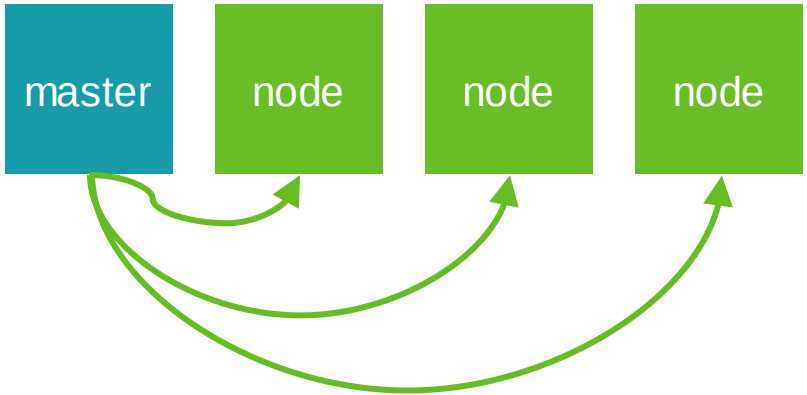
deploy.yaml

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: rails-deployment
  labels:
    app: rails
spec:
  replicas: 4
  selector:
  matchLabels:
    app: rails
  template:
    metadata:
      labels:
        app: rails
    spec:
      containers:
      - name: key-value
        image: redis
        ports:
        - containerPort: 6379
      - name: rails-frontend
        image: rails
        ports:
        - containerPort: 3000
```

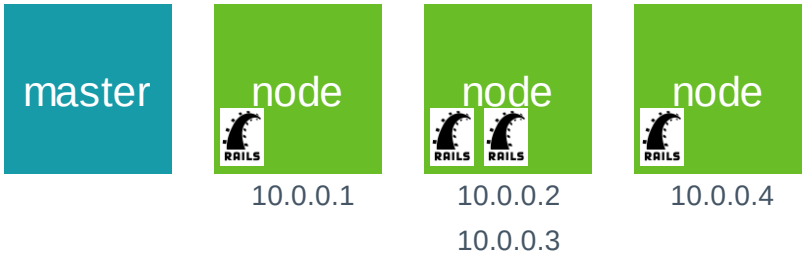
deploy.yaml



deploy.yaml



deploy.yaml

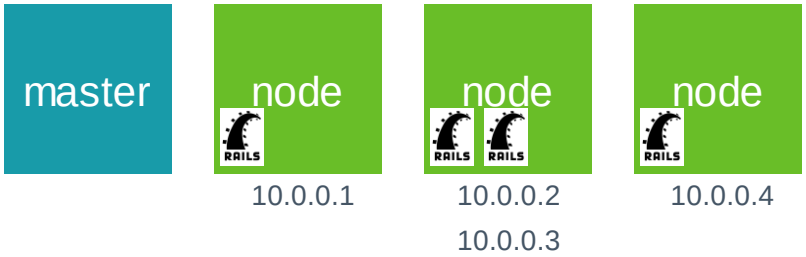


deploy.yaml

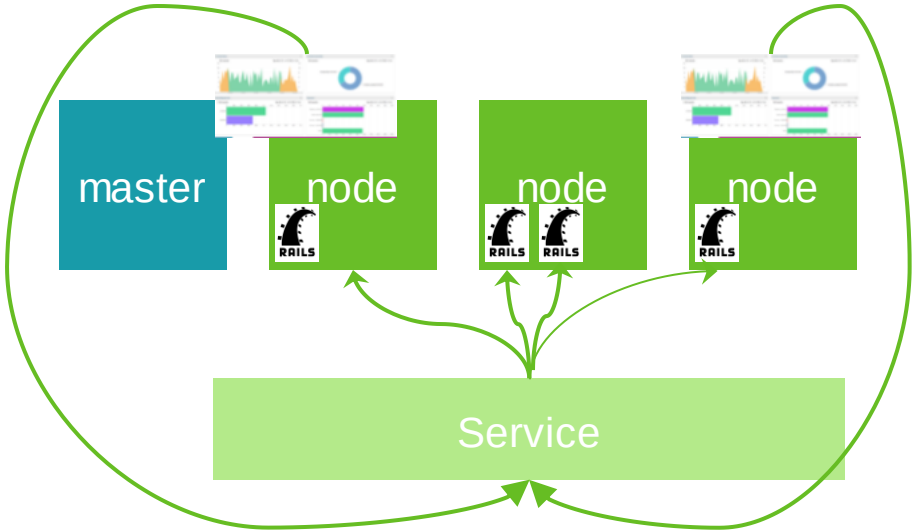
service

abstraction layer that enables pod communication

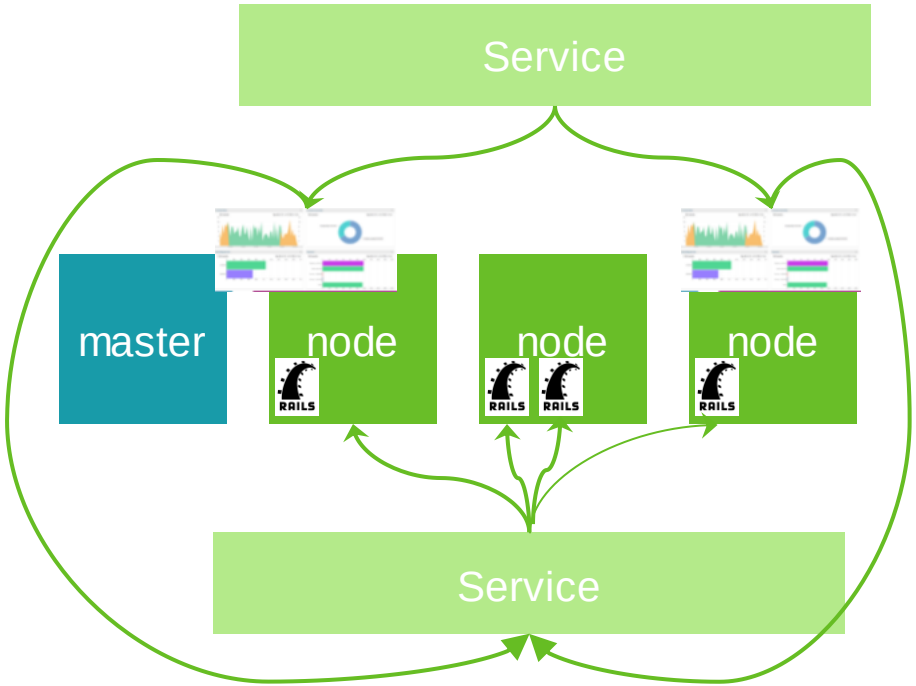
service



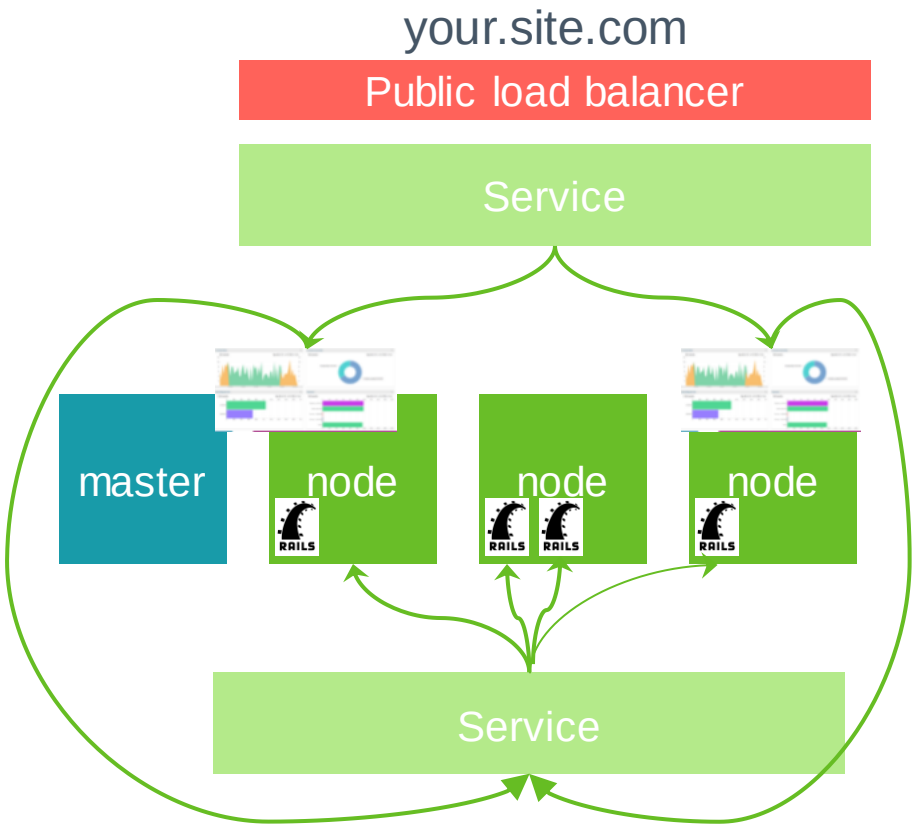
service



service



service


your.site.com
Public load balancer
Service
master
node
node
node
Service

service

```yaml
kind: Service
apiVersion: v1
metadata:
  name: web-frontend
spec:
  ports:
  - name: http
    port: 80
    targetPort: 3000
    protocol: TCP
  selector:
    app: rails
  type: LoadBalancer
```

svc.yaml

```yaml
kind: Service
apiVersion: v1
metadata:
  name: web-frontend
spec:
  ports:
  - name: http
    port: 80
    targetPort: 3000
    protocol: TCP
  selector:
    app: rails
  type: LoadBalancer
```

svc.yaml

```
kind: Service
apiVersion: v1
metadata:
  name: web-frontend
spec:
  ports:
  - name: http
    port: 80
    targetPort: 3000
    protocol: TCP
  selector:
    app: rails
  type: LoadBalancer
```

svc.yaml

Labels and Selectors

Metadata (key-value) which can be attached to a resource

Labels

Labels

Used for identification such as app name, tier, environment

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: rails-deployment
  labels:
    app: rails
spec:
  replicas: 4
  selector:
  matchLabels:
    app: rails
  template:
    metadata:
      labels:
        app: rails
    spec:
      containers:
      - name: key-value
        image: redis
        ports:
        - containerPort: 6379
      - name: rails-frontend
        image: rails
        ports:
        - containerPort: 3000
```

deploy.yaml

Provides loose coupling between objects

Selectors

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: rails-deployment
  labels:
    app: rails
spec:
  replicas: 4
  selector:
  matchLabels:
    app: rails
  template:
    metadata:
      labels:
        app: rails
    spec:
      containers:
      - name: key-value
        image: redis
        ports:
        - containerPort: 6379
      - name: rails-frontend
        image: rails
        ports:
        - containerPort: 3000
```

deploy.yaml

Ingress

configure external access to your cluster

Ingress

```yaml
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
  name: web-ingress
spec:
  backend:
    serviceName: web-frontend
    servicePort: 80
```

ingress.yaml

```
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
  name: web-ingress-vhosts
  rules:
  - host: sub.domain.com
     http: paths:
      - backend:
          serviceName: web-frontend-1
          servicePort: 80
 - host: other.domain.com
    http: paths:
      - backend:
          serviceName: web-frontend-2
          servicePort: 80
```

ingress.yaml

manage different environments in the same cluster

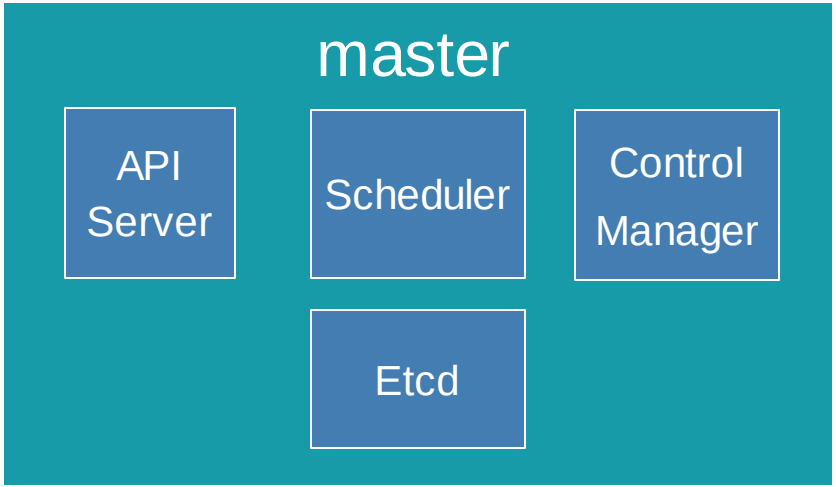
namespace

```yaml
kind: Namespace
apiVersion: v1
metadata:
  name: development
```

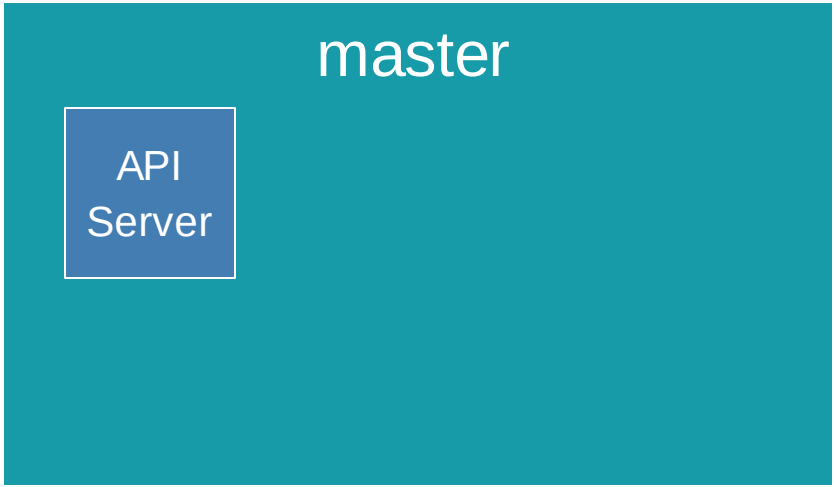
ns.yaml

K8s internals

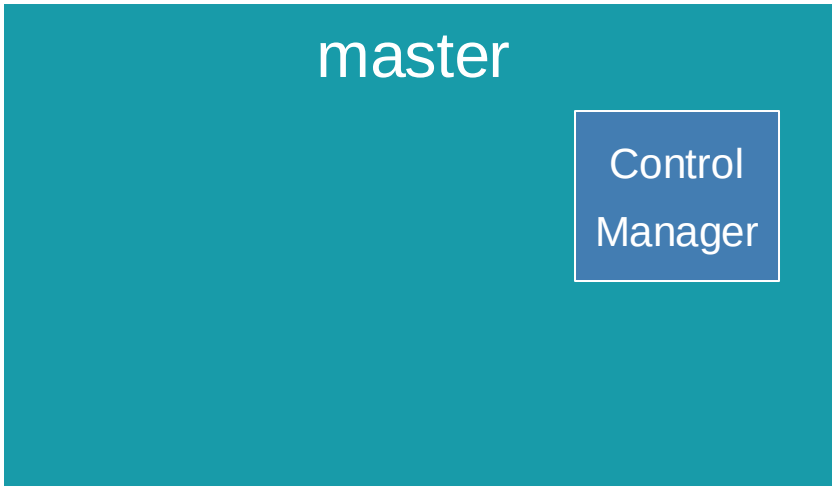
cluster



master



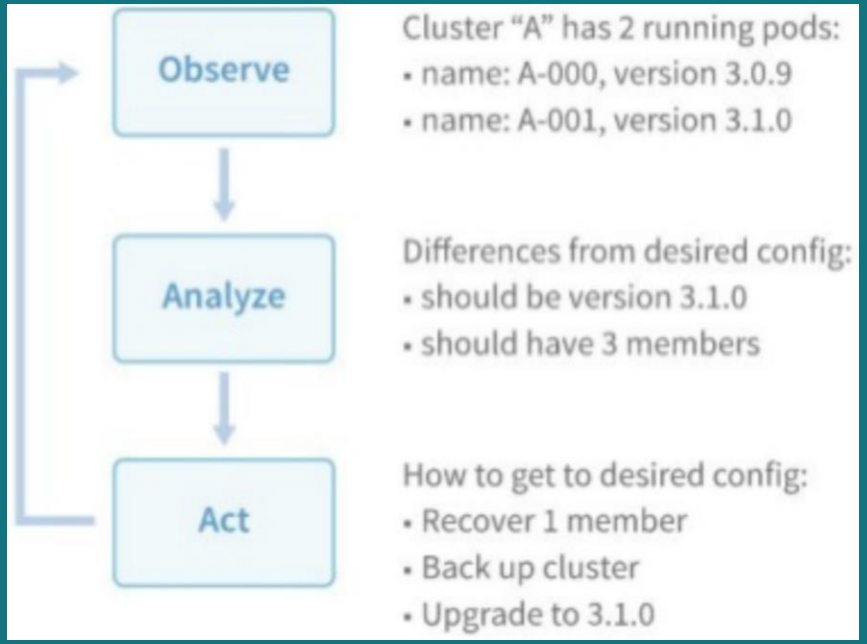
master



master

Controllers

Perform reconciliation loops on cluster resources and converge ***desired*** state with the ***actual*** state.




master
Control Manager
Node Controller
Replication Controller
ServiceAccount Controller
…

master

Node Controller

Responsible for noticing when nodes go up and down

Replication Controller

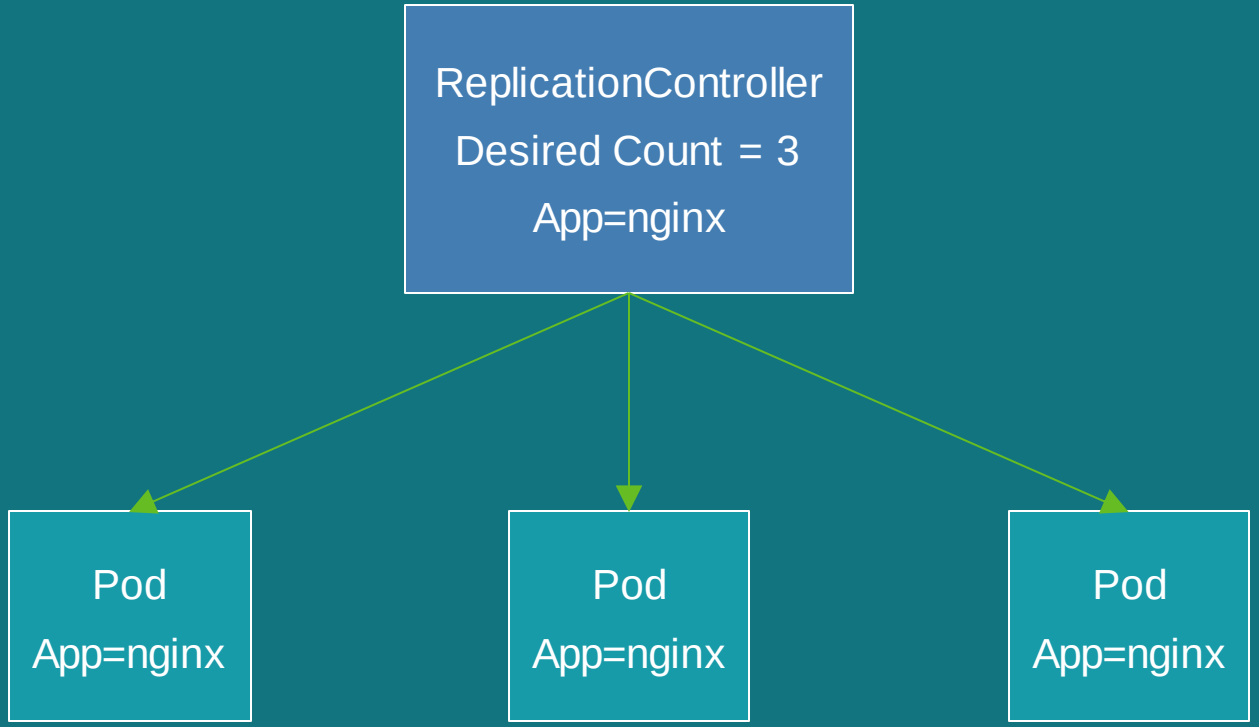
Responsible for maintaining the correct number of pods for every replication controller object

ServiceAccount Controller

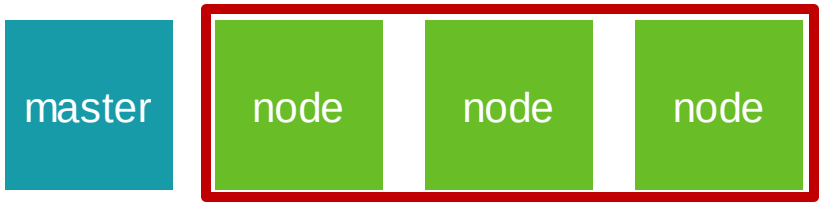
Creates default accounts and API access tokens

...

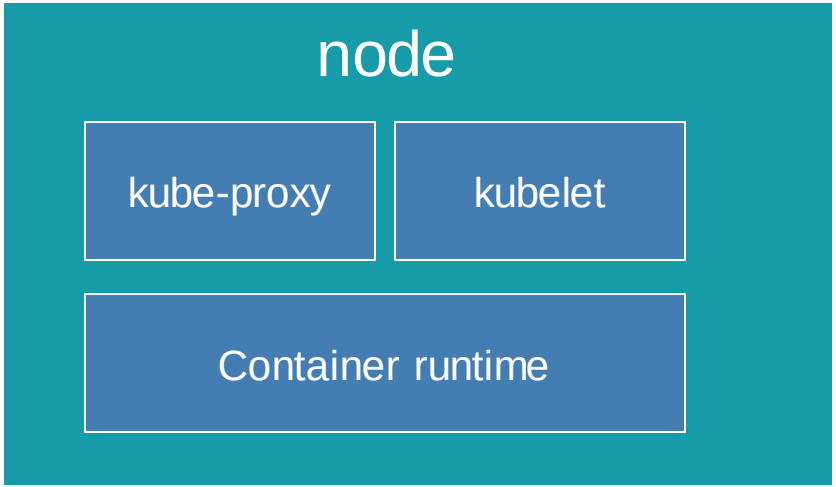
master


ReplicationController
Desired Count = 3
App=nginx
Pod
App=nginx
Pod
App=nginx
Pod
App=nginx

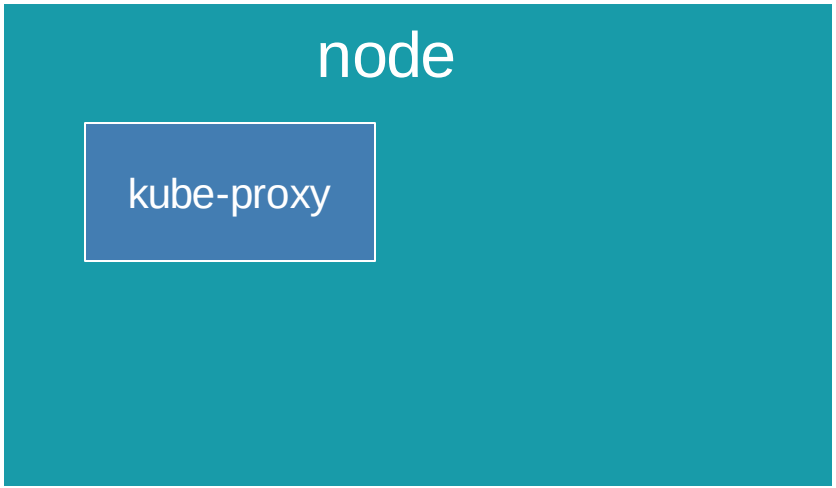
master



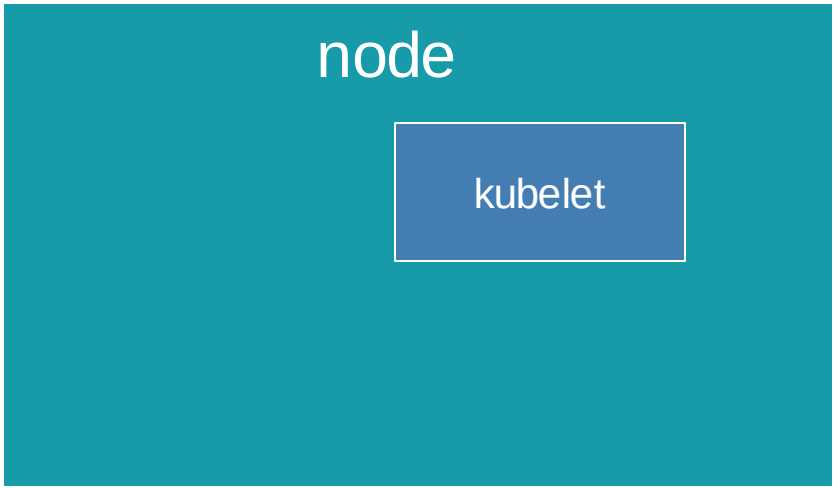
node



node



node



node

node

Container runtime

rkt

node

Lifecycle of a pod in 11 steps

kubectl

HTTPS

master

apiserver

scheduler

etcd

Replication Controller

node

node

kubelet

proxy

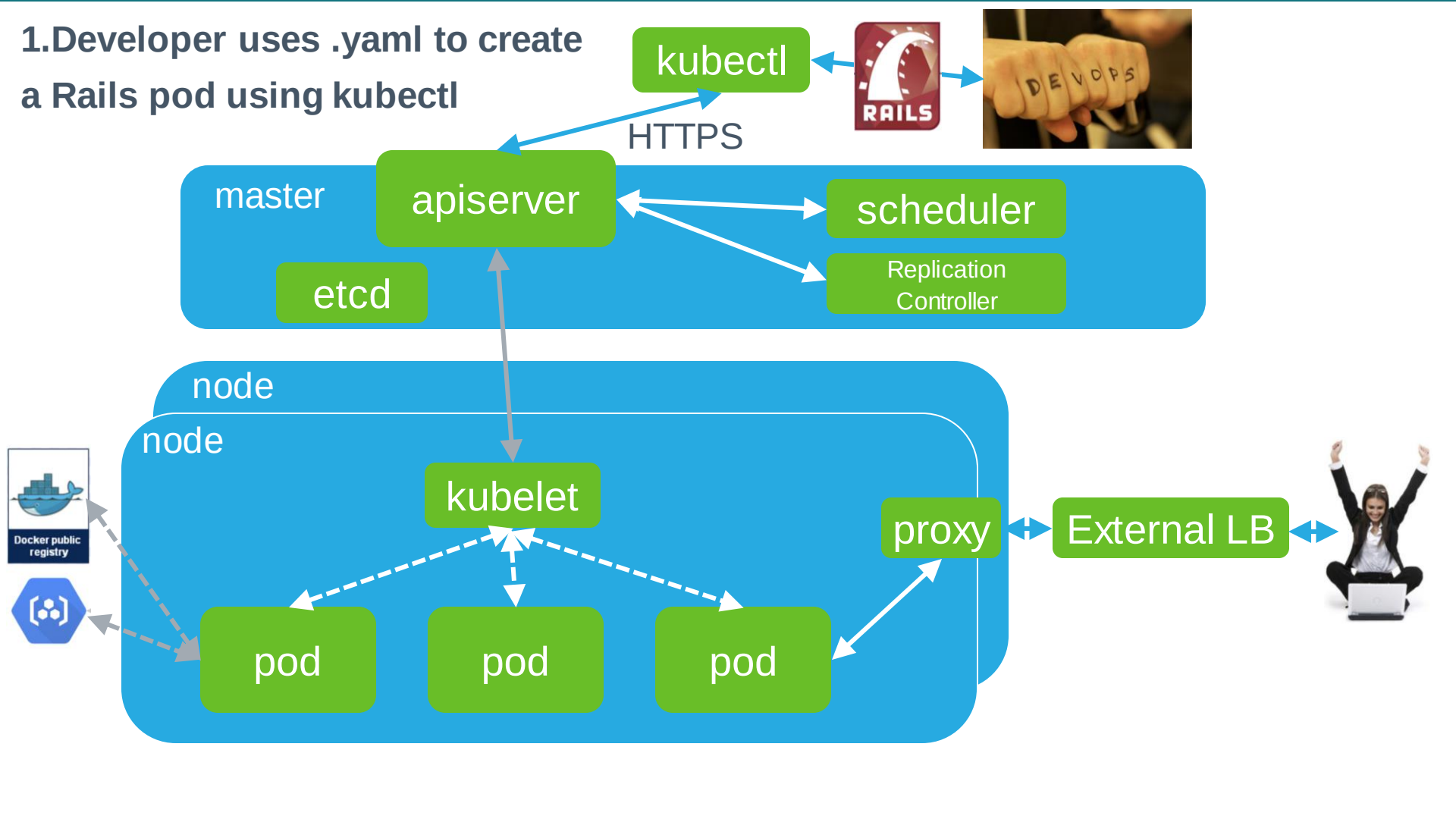
External LB

pod

pod

pod

1.Developer uses .yaml to create a Rails pod using kubectl



2. API Server validates the request

kubectl

HTTPS

RAILS

master

apiserver

scheduler

Replication Controller

etcd

node

node

kubelet

proxy

External LB

Docker public registry

pod

pod

pod

3. The desired state is persisted to etcd

kubectl

HTTPS

master

apiserver

scheduler

Replication Controller

etcd

node

node

kubelet

proxy

External LB

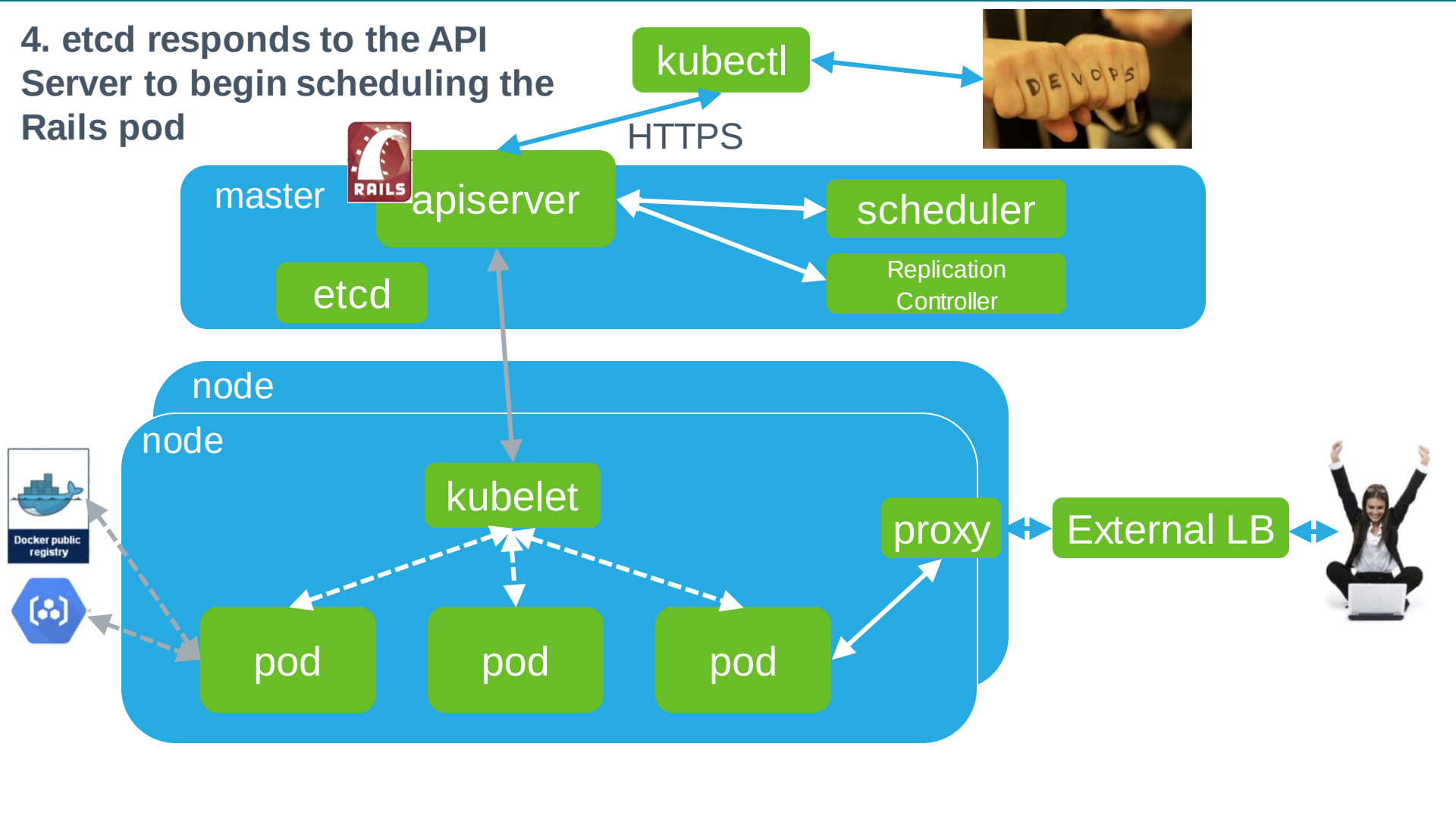
Docker public registry

pod

pod

pod

4. etcd responds to the API Server to begin scheduling the Rails pod



5. The Scheduler decides where the pod should run

kubectl

HTTPS

master

apiserver

scheduler

RAILS

etcd

Replication Controller

node

node

kubelet

proxy

External LB

Docker public registry

pod

pod

pod

5. The Scheduler decides where the pod should run

kubectl

HTTPS

master

apiserver

scheduler

RAILS

etcd

Replication Controller

node

node

kubelet

proxy

External LB

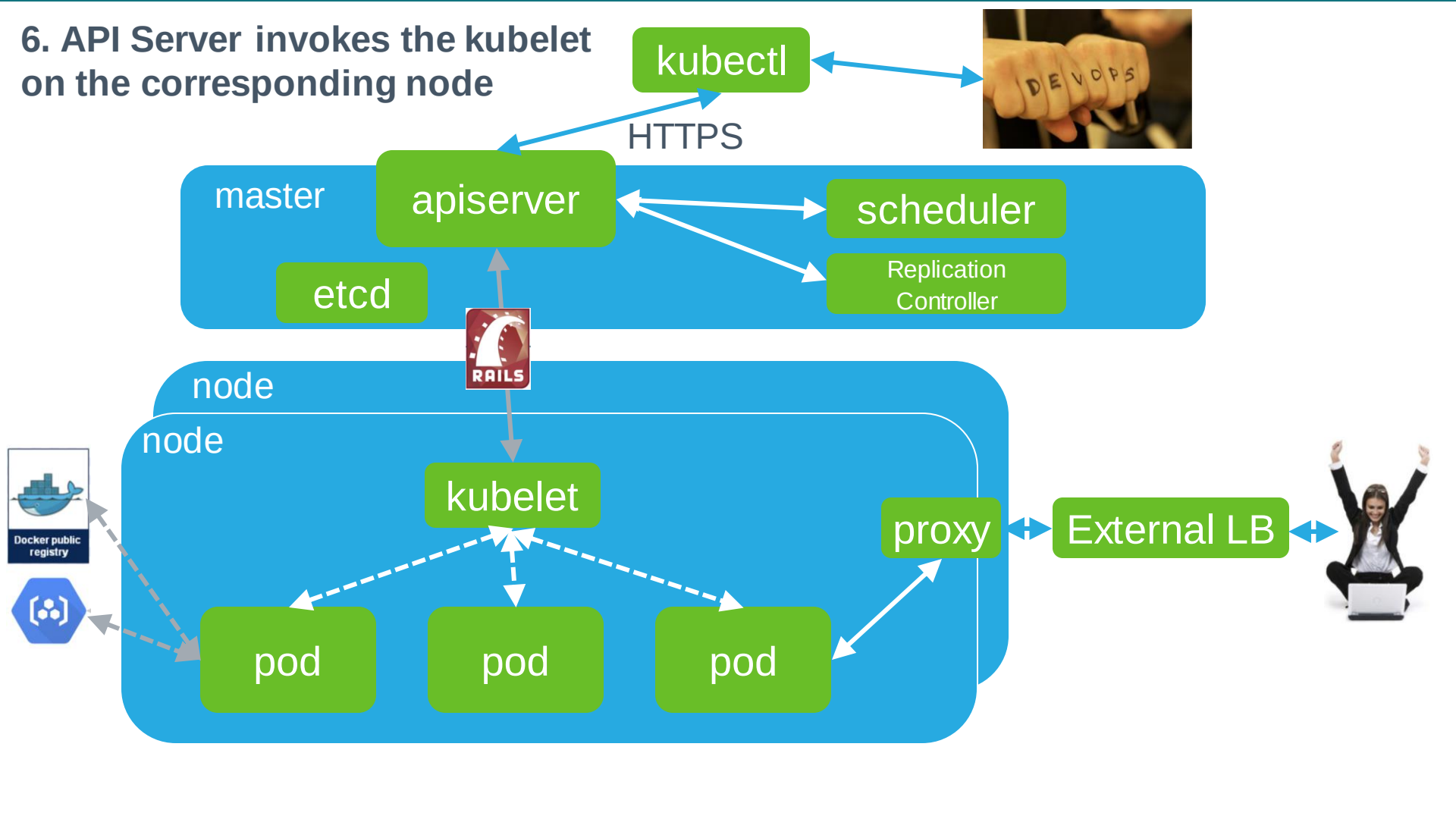
Docker public registry

pod

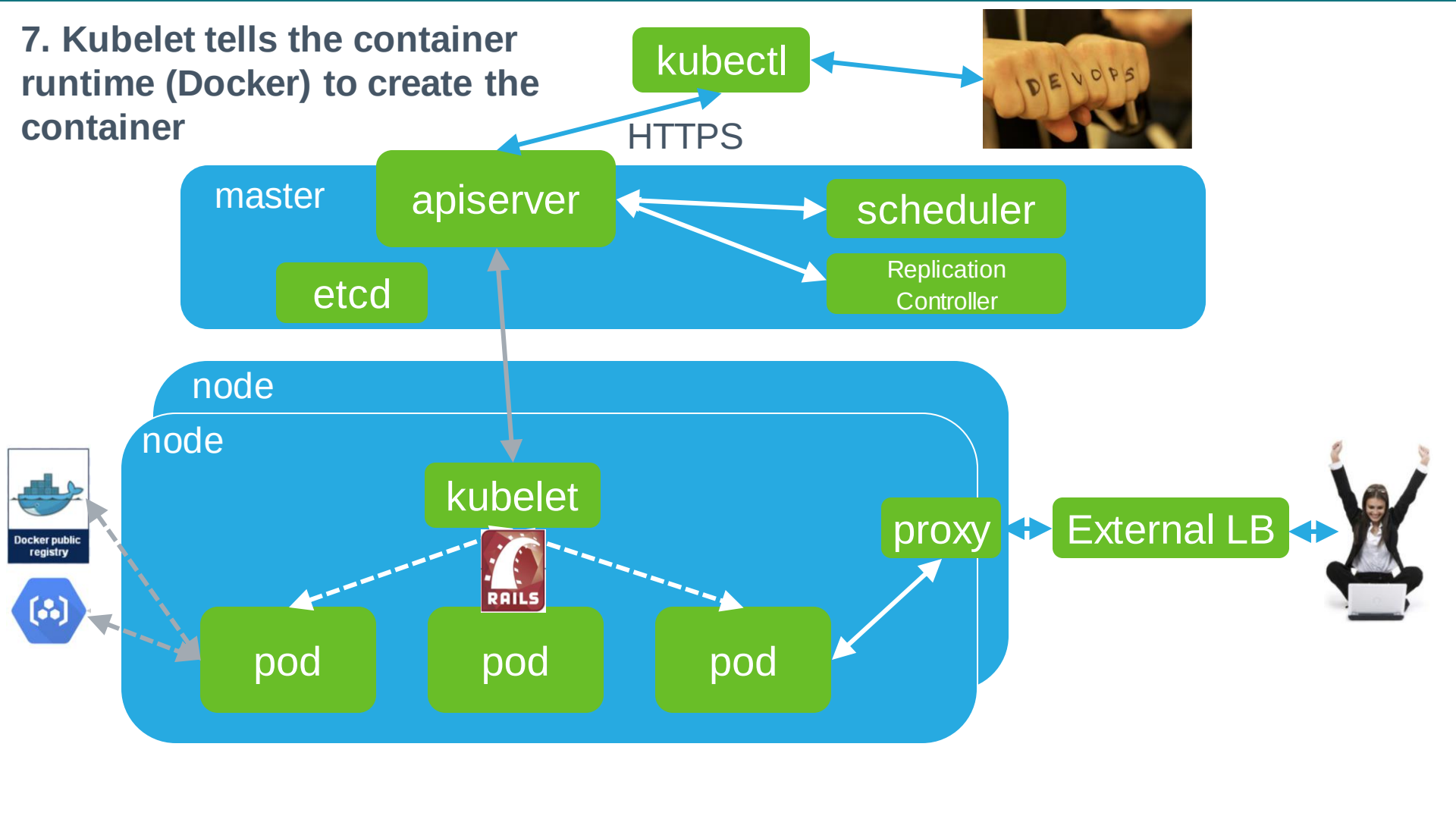
pod

pod

6. API Server invokes the kubelet on the corresponding node



7. Kubelet tells the container runtime (Docker) to create the container



8. The image is pulled from an external repo

kubectl

HTTPS

master

apiserver

scheduler

etcd

Replication Controller

node

node

kubelet

proxy

External LB

Docker public registry

RAILS

pod

pod

pod

9. Kubelet informs the API Server of the pod status

kubectl

HTTPS

master

apiserver

scheduler

etcd

Replication Controller

RAILS

node

node

kubelet

proxy

External LB

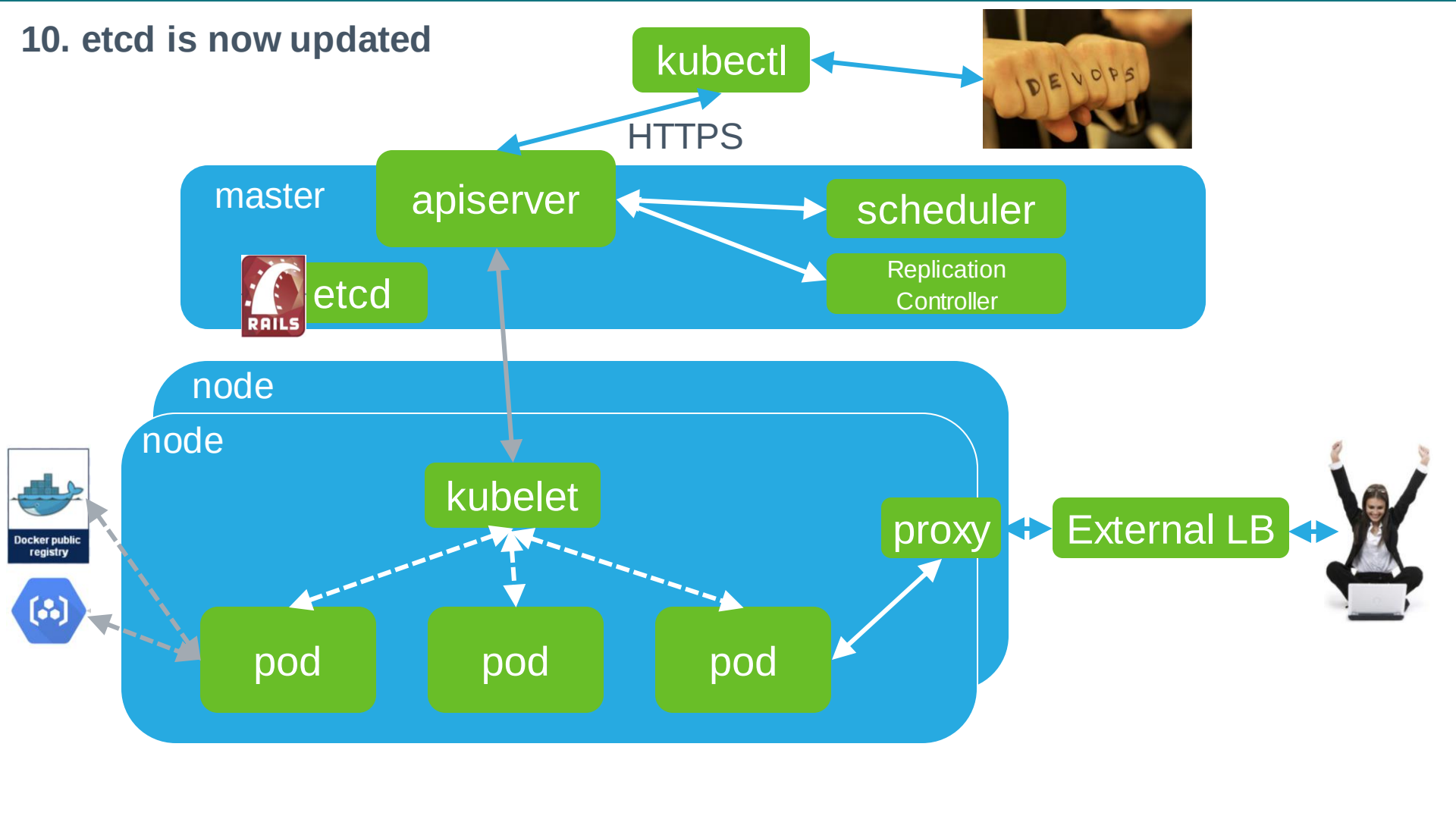
Docker public registry

pod

pod

pod

10. etcd is now updated



11. ReplicationController ensures desired state is reality

kubectl

HTTPS

master

apiserver

scheduler

etcd

Replication Controller

RAILS

node

node

kubelet

proxy

External LB

Docker public registry

pod

pod

pod

Lab 3

003-K8S-Cluster-Setup

Lab Goals:

- Launch API in cluster
- Exec to container
- Expose via LoadBalancer
- Using YAML manifests for deployment