Kubernetes 101

Python Girona - September 2019

Jordi Bagot (https://github.com/jbagot), Xavi Torelló (https://github.com/XaviTorello)

Remember

- Container
- Docker
- Docker file
- Docker build

You can check the docker101 slides:

https://github.com/pygrn/docker101/tree/a0d402917ec3fab8e8036a9d4cbf57f04c8929
(https://github.com/pygrn/docker101/tree/a0d402917ec3fab8e8036a9d4cbf57f04c8929

Curiosities of Kubernetes

- kubernetes = k(ubernete)s = k8s
- Created by Google in 2014, based on Borg (Google container orchestrator)
- It's open source
- Mainteined by the Cloud Native Computing Foundation

What's k8s?

- Container orchestrator
- The most famous one
- Layer to manage a cluster of containers
- Auto-scaling

Container orchestrator

- Easy deploy, gracefully, stateless
- Replication: run X copies
- Built-in load balancers
- Auto-scaling: better resource use

Layer to manage a cluster

- Manifest: one file to define all your cluster
- Automate cluster maintenance
- Load balancers
- Manage secrets

Scalability

- Specify number of container replicas
- Auto pod scaling, based on CPU, memory or custom metrics
- It's a cluster, add more nodes

Kubernetes structure

- Pod
- Service
- Namespace
- Node

Pod

- The smallest structure, minimal unit of deployment
- One or more containers
- One IP
- Stateless

How can we access the pods?

There are multiple of them, each one with one IP...

with Services

Service

- Group of pods
- Built-in load balancer
- Own static IP

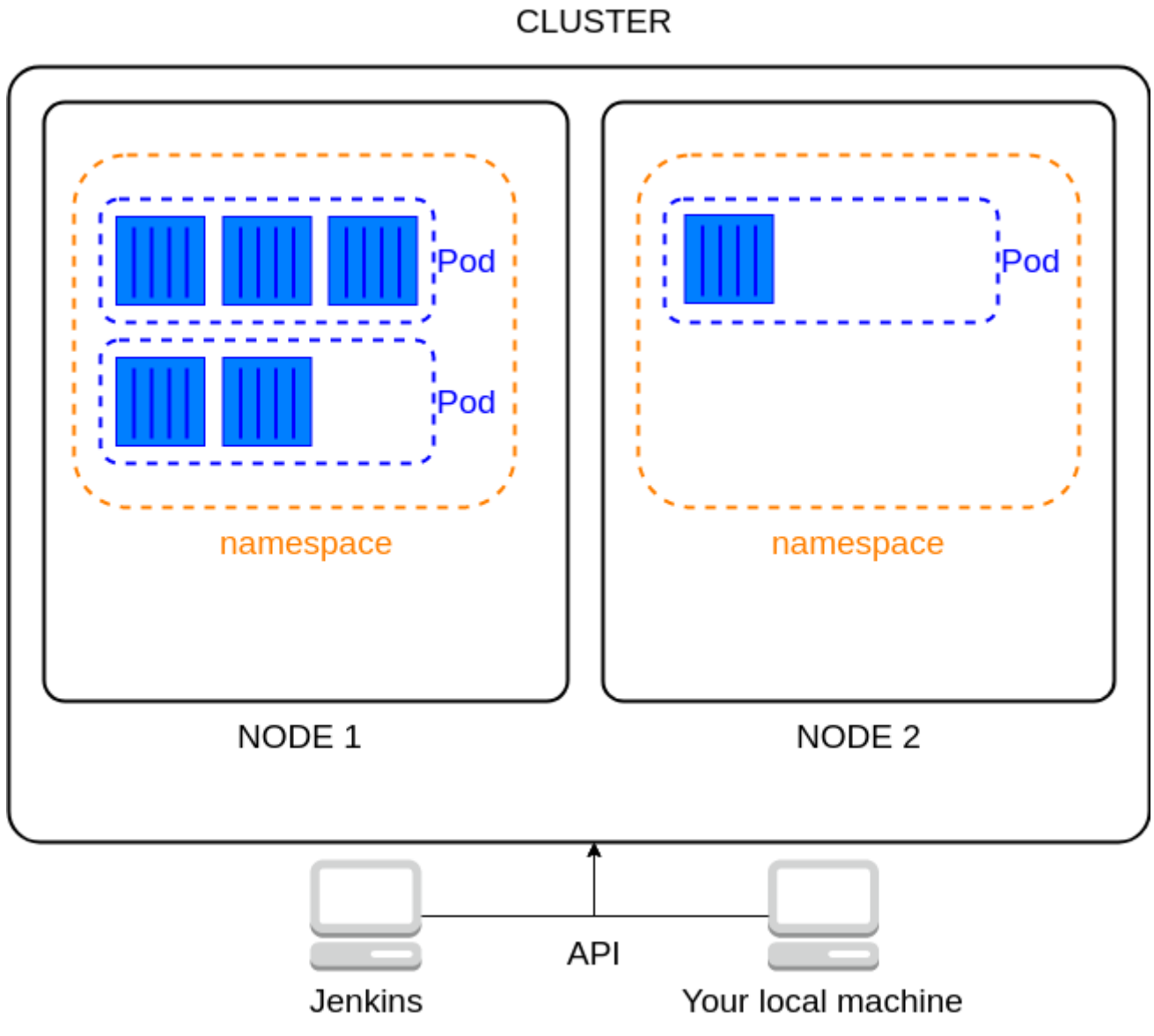
```
apiVersion: v1
kind: Service
metadata:
  name: docs
spec:
  ports:
  - port: 8000
    targetPort: 8000
  selector:
    app: docs
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: docs
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0
  selector:
    matchLabels:
      name: docs
  template:
    metadata:
      labels:
        name: docs
        app: docs
        hash: {{deploy_hash}}
    spec:
      containers:
      - name: docs
        image: {{doc_image}}:{{deploy_version}}
        imagePullPolicy: Always
        # defines the health checking
        ports:
          - containerPort: 8000
            name: docs
        readinessProbe:
```

Namespace

- Contain services, pods, network
- Isolated with other namespaces
- Is common to use it to separete environments

Node

- Hosts, servers
- Multiple nodes form the cluster


CLUSTER
Pod
Pod
namespace
Pod
namespace
NODE 1
NODE 2
API
Jenkins
Your local machine

Volume

- Stateful applications
- A volume is a mapped directory where the pods will point to get the data from it
- Persistent

```
kind: PersistanceVolume
apiVersion: v1
metadata:
  name: pv-name
  labels:
    type: local
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
    hostPath:
      path: "/tmp/pvs"
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: claim-name
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Liveness and readiness

- Same function as supervisor
- Check if the services are running
- Spin up new pods if the probes fail

DNS - Ingress

```
foo.bar.com -> 178.91.123.132 -> / foo    service1:4200
                                 / bar    service2:8080
```

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: service1
          servicePort: 4200
      - path: /bar
        backend:
          serviceName: service2
          servicePort: 8080
```

```
kubectl describe ingress simple-fanout-example
```

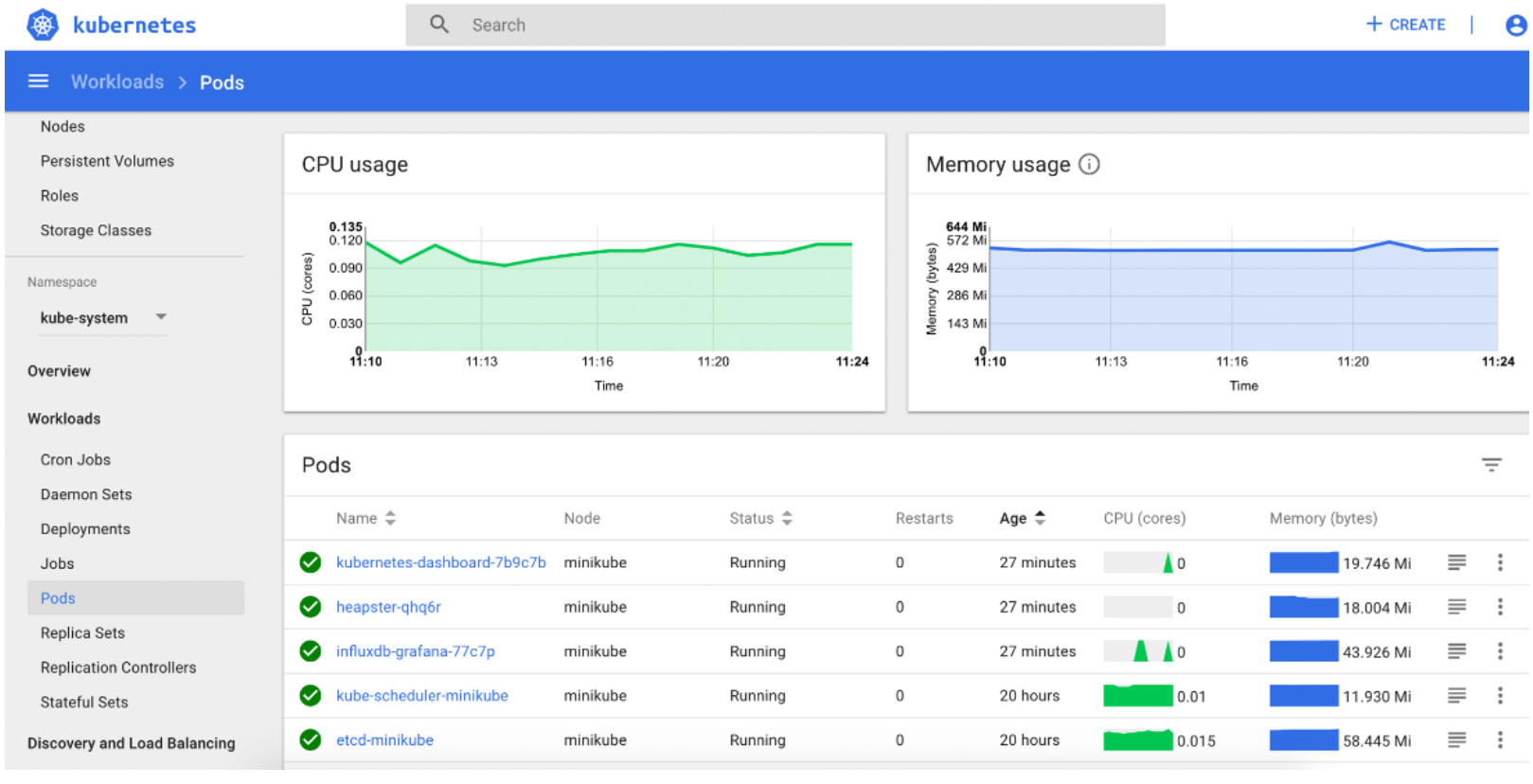
```
Name:             simple-fanout-example
Namespace:        default
Address:          178.91.123.132
Default backend:  default-http-backend:80 (10.8.2.3:8080)
Rules:
  Host         Path  Backends
  ----         ----  --------
  foo.bar.com
               /foo   service1:4200 (10.8.0.90:4200)
               /bar   service2:8080 (10.8.0.91:8080)
Annotations:
  nginx.ingress.kubernetes.io/rewrite-target:  /
Events:
  Type     Reason  Age                From                     Message
  ----     ------  ----               ----                     -------
  Normal   ADD     22s                loadbalancer-controller  default/test
```

kubectl

- API to connect and manage the cluster
- Used in local
- Used to deploy in the pipelines

WebUI

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta4/aio/deploy/recommended.yaml
```


kubernetes
Search
+ CREATE
Workloads > Pods
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
kube-system
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
CPU usage
0.135
0.120
0.090
0.060
0.030
0
CPU (cores)
11:10
11:13
11:16
11:20
11:24
Time
Memory usage
644 Mi
572 Mi
429 Mi
286 Mi
143 Mi
0
Memory (bytes)
11:10
11:13
11:16
11:20
11:24
Time
Pods
Name
Node
Status
Restarts
Age
CPU (cores)
Memory (bytes)
kubernetes-dashboard-7b9c7b minikube Running 0 27 minutes 0 19.746 Mi
heapster-qhq6r minikube Running 0 27 minutes 0 18.004 Mi
influxdb-grafana-77c7p minikube Running 0 27 minutes 0 43.926 Mi
kube-scheduler-minikube minikube Running 0 20 hours 0.01 11.930 Mi
etcd-minikube minikube Running 0 20 hours 0.015 58.445 Mi

helm

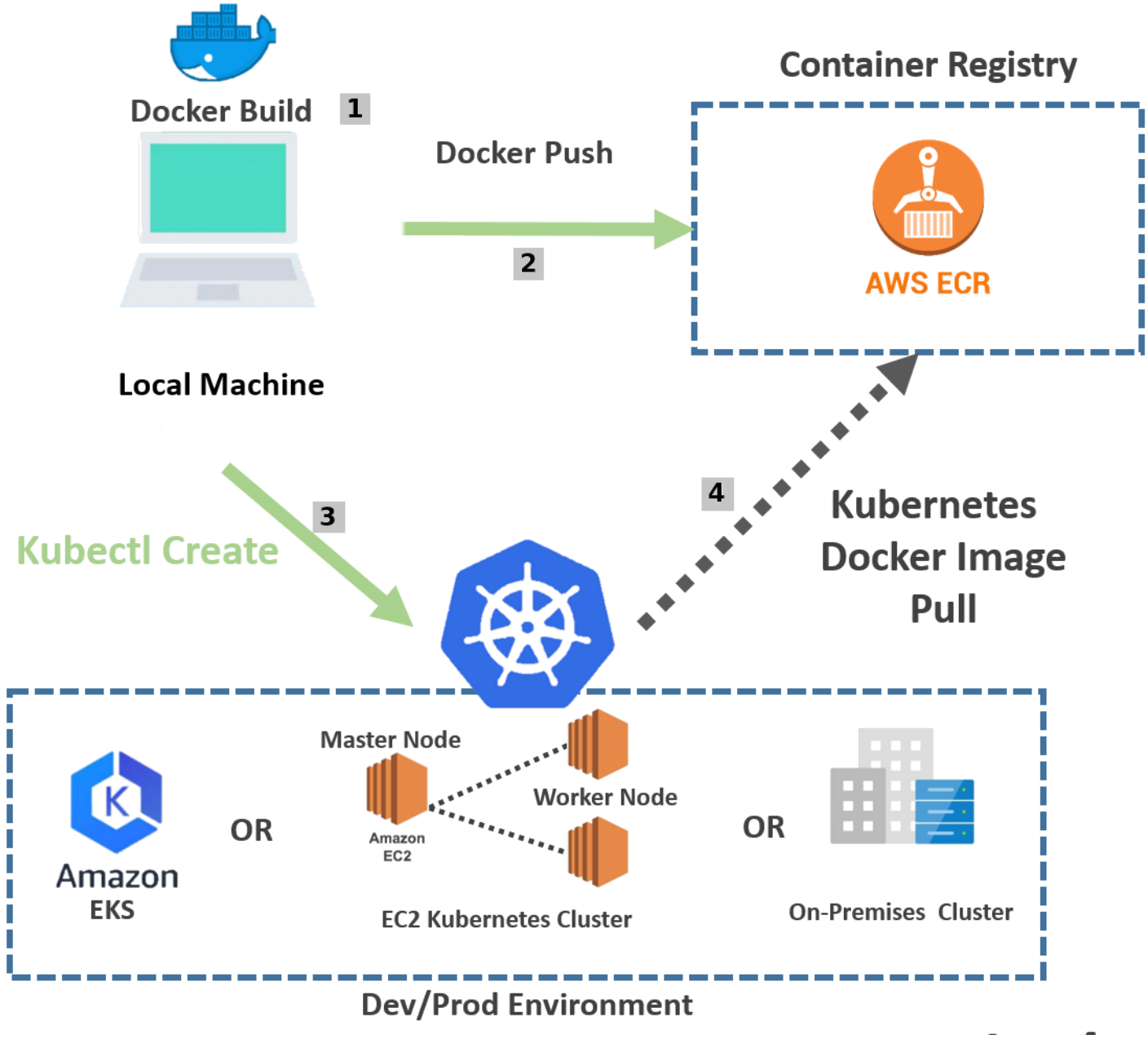
- Help manage k8s apps
- Charts
- Define, install, upgrade
- Share with the community

```
helm install stable/redis --values values-production.yaml
```

Deploy our app in k8s

- Direct deployment from local
- Deployment from CI/CD pipeline

Deploy from local


Container Registry
Docker Build
1
Docker Push
2
AWS ECR
Local Machine
3
Kubectl Create
4
Kubernetes
Docker Image
Pull
Master Node
Worker Node
Amazon
EC2
OR
OR
Amazon
EKS
EC2 Kubernetes Cluster
On-Premises Cluster
Dev/Prod Environment

Deploy from CI/CD