KUMULUS
TECHNOLOGIES

Basics of Kubernetes

Agenda

- An Overview of Kubernetes
- Getting an Environment Deployed - minikube
- Building an application
  - Pods
  - Deployments and Replication Sets (Controllers)
  - Services
- Advanced Topics
- Next Steps

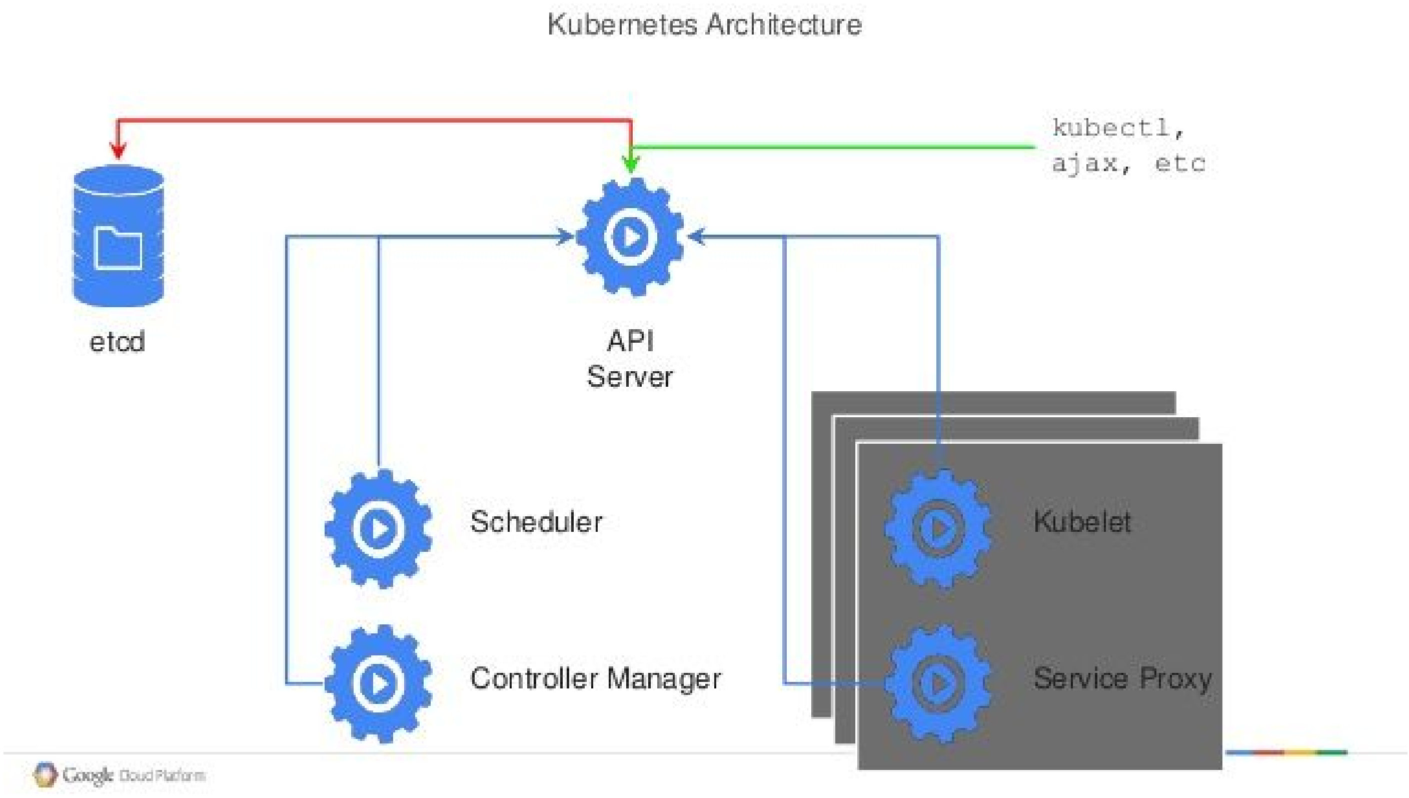
Who are we?

Robert Starmer: 🐦 @rstarmer
CTO/Founder of Kumulus Technologies
OpenStack Ops contributor since 2012
Supporting Cloud enablement for Enterprise
OpenStack, Kubernetes, BareMetal to App CD

Kumulus Technologies:🐦 @kumulustech
Systems consultants supporting cloud migration
Kumulus Tech Newsletter: https://kumul.us/newsletter/
Five Minutes of Cloud: https://youtube.com/fiveminutesofcloud

Kubernetes Overview

Kubernetes Architecture



Underlying Pieces

Containers on x86 compute (usually Linux)

Docker or rkt (beta) < which is really just how do you launch a runC container...

Storage

Persistence via network shared local storage or provider solution (GCE/AWS)

Network

Direct attachment or via Tunneling/NAT (iptables)

Kubernetes on OpenStack

OpenStack can provide all the necessary resources:

- Magnum or HEAT to deploy Docker and Kubernetes services
- Cinder and Manila to provide persistent storage
- Neutron + Kuryr to enable network services

Key Components

PODs

Grouping of Containers, Storage, and shared namespaces (e.g. IPC)

Replication Controller/Set

Scale management and Scheduling

Deployment

Lifecycle/Resiliancy management

Service

Loadbalancing/Discovery/Mapping management

Label

Key:Value pair that is used to select subsets of resources

Getting Kubernetes

VirtualBox

In order to deploy kubernetes on our laptops, we need a virtualization manager.

Install VirtualBox:

https://virtualbox.org/wiki/Downloads

It may be possible to use other virtualization layers, but VirtualBox appears to be the best solution for the minikube installation of Kubernetes (a single node installation for basic testing/understanding).

Minikube

Deploy minikube on your local machine. Minikube is an application that automatically deploys of a single node Kubernetes environment by createing a virtual machine in the VirtualBox virtualization layer.

http://kubernetes.io/docs/getting-started-guides/minikube/

- Download minikube app to your local laptop
- Download kubectl app to your local laptop
- Launch minikube -> minikube start
- Verify service with kubectl -> kubectl get nodes

Minikube quickstart

Start it

```
minikube start # launch the minikube application

minikube ip # check if minikube is running, find the IP for application access

minikube dashboard # launch the kubernetes UI in a web browser

kubectl get nodes # verfiy that kubectl CLI tool is installed and working
```

May need to set up docker access (to kill a container)

```
eval $(minikube docker-env)

docker ps
```

Deploy your first App

Launch a Pod

Code for these services are available here:

https://github.com/kumulustech/dev-kub100

Create a single container POD:

- Create Definition
- Launch container/pod with ‘kubectl create -f pod.yml’

What goes into a Pod specification

Example pod spec (pod.yml)

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    ports:
    - containerPort: 80
      protocol: TCP
```

Managing scale

Create a deployment, which automatically creates a replica set (new version of a replication controller)

```
  kubectl run nginx --image=nginx --port=80
```

Scale up to 3 instances

```
  kubectl edit #uses your default text editor to modify the deployment
configuration
  kubectl get deployment
```

Scale down to 2 instances

```
 kubectl scale --replicas=2 deploy/nginx
```

Updating a Pod to a Deployment

Deployments are part of the beta api:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        run: nginx
...
```

```
...
    spec:
      containers:
      - image: nginx
        name: nginx
        ports:
        - containerPort: 80
          protocol: TCP
```

Keeping it alive

Now that our deployment is running, we might want to see how it works, keeping containers alive for us.

Let’s see if we can kill a container:

```
eval $(minikube docker-env)

docker stop {container_id}; docker rm {container_id}

kubectl get deployment nginx
```

But how do I get to it? - Services

Create a service with a service spec, or use the “expose” command

```
kubectl expose deployment nginx --type=NodePort  --target-port=80
```

Now to learn what port that is:

```
kubectl describe svc nginx --output='jsonpath="{.spec.ports[0].NodePort}"'
```

With minikube:

```
minkube service nginx
```

Service Specification

The service specification can include ports and define the service access type (node, loadbalancer, etc.)

```
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  ports:
  - nodePort: 32500
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: NodePort
```

Putting it all together

One yml file to bind them all

```
kubectl get deployment nginx > nginx.yaml
echo '---' >> nginx.yml
kubectl get svc nginx >> nginx.yaml
```

Now that we have a description, (which we need to remove some of the state from) we can delete the current deployment and service. See the example on the right ->

```
kubectl delete deployment nginx
kubectl delete svc nginx
```

Then launch the service again:

```
kubectl create -f nginx.yml
```

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        run: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
        ports:
        - containerPort: 80
          protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  ports:
  - nodePort: 32500
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: NodePort
```

Advanced Kubernetes

Logs and Exec

When containers fail, often logs can provide a first level of feedback. Getting logs from Kubernetes requires understanding names/services.

```
  kubectl get pods
  kubectl logs nginx-{dep}-{hash}
```

But sometimes that's not enough, and you'd like to actually look at the internal state of the container.

```
  kubectl exec -ti nginx-{dep}-{hash} -- bash
```

NOTE: This assumes your container has a bash executable included in the image!

Namespaces (kubernetes)

Inintial “tenant” segregation, provides a way to limit the scope of queries and commands

Not true “multi-tenancy”. Users can still querry “all”

Get namespaces

```
kubectl get ns
kubectl create namespace nginx
```

Set namespace (minicube context)

```
kubectl config set-context $CONTEXT --namespace=nginx
```

Next Steps

Tutorials and docs on Kubernetes.org

http://kubernetes.io/

http://kubernetes.io/docs/tutorials/kubernetes-basics/ <similar to this class

https://www.digitalocean.com/community/tutorials/an-introduction-to-kubernetes

Take a more complete class

Kumulus Technologies - Kubernetes Fundamentals - Dev-101

https://kumul.us/kub100

For the class, there’s a special discount code too: **SFBOS-KUB050**

50% off our upcoming class for the next 2 weeks (Until midnight PST December 15)