Kubernetes 101

Tina Coleman

Docker (*)

- Isolated “containerized” process, with all of its dependencies included
- Communications into container through explicitly exposed ports, drive mappings, etc.
- Deployable across any Docker-supported environment - build on dev box, deploy to prod environment
- (*) = Kubernetes aims to support other container runtimes, in addition to Docker

“The Cloud”

- Multiple servers
- Do not have to be homogenous (specifications, platform, operating system, ...)
- Does not have to be static
- Examples: Amazon AWS, Microsoft Azure, Google Cloud Platform, on-premises cluster

Orchestrating Containers

- How to get those docker images out on “the cloud”?
- When cloud nodes die, how to move those docker images and their processing?
- If demand surges or collapses, can we use the machines more efficiently?

Kubernetes

Kubernetes =
Docker + “the cloud” + orchestration

The so what:

- Predictable isolated processes
- Able to take advantage of full capacity of cloud
- In the cloud’s state today and as its load or available server capacity changes
-

History

- Google develops Borg for use as in-house container scheduler
- Project Seven == friendlier Borg (Star Trek reference)
- Seven becomes Kubernetes (k8s), announced in 2014
- Remember Pokemon Go?
- Now managed by Cloud Native Foundation
- Built into RedHat OpenShift, Rancher, CoreOS Tectonic, and others
- Google Container Engine == SaaS implementation

Use by our customer

....

Key Concepts

Images credit to “The Children’s Illustrated Guide to Kubernetes”, https://deis.com/blog/2016/kubernetes-illustrated-guide/

Interacting

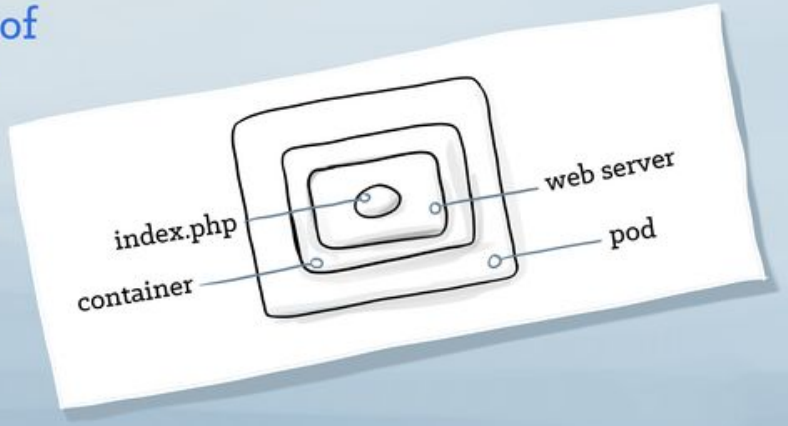
- Via a RESTful API
  - Directly
  - Kubectl
- Requires login, which grants a token
  - Users can be confined to certain namespaces, certain operations, quotas, ...
- Always remote
  - “K8s, go use this docker container and give it this data and mount in these files” - describe a desired state
  - K8s then makes it so or tells you it can’t

Kubernetes Uses Labels

- Can query based on these labels

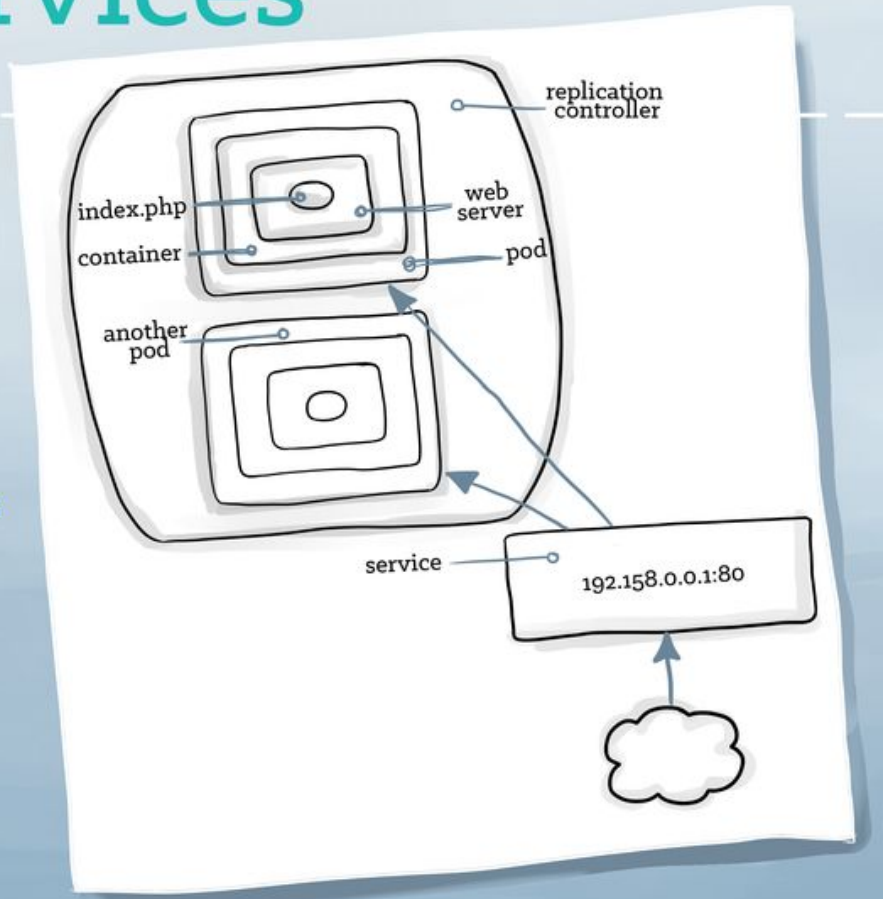
Pods

- A pod can hold any number of containers, but usually only holds two
- We pretend one of those containers doesn't exist
- A pod is connected via an overlay network to the rest of the environment



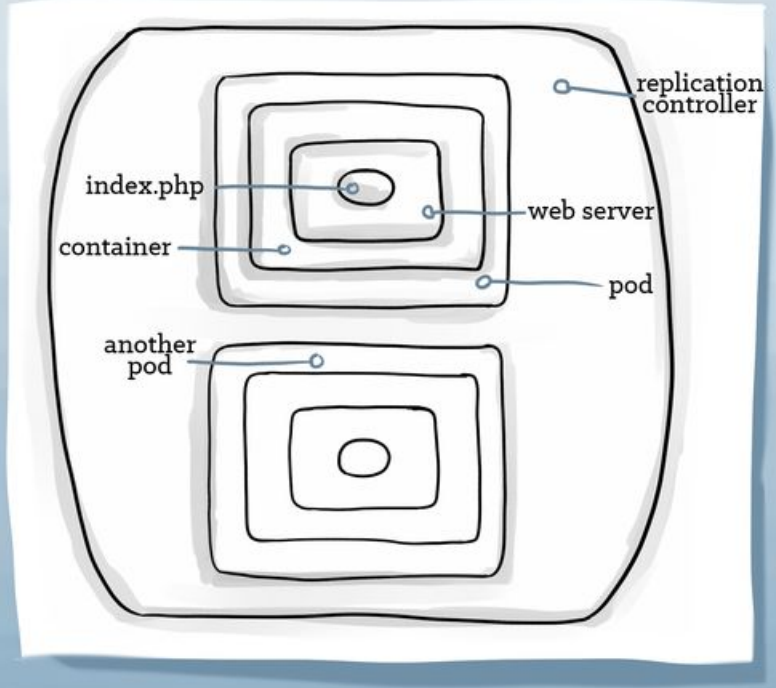
Services

- Persistent
- Provide discovery
- Provide load balancing
- Provide stable service address
- Find pods by label selector



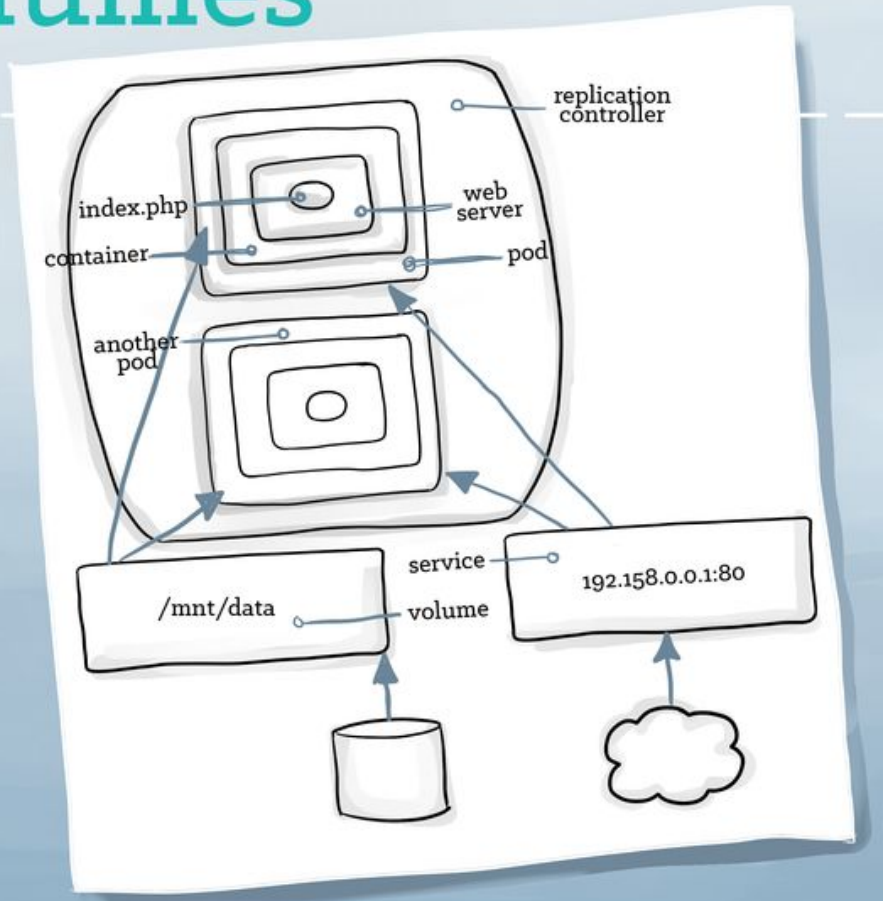
Replication Controllers

- Have a *pod template* for creating any number of pod copies
- Provide logic for scaling the pod up or down
- Can be used for rolling deploys



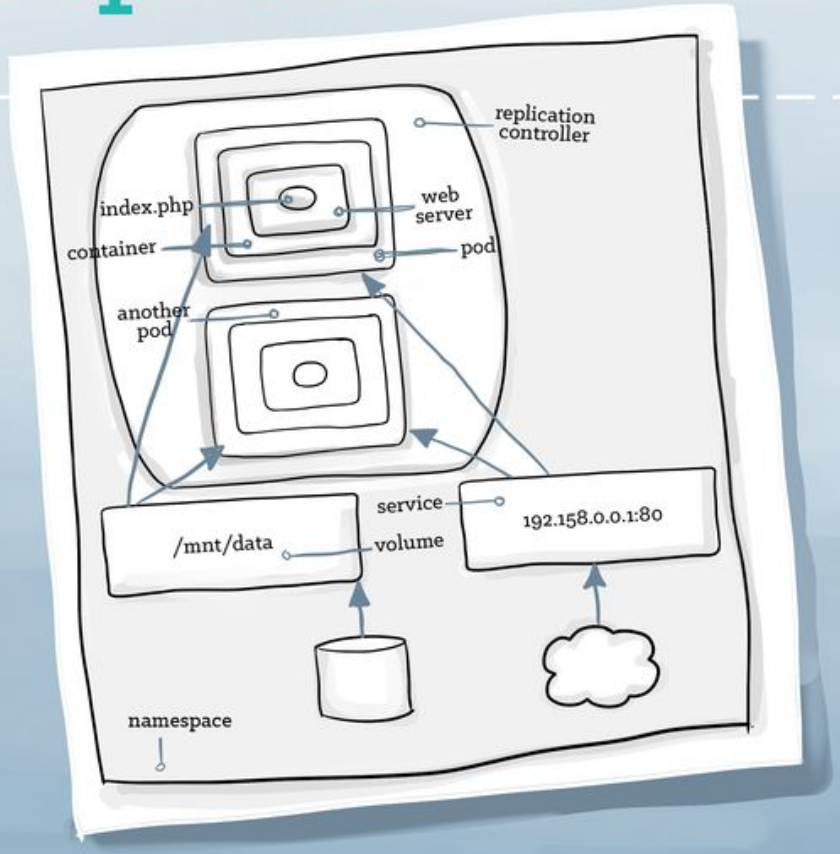
Volumes

- Providers expose both persistent and ephmeral storage
  - › EBS
  - › Ceph
  - › Gluster...
- Pods can mount volumes like filesystems



Namespaces

- Group + segment pods, rcs, volumes,& secrets from each other



Playing Along

Getting Started

- Use someone's existing cluster or hosted solution
  - Google Container Engine (GKE)
  - OpenShift.io
  - … (available clusters within work environment)
- Minikube (single machine instance)
- Kubeadm "to easily bring up a cluster with a single command per machine"

Getting started - even easier

Do the Kubernetes tutorials at
https://kubernetes.io/docs/tutorials/

Minikube

- Instructions at https://github.com/kubernetes/minikube
  - Install VirtualBox (or other options)
  - Download installer: I used `` `brew cask install minikube` ``
  - This will also install kubectl, as well as the bash completions for kubectl
- `` `minikube start` ``: now have a kubernetes “cluster” running with kubectl configured to use it
- `eval $(minikube docker-env)`
  - Sets up Docker to interact with the daemon within your minikube
  - Meaning, builds/lists your docker images based on its cache

Deploying an app

What the tutorials show you as the starting point:

```
kubectl run node-example --image=k8s-node-example:1.0
--port=8080
```

What happened?

Look at the results

1) `kubectl get deployments:` created a “deployment” using the name given in the run
2) `kubectl describe deployment node-example:` events include ‘scaled up replicaset’
3) `kubectl get replicasets`
4) `kubectl get pods`
5) `kubectl describe pod ...`

How it's really usually done...

First, look at what k8s is trying to maintain:

```
kubectl get deployment node-example -o yaml
```

Now, look at deployment-example.yaml - declarative configuration

Tell k8s to use it:

```
kubectl create -f deployment-example.yaml --record
```

Expose that bit of code

Right now, it’s accessible within k8s network in your “namespace”, but not external.

```
kubectl create -f service-name.yaml
```

Testing service selectors

```
kubectl get pods --show-labels

kubectl get pods -l 'app=tutorial-app'

kubectl get pods -l 'app!=tutorial-app'
```

Service oughta-knows

- The default type of service is ClusterIP: make me available within the cluster
- Addressable via service-name: e.g. node-service:8080
- We used type: NodePort, which says, allocate a port on the k8s cluster external fabric
  - `kubectl get service node-example`
  - Use indicated port on address of cluster
-

Scaling up a deployment

Let’s take a look at current state:

```
kubectl get pods -o wide -l 'app=tutorial-app'
```

Now, let’s add more pods:

```
kubectl scale deployment/node-deployment
--replicas=7
```

Scaling down works, too (skip for demo)

```
kubectl scale deployment/node-deployment
--replicas=0
```

New code… Updating a deployment

```
kubectl set image deployment/node-deployment
node=k8s-node-example:2.0

kubectl describe deployment node-deployment

kubectl get pods
kubectl logs …
```

Uh oh… Rolling back a deployment

```
kubectl rollout history deployment/node-deployment

kubectl rollout undo deployment/node-deployment

kubectl describe deployment node-deployment
```

Check image version

Check events

Ok, fix the issue

```
kubectl set image deployment/node-deployment
node=k8s-node-example:3.0
```

Things we didn't cover

More things to take advantage of...

Configuring pods

- ConfigMaps
- Secrets

Mounting data in via volumes, persistent volumes

Jobs, CronJobs

Node affinity (select compute nodes that meet right conditions)

...

Useful complementary tools

Helm - keep configuration information out of yaml; readily deploy multiple clones

Kompose - convert docker-compose files to kubernetes manifests