Docker and Kubernetes Training

Kubectl commands usage

Running Containers with Docker

docker run -d --name kuard --publish 8080:8080 gcr.io/kuar-demo/kuard-amd64:1

Limiting memory resources

docker run -d --name kuard --publish 8080:8080 --memory 200m --memory-swap 1G gcr.io/kuar-demo/kuard-amd64:1

Limiting CPU resources

docker run -d --name kuard --publish 8080:8080 --memory 200m --memory-swap 1G --cpu-shares 1024 gcr.io/kuar-demo/kuard-amd64:1

Google Container Service

gcloud config set compute/zone us-west1-a

gcloud container clusters create kuar-cluster

gcloud auth application-default login

Installing Kubernetes with Azure Container Service

az group create --name=kuar --location=westus

az acs create --orchestrator-type=kubernetes --resource-group=kuar --name=kuar-cluster
az acs kubernetes get-credentials --resource-group=kuar --name=kuar-cluster

az acs kubernetes install-cli

Checking Cluster Status

kubectl version

kubectl get componentstatuses

Listing Kubernetes Worker Nodes

kubectl get nodes

kubectl describe nodes node-1

Kubernetes Proxy

kubectl get daemonSets --namespace=kube-system kube-proxy

Kubernetes DNS

kubectl get deployments --namespace=kube-system kube-dns

kubectl get services --namespace=kube-system kube-dns

Kubernetes UI

kubectl get deployments --namespace=kube-system kubernetes-dashboard

kubectl get services --namespace=kube-system kubernetes-dashboard

kubectl proxy

http://localhost:8001/ui

Contexts

kubectl config set-context my-context --namespace=mystuff

kubectl config use-context my-context

Viewing Kubernetes API Objects

kubectl get <resource-name> <object-name>

kubectl get pods my-pod -o jsonpath --template={.status.podIP}

kubectl describe <*resource-name*> <*obj-name*>

Creating, Updating, and Destroying Kubernetes Objects

kubectl apply -f obj.yaml

kubectl edit <*resource-name*> <*obj-name*>

kubectl delete -f obj.yaml

kubectl delete <*resource-name*> <*obj-name*>

Labeling and Annotating Objects

kubectl label pods bar color=red

kubectl label pods bar -color

Debugging Commands

kubectl logs <*pod-name*>

kubectl exec -it <*pod-name*> -- bash

kubectl cp <*pod-name*>:*/path/to/remote/file /path/to/local/file*

Creating a Pod

kubectl run kuard --image=gcr.io/kuar-demo/kuard-amd64:1

kubectl get pods

kubectl delete deployments/kuard

Creating a Pod Manifest

docker run -d --name kuard --publish 8080:8080 gcr.io/kuar-demo/kuard-amd64:1
kuard-pod.yaml

Running Pods

$ **kubectl apply -f kuard-pod.yaml**

Listing Pods

$ **kubectl get pods**

Pod Details

$ **kubectl describe pods kuard**

Deleting a Pod

$ **kubectl delete pods/kuard**

$ **kubectl delete -f kuard-pod.yaml**

Using Port Forwarding

$ **kubectl port-forward kuard 8080:8080**

Getting More Info with Logs

$ **kubectl logs kuard**

Running Commands in Your Container with exec

$ **kubectl exec kuard date**

$ **kubectl exec -it kuard ash**

Copying Files to and from Containers

$ **kubectl cp <*pod-name*>:/captures/capture3.txt ./capture3.txt**

$ **kubectl cp $HOME/config.txt <*pod-name*>:/config.txt**

Liveness Probe

kuard-pod-health.yaml

$ **kubectl apply -f kuard-pod-health.yaml**
$ **kubectl port-forward kuard 8080:8080**

kubectl describe kuard

Resource Requests: Minimum Required Resources

kuard-pod-resreq.yaml

Capping Resource Usage with Limits

kuard-pod-reslim.yaml

Using Volumes with Pods

kuard-pod-vol.yaml

Labels

- key/value pairs that can be attached to Kubernetes objects such as Pods and ReplicaSets.
- They can be arbitrary, and are useful for attaching identifying information to Kubernetes objects.
- Labels provide the foundation for grouping objects.

Annotations

Provides a storage mechanism that resembles labels: annotations are key/value pairs designed to hold nonidentifying information that can be leveraged by tools and libraries.

Applying Labels

$ **kubectl run alpaca-prod --image=gcr.io/kuar-demo/kuard-amd64:1 --replicas=2 --labels="ver=1,app=alpaca,env=prod"**

```
$ kubectl run alpaca-test --image=gcr.io/kuar-demo/kuard-amd64:2
--replicas=1 --labels="ver=2,app=alpaca,env=test"

$ kubectl run bandicoot-prod --image=gcr.io/kuar-demo/kuard-
amd64:2 --replicas=2 --labels="ver=2,app=bandicoot,env=prod"

$ kubectl run bandicoot-staging --image=gcr.io/kuar-demo/kuard-
amd64:2 --replicas=1 --labels="ver=2,app=bandicoot,env=staging"


$ kubectl get deployments --show-labels
```

Modifying Labels

```
$ kubectl label deployments alpaca-test "canary=true"

$ kubectl get deployments -L canary

$ kubectl label deployments alpaca-test "canary-"
```

Label Selectors

```
$ kubectl get pods --show-labels

$ kubectl get pods --selector="ver=2"

$ kubectl get pods --selector="app=bandicoot,ver=2"

$ kubectl get pods --selector="app in (alpaca,bandicoot)"

$ kubectl get deployments --selector="canary"
```

Operator Description

key=value	key is set to value
key!=value	key is not set to value
key in (value1, value2)	key is one of value1 or value2
key notin (value1, value2)	key is not one of value1 or value2

key	key is set
!key	key is not set

Label Selectors in API Objects

```
selector:
  matchLabels:
     app: alpaca
  matchExpressions:
     - {key: ver, operator: In, values: [1, 2]}

selector:
  app: alpaca
  ver: 1
```

Annotations

```
metadata:
  annotations:
     example.com/icon-url: https://example.com/icon.png
```

Cleanup

kubectl delete deployments –all

The Service Object

kubectl run alpaca-prod --image=gcr.io/kuar-demo/kuard-amd64:1 --replicas=3 --port=8080 --labels="ver=1,app=alpaca,env=prod"

kubectl expose deployment alpaca-prod

**kubectl run bandicoot-prod --image=gcr.io/kuar-demo/kuard-amd64:2
--replicas=2 --port=8080 --labels="ver=2,app=bandicoot,env=prod"**

```
kubectl expose deployment bandicoot-

$ALPACA_POD=$(kubectl get pods -l app=alpaca -o
jsonpath='{.items[0].metadata.name}')

kubectl port-forward $ALPACA_POD 48858:8080
```

Readiness Checks

```
$ kubectl edit deployment/alpaca-prod
```

```
readinessProbe:
  httpGet:
    path: /ready
    port: 8080
  periodSeconds: 2
  initialDelaySeconds: 0
  failureThreshold: 3
  successThreshold: 1
```

```
ALPACA_POD=$(kubectl get pods -l app=alpaca -o
jsonpath='{.items[0].metadata.name}')

kubectl port-forward $ALPACA_POD 48858:8080

kubectl get endpoints alpaca-prod –watch

kubectl edit service alpaca-prod

kubectl describe service alpaca-prod

ssh <node> -L 8080:localhost:32711

gcloud compute ssh <node> --zone <zone>

kubectl describe service alpaca-prod
```

Endpoints

```
kubectl describe endpoints alpaca-prod

kubectl get endpoints alpaca-prod –watch
```

kubectl delete deployment alpaca-prod

kubectl run alpaca-prod --image=gcr.io/kuar-demo/kuard-amd64:1 --replicas=3 --port=8080 --labels="ver=1,app=alpaca,env=prod"

Manual Service Discovery

kubectl get pods -o wide --show-labels

kubectl get pods -o wide --selector=app=alpaca,env=prod

BANDICOOT_POD=$(kubectl get pods -l app=bandicoot -o jsonpath='{.items[0].metadata.name}')

kubectl port-forward $BANDICOOT_POD 48858:8080

kubectl delete services,deployments -l app

Inspecting a ReplicaSet

kubectl describe rs kuard

Finding a ReplicaSet from a Pod

kubectl get pods <*pod-name*> -o yaml

Finding a Set of Pods for a ReplicaSet

kubectl get pods -l app=kuard,version=2

Imperative Scaling with kubectl Scale

kubectl scale kuard --replicas=4

Declaratively Scaling with kubectl apply

```
spec:
  replicas: 3
```

Autoscaling a ReplicaSet

```
kubectl autoscale rs kuard --min=2 --max=5 --cpu-percent=80

kubectl get hpa
```

Deleting ReplicaSets

```
kubectl delete rs kuard

kubectl get pods

kubectl delete rs kuard --cascade=false
```

DaemonSet

```
kubectl describe daemonset fluentd

kubectl get pods -o wide
```

Adding Labels to Nodes

```
kubectl label nodes k0-default-pool-35609c18-z7tb ssd=true

kubectl get nodes

kubectl get nodes --selector ssd=true
```

Updating a DaemonSet by Deleting Individual Pods

```
PODS=$(kubectl get pods -o jsonpath -
template='{.items[*].metadata.name}'
for x in $PODS; do
kubectl delete pods ${x}
sleep 60
done
```

Job Patterns

Type	Use Case	Behavior	completions	parallelism
One shot	Database migrations	A single pod running once until successful termination	1	1
Parallel fixed completions	Multiple pods processing a set of work in parallel	One or more pods running one or more times until reaching a fixed completion count	1+	1+
Work queue: parallel jobs	Multiple pods processing from a centralized work queue	One or more pods running once until successful termination	1	2+

One Shot

```
kubectl run -i oneshot --image=gcr.io/kuar-demo/kuard-amd64:1 --
restart=OnFailure -- --keygen-enable --keygen-exit-on-complete --
keygen-num-to-gen 10
```

```
kubectl delete jobs oneshot
```

```
kubectl apply -f job-oneshot.yaml
```

```
kubectl describe jobs oneshot
```

```
kubectl logs oneshot-4kfdt
```

job-oneshot-failure1.yaml

```
$ kubectl get pod -a -l job-name=oneshot
```

$ **kubectl get pod -l job-name=oneshot -a**
kubectl delete jobs oneshot

Parallelism

job-parallel.yaml

kubectl apply -f job-parallel.yaml
kubectl get pods -w
kubectl delete job parallel

Work Queues

Starting a work queue

rs-queue.yaml

kubectl apply -f rs-queue.yaml

QUEUE_POD=$(kubectl get pods -l app=work-queue,component=queue -o jsonpath='{.items[0].metadata.name}')

kubectl port-forward $QUEUE_POD 8080:8080

service-queue.yaml

kubectl apply -f service-queue.yaml

Loading up the queue

load-queue.sh

curl 127.0.0.1:8080/memq/server/stats

Creating the consumer job

job-consumers.yaml

kubectl apply -f job-consumers.yaml

kubectl get pods

kubectl delete rs,svc,job -l topic=jobs

Creating ConfigMaps

kubectl create configmap my-config --from-file=my-config.txt --from-literal=extra-param=extra-value --from-literal=another-param=another-value

kubectl get configmaps my-config -o yaml

Using a ConfigMap

kuard-config.yaml

kubectl apply -f kuard-config.yaml

kubectl port-forward kuard-config 8080

Creating Secrets

curl -O https://storage.googleapis.com/kuar-demo/kuard.crt
curl -O https://storage.googleapis.com/kuar-demo/kuard.key

kubectl create secret generic kuard-tls --from-file=kuard.crt --from-file=kuard.key

kubectl describe secrets kuard-tls

Secrets volumes

kuard-secret.yaml

kubectl apply -f kuard-secret.yaml

kubectl port-forward kuard-tls 8443:8443

Private Docker Registries

kubectl create secret docker-registry my-image-pull-secret --docker-username=<*username*> --docker-password=<*password*> --docker-email=<*email-address*>

kuard-secret-ips.yaml

Naming Constraints

- They may begin with a dot followed by a letter or number. Following characters include dots, dashes, and underscores.
- Dots cannot be repeated and dots and underscores or dashes cannot be adjacent to each other.
- More formally, this means that they must conform to the regular
- expression [.]?[a-zA-Z0-9]([.]?[-_a-zA-Z0-9]*[a-zA-Z0-9])*.

Valid key name	Invalid key name
.auth_token	Token..properties
Key.pem	auth file.json
config_file	_password.txt

kubectl get secrets

kubectl get configmaps

kubectl describe configmap my-config

kubectl get configmap my-config -o yaml

kubectl get secret kuard-tls -o yaml

Deployment

kubectl run nginx --image=nginx:1.7.12

kubectl get deployments nginx

```
kubectl get deployments nginx -o jsonpath --template
{.spec.selector.matchLabels}
```

```
kubectl get replicasets --selector=run=nginx
```

```
kubectl scale deployments nginx --replicas=2
```

```
kubectl get replicasets --selector=run=nginx
```

```
kubectl scale replicasets nginx-1128242161 --replicas=1
```

```
kubectl get replicasets --selector=run=nginx
```

Creating Deployments

```
kubectl get deployments nginx --export -o yaml > nginx-
deployment.yaml
```

```
kubectl replace -f nginx-deployment.yaml --save-config
```

Managing Deployments

```
kubectl describe deployments nginx
```

Scaling a Deployment

```
spec:
  replicas: 3
```

```
kubectl apply -f nginx-deployment.yaml
```

```
kubectl get deployments nginx
```

Updating a Container Image

```
containers:
  - image: nginx:1.9.10
      imagePullPolicy: Always
```

```
spec:
  ...
  template:
    annotations:
      kubernetes.io/change-cause: "Update nginx to 1.9.10"
```

kubectl apply -f nginx-deployment.yaml

kubectl rollout status deployments nginx

kubectl get replicasets -o wide

kubectl rollout pause deployments nginx

kubectl rollout resume deployments nginx

Rollout History

kubectl rollout history deployment nginx

kubectl rollout history deployment nginx --revision=2

kubectl rollout history deployment nginx

kubectl rollout undo deployments nginx

kubectl get replicasets -o wide

kubectl rollout history deployment nginx

kubectl rollout undo deployments nginx --to-revision=3