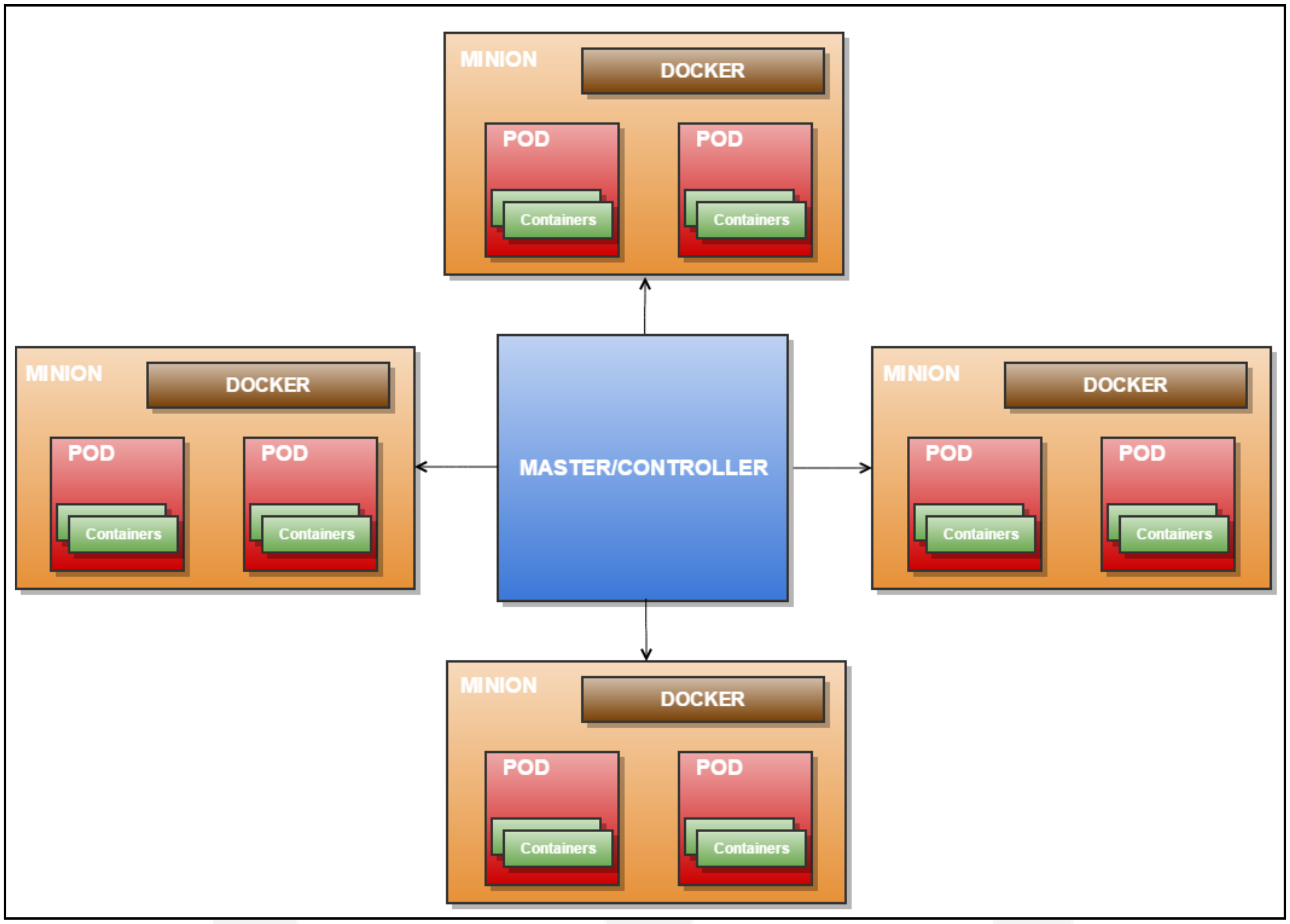
Linux Academy

Study Guide

Kubernetes Cheat Sheet

Contents

Architecture Diagram - High Level



Prerequisites (CentOS 7)

Master Controller

Packages

- `ntpd`
- `etcd`
- `kubernetes`

Notes

- Install from official repository

- Add the following to */etc/yum.repos.d/virt7-docker-common-release.repo*; create the file, if necessary

```
[virt7-docker-common-release]
name=virt7-docker-common-release
baseurl=http://cbs.centos.org.repos/virt7-docker-common-release/x86_64/
os/
Host Names
```

- Be sure you can resolve all names and IPs in your environment
- If not, add hosts (or aliases like *centos-master* or *centos-minion1* to */etc/hosts*)
- Ensure ports 8080, 2439 are open between all hosts in environment

Minions/Nodes

Packages

- `ntpd`
- `etcd`
- `kubernetes`
- `docker`

Notes

- Install from official repository
- Add the following to */etc/yum.repos.d/virt7-docker-common-release.repo*; create the file, if necessary

```
[virt7-docker-common-release]
name=virt7-docker-common-release
baseurl=http://cbs.centos.org.repos/virt7-docker-common-release/x86_64/
os/
Host Names
```

- Be sure you can resolve all names and IPs in your environment
- If not, add hosts (or aliases like *centos-master* or *centos-minion1* to */etc/hosts*)
- Ensure ports 8080, 2439 are open between all hosts in the environment

Installation

Master Controller

Configuration Files

/etc/kubernetes/config

```
# Comma separated list of nodes in the etcd cluster
KUBE_ETCD_SERVERS="--etcd-servers=http://centos-master:2379"
# logging to stderr means we get it in the systemd journal
KUBE_LOGTOSTDERR="--logtostderr=true"
# journal message level, 0 is debug
KUBE_LOG_LEVEL="--v=0"
# Should this cluster be allowed to run privileged docker containers
KUBE_ALLOW_PRIV="--allow-privileged=false"
# How the replication controller and scheduler find the kube-apiserver
KUBE_MASTER="--master=http://centos-master:8080"
```

/etc/etcd/etcd.conf

```
# [member]
ETCD_NAME=default
ETCD_DATA_DIR="/var/lib/etcd/default.etcd"
ETCD_LISTEN_CLIENT_URLS="http://0.0.0.0:2379"
# [cluster]
ETCD_ADVERTISE_CLIENT_URLS="http://0.0.0.0:2379"
```

/etc/kubernetes/apiserver

```
# The address on the local server to listen to.
KUBE_API_ADDRESS="--address=0.0.0.0"
# The port on the local server to listen on.
KUBE_API_PORT="--port=8080"
# Port kubelets listen on
KUBELET_PORT="--kubelet-port=10250"
# Address range to use for services
KUBE_SERVICE_ADDRESSES="--service-cluster-ip-range=10.254.0.0/16"
# Add your own!
KUBE_API_ARGS=""
```

Enable and Start Required Services

- `systemctl enable/start`
  - » `ntpd`
  - » `etcd`
  - » `kube-apiserver`
  - » `kube-controller-manager`
  - » `kube-scheduler`

- The master controller needs to be running **before** nodes are started and attempt to register

Minions/Nodes

Configuration Files

/etc/kubernetes/config

```
# Comma separated list of nodes in the etcd cluster
KUBE_ETCD_SERVERS="--etcd-servers=http://centos-master:2379"
# logging to stderr means we get it in the systemd journal
KUBE_LOGTOSTDERR="--logtostderr=true"
# journal message level, 0 is debug
KUBE_LOG_LEVEL="--v=0"
# Should this cluster be allowed to run privileged docker containers
KUBE_ALLOW_PRIV="--allow-privileged=false"
# How the replication controller and scheduler find the kube-apiserver
KUBE_MASTER="--master=http://centos-master:8080"
```

/etc/kubernetes/kubelet

```
# The address for the info server to serve on
KUBELET_ADDRESS="--address=0.0.0.0"
# The port for the info server to serve on
KUBELET_PORT="--port=10250"
# You may leave this blank to use the actual hostname
KUBELET_HOSTNAME="--hostname-override=centos-minion1"
# Location of the api-server
KUBELET_API_SERVER="--api-servers=http://centos-master:8080"
# Add your own!
KUBELET_ARGS=""
```

Enable and Start Required Services

- `systemctl enable/start`
  - » `ntpd`
  - » `kube-proxy`
  - » `kubelet`
  - » `docker`
- The master controller needs to be running **before** nodes are started or attempt to register

Creating Pods

Pod Definition

Process

- Create a directory called *pods* in a central location where you want to manage your YAML files, or check them into your Git repository

- Sample configuration file, called *myapache.yaml*:

```
apiVersion: v1
kind: Pod
metadata:
  name: myapache
spec:
  containers:
  - name: myapache
image: user/myapache:latest
ports:
- containerPort: 80
```

Create the Pod

- Using the `kubectl utility:`

```
kubectl create -f /path/to/myapache.yaml
```

Verify Nodes

```
kubectl get nodes
# or
kubectl describe nodes
```

Verify Pod (Above)

```
kubectl get pods
# or
kubectl describe pods
```

Test Pod Availability and Communication Within Pod Structures

- Containers within a pod can communicate with other containers in the same pod or cluster
- Start a *busybox* container to test the pod we just created; for example:

```
# kubectl run busybox --image=busybox --restart=Never --tty -i
--generator=run-pod/v1 --env "POD_IP=$(kubectl get pod myapache -o go-
template='{{.status.podIP}}')"
/$ wget -qO- http://$POD_IP
/$ exit
# kubectl delete pod busybox # Clean up the pod we created with “kubectl
run”
```

This should return the default Apache site for the container deployed in our pod, provided that the image you pulled starts Apache by default on container launch

kubectl - Function Listing

******Taken from Kubernetes official documentation; always check http://kubernetes.io for latest information******

Viewing and Finding Resources

```
# Columnar output
$ kubectl get services                          # List all services in
the namespace
$ kubectl get pods --all-namespaces             # List all pods in all
namespaces
$ kubectl get pods -o wide                      # List all pods in the
namespace, with more details
$ kubectl get rc <rc-name>                      # List a particular
replication controller
$ kubectl get replicationcontroller <rc-name>   # List a particular RC


# Verbose output
$ kubectl describe nodes <node-name>
$ kubectl describe pods <pod-name>
$ kubectl describe pods/<pod-name>              # Equivalent to previous
$ kubectl describe pods <rc-name>               # Lists pods created by
<rc-name> using common prefix


# List Services Sorted by Name
$ kubectl get services --sort-by=.metadata.name


# List pods Sorted by Restart Count
$ kubectl get pods --sort-by=.status.containerStatuses[0].restartCount


# Get the version label of all pods with label app=cassandra
$ kubectl get pods --selector=app=cassandra rc -o ‘jsonpath={.items[*].
metadata.labels.version}’


# Get ExternalIPs of all nodes
$ kubectl get nodes -o jsonpath=’{.items[*].status.addresses[?(@.
type==”ExternalIP”)].address}’
```

```
# List Names of Pods that belong to Particular RC
# “jq” command useful for transformations that are too complex for
jsonpath
$ sel=$(./kubectl get rc <rc-name> --output=json | jq -j ‘.spec.selector
| to_entries | .[] | “\(.key)=\(.value),”’)
$ sel=${sel%?} # Remove trailing comma
$ pods=$(kubectl get pods --selector=$sel --output=jsonpath={.items..
metadata.name})`


# Check which nodes are ready
$ kubectl get nodes -o jsonpath=’{range .items[*]}{@.metadata.
name}:{range @.status.conditions[*]}{@.type}={@.status};{end}{end}’| tr
‘;’ “\n”  | grep “Ready=True”
```

Modifying and Deleting Resources

```
$ kubectl label pods <pod-name> new-label=awesome                  # Add
a Label
$ kubectl annotate pods <pod-name> icon-url=http://goo.gl/XXBTWq   # Add
an annotation
```

Interacting with Existing/Running Pods

```
$ kubectl logs <pod-name>           # dump pod logs (stdout)
$ kubectl logs -f <pod-name>        # stream pod logs (stdout) until
canceled (ctrl-c) or timeout


$ kubectl run -i --tty busybox --image=busybox -- sh      # Run pod as
interactive shell
$ kubectl attach <podname> -i                             # Attach to
Running Container
$ kubectl port-forward <podname> <local-and-remote-port>  # Forward port
of Pod to your local machine
$ kubectl port-forward <servicename> <port>               # Forward port
to service
$ kubectl exec <pod-name> -- ls /                         # Run command
in existing pod (1 container case)
$ kubectl exec <pod-name> -c <container-name> -- ls /     # Run command
in existing pod (multi-container case)
```

Useful Commands

```
# List all pods in ps output format.
kubectl get pods


# List all pods in ps output format with more information (such as node
name).
```

```
kubectl get pods -o wide


# List a single replication controller with specified NAME in ps output
format.
kubectl get replicationcontroller web


# List a single pod in JSON output format.
kubectl get -o json pod web-pod-13je7


# List a pod identified by type and name specified in “pod.yaml” in JSON
output format.
kubectl get -f pod.yaml -o json


# Return only the phase value of the specified pod.
kubectl get -o template pod/web-pod-13je7 --template=


# List all replication controllers and services together in ps output
format.
kubectl get rc,services


# List one or more resources by their type and names.
kubectl get rc/web service/frontend pods/web-pod-13je7
```