1 Kubectl Kubernetes CheatSheet CLOUD

- PDF Link: cheatsheet-kubernetes-A4.pdf, Category: Cloud
- Blog URL: `https://cheatsheet.dennyzhang.com/cheatsheet-kubernetes-A4`
- Related posts: Kubectl CheatSheet, Kubernetes Yaml, #denny-cheatsheets

File me Issues or star this repo.

1.1 Common Commands

Name	Command
Run curl test temporarily	`kubectl run --generator=run-pod/v1 --rm mytest --image=yauritux/busybox-cu`
Run wget test temporarily	`kubectl run --generator=run-pod/v1 --rm mytest --image=busybox -it wget`
Run nginx deployment with 2 replicas	`kubectl run my-nginx --image=nginx --replicas=2 --port=80`
Run nginx pod and expose it	`kubectl run my-nginx --restart=Never --image=nginx --port=80 --expose`
Run nginx deployment and expose it	`kubectl run my-nginx --image=nginx --port=80 --expose`
List authenticated contexts	`kubectl config get-contexts`, `~/.kube/config`
Set namespace preference	`kubectl config set-context <context_name> --namespace=<ns_name>`
List pods with nodes info	`kubectl get pod -o wide`
List everything	`kubectl get all --all-namespaces`
Get all services	`kubectl get service --all-namespaces`
Get all deployments	`kubectl get deployments --all-namespaces`
Show nodes with labels	`kubectl get nodes --show-labels`
Get resources with json output	`kubectl get pods --all-namespaces -o json`
Validate yaml file with dry run	`kubectl create --dry-run --validate -f pod-dummy.yaml`
Start a temporary pod for testing	`kubectl run --rm -i -t --image=alpine test-$RANDOM -- sh`
kubectl run shell command	`kubectl exec -it mytest -- ls -l /etc/hosts`
Get system conf via configmap	`kubectl -n kube-system get cm kubeadm-config -o yaml`
Get deployment yaml	`kubectl -n denny-websites get deployment mysql -o yaml`
Explain resource	`kubectl explain pods`, `kubectl explain svc`
Watch pods	`kubectl get pods -n wordpress --watch`
Query healthcheck endpoint	`curl -L http://127.0.0.1:10250/healthz`
Open a bash terminal in a pod	`kubectl exec -it storage sh`
Check pod environment variables	`kubectl exec redis-master-ft9ex env`
Enable kubectl shell autocompletion	`echo "source <(kubectl completion bash)" >~/.bashrc`, and reload
Use minikube dockerd in your laptop	`eval $(minikube docker-env)`, No need to push docker hub any more
Kubectl apply a folder of yaml files	`kubectl apply -R -f .`
Get services sorted by name	kubectl get services –sort-by=.metadata.name
Get pods sorted by restart count	kubectl get pods –sort-by='.status.containerStatuses[0].restartCount'
List pods and images	kubectl get pods -o='custom-columns=PODS:.metadata.name,Images:.spec.containers[*
List all container images	list-all-images.sh
kubeconfig skip tls verification	skip-tls-verify.md
Ubuntu install kubectl	`"deb https://apt.kubernetes.io/ kubernetes-xenial main"`
Reference	GitHub: kubernetes releases
Reference	minikube cheatsheet, docker cheatsheet, OpenShift CheatSheet

1.2 Check Performance

Name	Command
Get node resource usage	`kubectl top node`
Get pod resource usage	`kubectl top pod`
Get resource usage for a given pod	`kubectl top <podname> --containers`
List resource utilization for all containers	`kubectl top pod --all-namespaces --containers=true`

1.3 Resources Deletion

Name	Command
Delete pod	`kubectl delete pod/<pod-name> -n <my-namespace>`
Delete pod by force	`kubectl delete pod/<pod-name> --grace-period=0 --force`
Delete pods by labels	`kubectl delete pod -l env=test`
Delete deployments by labels	`kubectl delete deployment -l app=wordpress`
Delete all resources filtered by labels	`kubectl delete pods,services -l name=myLabel`
Delete resources under a namespace	`kubectl -n my-ns delete po,svc --all`
Delete persist volumes by labels	`kubectl delete pvc -l app=wordpress`
Delete state fulset only (not pods)	`kubectl delete sts/<stateful_set_name> --cascade=false`

1.4 Log & Conf Files

Name	Comment
Config folder	`/etc/kubernetes/`
Certificate files	`/etc/kubernetes/pki/`
Credentials to API server	`/etc/kubernetes/kubelet.conf`
Superuser credentials	`/etc/kubernetes/admin.conf`
kubectl config file	`~/.kube/config`
Kubernets working dir	`/var/lib/kubelet/`
Docker working dir	`/var/lib/docker/`, `/var/log/containers/`
Etcd working dir	`/var/lib/etcd/`
Network cni	`/etc/cni/net.d/`
Log files	`/var/log/pods/`
log in worker node	`/var/log/kubelet.log`, `/var/log/kube-proxy.log`
log in master node	`kube-apiserver.log`, `kube-scheduler.log`, `kube-controller-manager.log`
Env	`/etc/systemd/system/kubelet.service.d/10-kubeadm.conf`
Env	export KUBECONFIG=/etc/kubernetes/admin.conf

1.5 Pod

Name	Command
List all pods	`kubectl get pods`
List pods for all namespace	`kubectl get pods -all-namespaces`
List all critical pods	`kubectl get -n kube-system pods -a`
List pods with more info	`kubectl get pod -o wide`, `kubectl get pod/<pod-name> -o yaml`
Get pod info	`kubectl describe pod/srv-mysql-server`
List all pods with labels	`kubectl get pods --show-labels`
List all unhealthy pods	kubectl get pods –field-selector=status.phase!=Running –all-namespaces
List running pods	kubectl get pods –field-selector=status.phase=Running
Get Pod initContainer status	`kubectl get pod --template '{{.status.initContainerStatuses}}' <pod-name>`
kubectl run command	kubectl exec -it -n "$ns" "$podname" – sh -c "echo $msg »/dev/err.log"
Watch pods	`kubectl get pods -n wordpress --watch`
Get pod by selector	kubectl get pods –selector="app=syslog" -o jsonpath='{.items[*].metadata.name}'
List pods and images	kubectl get pods -o='custom-columns=PODS:.metadata.name,Images:.spec.containers[*].image'
List pods and containers	-o='custom-columns=PODS:.metadata.name,CONTAINERS:.spec.containers[*].name'
Reference	Link: kubernetes yaml templates

1.6 Label & Annontation

Name	Command
Filter pods by label	`kubectl get pods -l owner=denny`
Manually add label to a pod	`kubectl label pods dummy-input owner=denny`
Remove label	`kubectl label pods dummy-input owner-`
Manually add annonation to a pod	`kubectl annotate pods dummy-input my-url=https://dennyzhang.com`

1.7 Deployment & Scale

Name	Command
Scale out	`kubectl scale --replicas=3 deployment/nginx-app`
online rolling upgrade	`kubectl rollout app-v1 app-v2 --image=img:v2`
Roll backup	`kubectl rollout app-v1 app-v2 --rollback`
List rollout	`kubectl get rs`
Check update status	`kubectl rollout status deployment/nginx-app`
Check update history	`kubectl rollout history deployment/nginx-app`
Pause/Resume	`kubectl rollout pause deployment/nginx-deployment`, `resume`
Rollback to previous version	`kubectl rollout undo deployment/nginx-deployment`
Reference	Link: kubernetes yaml templates, Link: Pausing and Resuming a Deployment

1.8 Quota & Limits & Resource

Name	Command
List Resource Quota	`kubectl get resourcequota`
List Limit Range	`kubectl get limitrange`
Customize resource definition	`kubectl set resources deployment nginx -c=nginx --limits=cpu=200m`
Customize resource definition	`kubectl set resources deployment nginx -c=nginx --limits=memory=512Mi`
Reference	Link: kubernetes yaml templates

1.9 Service

Name	Command
List all services	`kubectl get services`
List service endpoints	`kubectl get endpoints`
Get service detail	`kubectl get service nginx-service -o yaml`
Get service cluster ip	kubectl get service nginx-service -o go-template='{{.spec.clusterIP}}'
Get service cluster port	kubectl get service nginx-service -o go-template='{{(index .spec.ports 0).port}}'
Expose deployment as lb service	`kubectl expose deployment/my-app --type=LoadBalancer --name=my-service`
Expose service as lb service	`kubectl expose service/wordpress-1-svc --type=LoadBalancer --name=ns1`
Reference	Link: kubernetes yaml templates

1.10 Secrets

Name	Command
List secrets	`kubectl get secrets --all-namespaces`
Generate secret	`echo -n 'mypasswd', then redirect to base64 --decode`
Get secret	`kubectl get secret denny-cluster-kubeconfig`
Get a specific field of a secret	kubectl get secret denny-cluster-kubeconfig -o jsonpath="{.data.value}"
Create secret from cfg file	kubectl create secret generic db-user-pass –from-file=./username.txt
Reference	Link: kubernetes yaml templates, Link: Secrets

1.11 StatefulSet

Name	Command
List statefulset	`kubectl get sts`
Delete statefulset only (not pods)	`kubectl delete sts/<stateful_set_name> --cascade=false`
Scale statefulset	`kubectl scale sts/<stateful_set_name> --replicas=5`
Reference	Link: kubernetes yaml templates

1.12 Volumes & Volume Claims

Name	Command
List storage class	`kubectl get storageclass`
Check the mounted volumes	`kubectl exec storage ls /data`
Check persist volume	`kubectl describe pv/pv0001`
Copy local file to pod	`kubectl cp /tmp/my <some-namespace>/<some-pod>:/tmp/server`
Copy pod file to local	`kubectl cp <some-namespace>/<some-pod>:/tmp/server /tmp/my`
Reference	Link: kubernetes yaml templates

1.13 Events & Metrics

Name	Command
View all events	`kubectl get events --all-namespaces`
List Events sorted by timestamp	kubectl get events –sort-by=.metadata.creationTimestamp

1.14 Node Maintenance

Name	Command
Mark node as unschedulable	`kubectl cordon $NODE_NAME`
Mark node as schedulable	`kubectl uncordon $NODE_NAME`
Drain node in preparation for maintenance	`kubectl drain $NODE_NAME`

1.15 Namespace & Security

Name	Command
List authenticated contexts	`kubectl config get-contexts`, `~/.kube/config`
Set namespace preference	`kubectl config set-context <context_name> --namespace=<ns_name>`
Switch context	`kubectl config use-context <cluster-name>`
Load context from config file	`kubectl get cs --kubeconfig kube_config.yml`
Delete the specified context	`kubectl config delete-context <cluster-name>`
List all namespaces defined	`kubectl get namespaces`
List certificates	`kubectl get csr`
Check user privilege	kubectl –as=system:serviceaccount:ns-denny:test-privileged-sa -n ns-denny auth can-i use pods/lis
Check user privilege	`kubectl auth can-i use pods/list`
Reference	Link: kubernetes yaml templates

1.16 Network

Name	Command
Temporarily add a port-forwarding	`kubectl port-forward redis-134 6379:6379`
Add port-forwaring for deployment	`kubectl port-forward deployment/redis-master 6379:6379`
Add port-forwaring for replicaset	`kubectl port-forward rs/redis-master 6379:6379`
Add port-forwaring for service	`kubectl port-forward svc/redis-master 6379:6379`
Get network policy	`kubectl get NetworkPolicy`

1.17 Patch

Name	Summary
Patch service to loadbalancer	kubectl patch svc svc_{name}$ -p '{"spec": {"type": "LoadBalancer"}}'

1.18 Extenstions

Name	Summary
Enumerates the resource types available	`kubectl api-resources`
List api group	`kubectl api-versions`
List all CRD	`kubectl get crd`
List storageclass	`kubectl get storageclass`

1.19 Components & Services

1.19.1 Services on Master Nodes

Name	Summary
kube-apiserver	API gateway. Exposes the Kubernetes API from master nodes
etcd	reliable data store for all k8s cluster data
kube-scheduler	schedule pods to run on selected nodes
kube-controller-manager	Reconcile the states. node/replication/endpoints/token controller and service account, etc
cloud-controller-manager	

1.19.2 Services on Worker Nodes

Name	Summary
kubelet	A node agent makes sure that containers are running in a pod
kube-proxy	Manage network connectivity to the containers. e.g, iptable, ipvs
Container Runtime	Kubernetes supported runtimes: dockerd, cri-o, runc and any OCI runtime-spec implementation.

1.19.3 Addons: pods and services that implement cluster features

Name	Summary
DNS	serves DNS records for Kubernetes services
Web UI	a general purpose, web-based UI for Kubernetes clusters
Container Resource Monitoring	collect, store and serve container metrics
Cluster-level Logging	save container logs to a central log store with search/browsing interface

1.19.4 Tools

Name	Summary
kubectl	the command line util to talk to k8s cluster
kubeadm	the command to bootstrap the cluster
kubefed	the command line to control a Kubernetes Cluster Federation
Kubernetes Components	Link: Kubernetes Components

1.20 More Resources

`https://kubernetes.io/docs/reference/kubectl/cheatsheet/`
`https://codefresh.io/kubernetes-guides/kubernetes-cheat-sheet/`

phoenixNAP® GLOBAL IT SERVICES

kubectl Commands Cheat Sheet

Listing Resources

kubectl get namespaces	Generate a plain-text list of all namespaces
kubectl get pods	Generate a plain-text list of all pods
kubectl get pods -o wide	Generate a detailed plain-text list of all pods
kubectl get pods --field-selector=spec.nodeName=[server-name]	Generate a list of all pods running on a particular node server
kubectl get replicationcontroller [replication-controller-name]	List a specific replication controller in plain text
kubectl get replicationcontroller, services	Generate a plain-text list of all replication controllers and services
kubectl get deamonset	Generate a plain-text list of all daemon sets

Creating a Resource

kubectl create namespace [namespace-name]	Create a new namespace
kubectl create –f [filename]	Create a resource from a JSON or YAML file

Applying & Updating a Resource

kubectl apply -f [service-name].yaml	Create a new service with the definition contained in [service-name].yaml
kubectl apply -f [controller-name].yaml	Create a new replication controller with the definition contained in [controller-name].yaml
kubectl apply -f [directory-name]	Create the objects defined in any .yaml, .yml, or .json file in a directory
kubectl edit svc/[service-name]	Edit a service
KUBE_EDITOR=" [editor-name]" kubectl edit svc/[service-name]	Edit a service in a non-default editor

Displaying the State of Resources

kubectl describe nodes [node-name]	See details about a particular node
kubectl describe pods [pod-name]	See details about a particular pod
Kubectl describe –f pod.json	See details about a pod whose name and type are listed in pod.json
kubectl describe pods [replication-controller-name]	See details about all pods managed by a specific replication controller
kubectl describe pods	See details about all pods

Deleting Resources

kubectl delete -f pod.yaml	Remove a pod using the name and type listed in pod.yaml:
kubectl delete pods,services -l [label-key]=[label-value]	Remove all the pods and services with a specific label:
kubectl delete pods --all	Remove all pods. The command will include uninitialized pods as well

Executing a Command

kubectl exec [pod-name] -- [command]	Receive output from a command run on the first container in a pod:
kubectl exec [pod-name] -c [container-name] -- [command]	Receive output from a command run on a specific container in a pod
kubectl exec -ti [pod-name] -- /bin/bash	Run /bin/bash from a specific pod. The output received comes from the first container

Modifying kubeconfig Files

kubectl config current-context	Display the current context
kubectl config set-cluster [cluster-name] --server=[server-name]	Set a cluster entry in kubeconfig
kubectl config unset [property-name]	Unset an entry in kubeconfig

Printing Container Logs

kubectl logs [pod-name]	Print logs from a pod
kubectl logs -f [pod-name]	Stream logs from a pod

Resource Types - Short Names

Short name	Full name
csr	certificatesigningrequests
cs	componentstatuses
cm	configmaps
ds	daemonsets
deploy	deployments
ep	endpoints
ev	events
hpa	horizontalpodautoscalers
ing	ingresses
limits	limitranges
ns	namespaces
no	nodes
pvc	persistentvolumeclaims
pv	persistentvolumes
po	pods
pdb	poddisruptionbudgets
psp	podsecuritypolicies
rs	replicasets
rc	replicationcontrollers
quota	resourcequotas
sa	serviceaccounts
svc	services

Kubernetes kubectl CLI Cheat Sheet

ALTOROS™

About Kubectl

Kubectl is a command line interface for running commands against **Kubernetes** clusters.

CLI config

Kubectl config contains cluster's API endpoint, credentials and can be configured to use several contexts.

`kubectl version` shows a kubectl and a kubernetes cluster components version

`kubectl config view` shows a kubectl config

`kubectl config current-context` shows a current context

`kubectl config use-context my-k8s` uses a particular context

```
kubectl config set-credentials \
  kubeuser/foo.kubernetes.com \
  --usename=kubeuser \
  --password=kubepassword
```
adds a new cluster and user credentials to your kubectl config

Namespaces

Kubernetes namespaces can be presented as directories, that help to group resources logically. The `default` namespace is used by default. The `kube-system` namespace is typically used for cluster resources.

`kubectl get ns` gets a list of all namespaces

`kubectl create ns jenkins` creates a namespace named jenkins

The `default` namespace will be used in every command by default. To change this behaviour, use `--namespace=<name>` and `--all-namespaces` flags.

Manage cluster

`kubectl cordon worker-1` marks a node as unschedulable

`kubectl drain worker-1` prepares worker-1 for maintenance, removes all resources from a node

`kubectl cluster-info` gets cluster information

`kubectl top node kubernetes-minion-group` gets system statistics from a node kubernetes-minion-group

`kubectl label worker-1 disk=ssd` adds a label to a node instance. Labels allow to manage resources in a more flexible way

Collect information from your cluster

Types of objects: pods/services/deployments/persistentVolumes/replicaSets/statefulSets/etc.

`kubectl get <object>` gets general info about cluster resource(s)

`kubectl get <object> -o wide` shows resource information with some additional parameters

`kubectl get <object> -o [json, yaml]` gets general information in a json or a yaml output format

`kubectl describe <object>` gets general information about cluster resource(s) in details

```
kubectl get pods \
  --namespace=kube-system
```
gets info about pods in a particular namespace

`kubectl describe nodes worker-1` gets verbose description of a node named worker-1

```
kubectl get pods
--field-selector=status.phase=Failed
--all-namespaces
```
gets all pods in a failed state from the whole cluster

```
kubectl describe all \
  --all-namespaces
```
describes all cluster resources

Create resources in your cluster

Do not mix `create` and `apply` techniques when creating objects. The `create` command does not retain `kubectl.kubernetes.io/last-applied-configuration` annotation, which is used by the `apply` command. `Apply` is imperative and can accumulate changes made to an object (e.g by scale command).

`kubectl create -f ./manifest.yaml` creates a resource described in a manifest

`kubectl apply -f ./dir` creates resources from all files in a directory

`kubectl run dev-nginx --image=nginx` runs a single nginx instance

Update resources

Kubernetes allows you to easily scale your resources.

```
kubectl scale deployment \
  --replicas=3 -l run=nginx-a
```
scales nginx to 3 replicas

You can easily make rolling updates with zero downtime.

```
kubectl rolling-update frontend-v1 \
  -f frontend-v2.json
```
updates pods of frontend with zero downtime

```
kubectl rollout undo \
  deployment/nginx-deployment \
  --to-revision=2
```
rollbacks a nginx deployment to a specified revision

```
kubectl autoscale deployment \
  nginx-deployment --min=10 \
  --max=15 --cpu-percent=80
```
autoscales a nginx deployment based on CPU load

```
kubectl replace --force -f \
  ./jenkins.json
```
replaces and updates resources described in a `jenkins.json` with downtime

```
kubectl label pods jenkins \
  new-label=devqa
```
creates a label on a pod jenkins

```
kubectl edit pod \
  kube-dns-565cd5b8c9-j6zmd \
  --namespace=kube-system
```
edits a resource manifest with your text editor

Delete resources

`kubectl delete -f ./pod.json` deletes resources described in a manifest

```
kubectl delete pods,services -l \
  name=myLabel --all-namespaces
```
deletes pods and services with the label myLabel from all namespaces

Pod debugging tools

`kubectl logs nginx-8586cf59-nj55x` collects logs from a pod

`kubectl top pod nginx` shows pod's metrics

`kubectl exec -it nginx -- /bin/bash` creates or starts an interactive shell into pod

`kubectl port-forward nginx 8080:80` forwards a container port 80 to a local port 8080 so that you can access your containerized app for debugging

```
kubectl cp hotfix.yaml \
  web1:/config/hotfix.yaml
```
copies a file to or from a container file system

NOTE: Using `kubectl cp` for any purposes other than debugging or hotfixing is considered to be an antipattern.

Configmaps and Secrets

Secret is a primitive to store sensitive data (passwords, keys, certificates, and etc.) in a container. **Configmap** is a primitive to store pod's configuration.

```
kubectl create configmap back-config \
  --from-file=my-config.txt \
  --from-literal=type=binary \
  --from-literal=ext_port=12803
```
creates config map from both separate vars and my-config.txt file

```
kubectl describe configmaps \
  back-1-config
```
gets configmap configuration values

```
kubectl create secret generic web-tls \
  --from-file=web.crt \
  --from-file=web.key
```
creates a secret object to store and use TSL certificates

```
kubectl delete secrets \
  dev-concourse-postgresql
```
deletes secrets from a stated object

Helm tool for Kubernetes

Helm is a tool, which helps with complex solutions (like db clusters, or CI tools) deployment to Kubernetes. It is used to install sets of resources called charts, that can be found in a helm repository

`helm init` Helm gets a cluster location and credentials from kubectl config and installs a container with a **tiller** - a helm server part

`helm repo update` makes sure that helm charts are in actual state

```
helm install --name dev-concourse \
  stable/concourse
```
installs a Concourse helm chart (creates a deployment and a corresponding service)

`helm delete dev-concourse` deletes dev-concourse chart resources

Kubernetes Cheat Sheet

What is Kubernetes?

Kubernetes is a platform for managing containerized workloads. Kubernetes orchestrates computing, networking and storage to provide a seamless portability across infrastructure providers.

Viewing Resource Information

Nodes

```
$ kubectl get no
$ kubectl get no -o wide
$ kubectl describe no
$ kubectl get no -o yaml
$ kubectl get node --selector=[label_name]
$ kubectl get nodes -o
jsonpath='{.items[*].status.addresses
[?(@.type=="ExternalIP")].address}'
$ kubectl top node [node_name]
```

Pods

```
$ kubectl get po
$ kubectl get po -o wide
$ kubectl describe po
$ kubectl get po --show-labels
$ kubectl get po -l app=nginx
$ kubectl get po -o yaml
$ kubectl get pod [pod_name] -o yaml
--export
$ kubectl get pod [pod_name] -o yaml
--export > nameoffile.yaml
$ kubectl get pods --field-selector
status.phase=Running
```

Namespaces

```
$ kubectl get ns
$ kubectl get ns -o yaml
$ kubectl describe ns
```

Deployments

```
$ kubectl get deploy
$ kubectl describe deploy
$ kubectl get deploy -o wide
$ kubectl get deploy -o yaml
```

Services

```
$ kubectl get svc
$ kubectl describe svc
$ kubectl get svc -o wide
$ kubectl get svc -o yaml
$ kubectl get svc --show-labels
```

DaemonSets

```
$ kubectl get ds
$ kubectl get ds --all-namespaces
$ kubectl describe ds [daemonset_name] -n
[namespace_name]
$ kubectl get ds [ds_name] -n [ns_name] -o
yaml
```

Events

```
$ kubectl get events
$ kubectl get events -n kube-system
$ kubectl get events -w
```

Logs

```
$ kubectl logs [pod_name]
$ kubectl logs --since=1h [pod_name]
$ kubectl logs --tail=20 [pod_name]
$ kubectl logs -f -c [container_name]
[pod_name]
$ kubectl logs [pod_name] > pod.log
```

Service Accounts

```
$ kubectl get sa
$ kubectl get sa -o yaml
$ kubectl get serviceaccounts default -o
yaml > ./sa.yaml
$ kubectl replace serviceaccount default -f
./sa.yaml
```

ReplicaSets

```
$ kubectl get rs
$ kubectl describe rs
$ kubectl get rs -o wide
$ kubectl get rs -o yaml
```

Roles

```
$ kubectl get roles --all-namespaces
$ kubectl get roles --all-namespaces -o yaml
```

Secrets

```
$ kubectl get secrets
$ kubectl get secrets --all-namespaces
$ kubectl get secrets -o yaml
```

ConfigMaps

```
$ kubectl get cm
$ kubectl get cm --all-namespaces
$ kubectl get cm --all-namespaces -o yaml
```

Ingress

```
$ kubectl get ing
$ kubectl get ing --all-namespaces
```

PersistentVolume

```
$ kubectl get pv
$ kubectl describe pv
```

PersistentVolumeClaim

```
$ kubectl get pvc
$ kubectl describe pvc
```

Kubernetes Cheat Sheet

Viewing Resource Information (cont.)

StorageClass

```
$ kubectl get sc
$ kubectl get sc -o yaml
```

Multiple Resources

```
$ kubectl get svc, po
$ kubectl get deploy, no
$ kubectl get all
$ kubectl get all --all-namespaces
```

Changing Resource Attributes

Taint

```
$ kubectl taint [node_name] [taint_name]
```

Labels

```
$ kubectl label [node_name] disktype=ssd
$ kubrectl label [pod_name] env=prod
```

Cordon/Uncordon

```
$ kubectl cordon [node_name]
$ kubectl uncordon [node_name]
```

Drain

```
$ kubectl drain [node_name]
```

Nodes/Pods

```
$ kubectl delete node [node_name]
$ kubectl delete pod [pod_name]
$ kubectl edit node [node_name]
$ kubectl edit pod [pod_name]
```

Deployments/Namespaces

```
$ kubectl edit deploy [deploy_name]
$ kubectl delete deploy [deploy_name]
$ kubectl expose deploy [deploy_name]
--port=80 --type=NodePort
$ kubectl scale deploy [deploy_name]
--replicas=5
$ kubectl delete ns
$ kubectl edit ns [ns_name]
```

Services

```
$ kubectl edit svc [svc_name]
$ kubectl delete svc [svc_name]
```

DaemonSets

```
$ kubectl edit ds [ds_name] -n kube-system
$ kubectl delete ds [ds_name]
```

Service Accounts

```
$ kubectl edit sa [sa_name]
$ kubectl delete sa [sa_name]
```

Annotate

```
$ kubectl annotate po [pod_name]
[annotation]
$ kubectl annotate no [node_name]
```

Adding Resources

Creating a Pod

```
$ kubectl create -f [name_of_file]
$ kubectl apply -f [name_of_file]
$ kubectl run [pod_name] --image=nginx
--restart=Never
$ kubectl run [pod_name]
--generator=run-pod/v1 --image=nginx
$ kubectl run [pod_name] --image=nginx
--restart=Never
```

Creating a Service

```
$ kubectl create svc nodeport [svc_name]
--tcp=8080:80
```

Creating a Deployment

```
$ kubectl create -f [name_of_file]
$ kubectl apply -f [name_of_file]
$ kubectl create deploy [deploy_name]
--image=nginx
```

Interactive Pod

```
$ kubectl run [pod_name] --image=busybox
--rm -it --restart=Never -- sh
```

Output YAML to a File

```
$ kubectl create deploy [deploy_name]
--image=nginx --dry-run -o yaml >
deploy.yaml
$ kubectl get po [pod_name] -o yaml --export
> pod.yaml
```

Getting Help

```
$ kubectl -h
$ kubectl create -h
$ kubectl run -h
$ kubectl explain deploy.spec
```

Requests

API Call

```
$ kubectl get --raw /apis/metrics.k8s.io/
```

Cluster Info

```
$ kubectl config
$ kubectl cluster-info
$ kubectl get componentstatuses
```

mimacom
Kubernetes Cheat Sheet

MIMACOM
Digital. Innovation. Engineers.

Kubernetes, is an open-source system for automating deployment, scaling, and management of containerized applications. It is built for planet scale to run millions of containers, while it's flexibility allows customization to meet any needs so in fact, Kubernetes can run anywhere from on-premises to hybrid or public cloud infrastructure.

AUTHENTICATION

A context connects to a K8s cluster with a specific user.

```
kubectl config get-contexts
```
View all available and configured contexts

```
kubectl config current-context
```
Show the context that is currently being used

```
kubectl config use-context <context-name>
```
Switch to another context and use it for subsequent commands

```
kubectl config view
```
View the merged configuration file

CREATING RESOURCES

Create resources declaratively with a manifest.

```
kubectl apply -f <manifest.yml>
```
Create and update resources to match definition in manifest file.

```
kubectl apply -k <directory>
```
Apply a folder using Kustomize configuration management.

```
kubectl create job <name> --image=busybox -- echo 'Hello mimacom!'
```
Imperatively create a resource, in this case a job.

```
kubectl create secret generic <name> --from-literal=<key>=<value>
```
Create a secret from the CLI. Do not store secrets in manifest files.

EXPLORING RESOURCE TYPES

Every cluster may have different resource types.

```
kubectl api-resources
```
Retrieve a list of all available resource types of the cluster.

```
kubectl explain <type>
```
Get the documentation for the resource type.

```
kubectl explain <type>.<property>
```
Get documentation for a specific property.

DELETING RESOURCES

Resources that are deleted are usually gone forever.

```
kubectl delete <type> <name>
```
Delete a specific resource of type identified by its name.

```
kubectl delete -f <manifest.yml>
```
Delete all resources mentioned in the manifest file.

EDITING RESOURCES

For a quick test you can manually edit resources.

```
kubectl edit <type> <name>
```
Edit a specific resource and sync any changes to the cluster.

```
KUBE_EDITOR="nano"
```
Set the editor to use to your liking.

VIEW AND FIND RESOURCES

Explore the resource objects available.

```
kubectl get <type>
```
Get all resources of a type in current namespace, e.g. pods.

```
kubectl get <type> -A
```
Get all resources of a type across all namespaces.

```
kubectl get <type> <name>
```
Get condensed info on a specific resource by name.

```
kubectl describe <type> <name>
```
Get verbose info on a specific resource by name.

```
kubectl get <type> <name> -o yaml
```
Get a specific resource in its complete YAML representation.

```
kubectl get <type> --show-labels
```
Get all resource along with the labels defined on each resource.

```
kubectl get <type> -l <label>=<value>
```
Show only resources that have label set to value.

OPERATING APPLICATIONS

Rollout and scale your applications.

```
kubectl rollout history <type> <name>
```
View the deployment history of a specific deployment

```
kubectl rollout undo <type> <name>
```
Rollback to the previous deployment.

```
kubectl rollout restart <type> <name>
```
Rolling restart of a resource.

```
kubectl scale --replicas=3 <type> <name>
```
Scale a resource to a number of replicas.

TROUBLESHOOTING

When things go south start investigating.

```
kubectl get <type> <name>
```
If you identify a faulty resource, start by investigating its details.

```
kubectl logs <pod>
```
Get logs of a specific pod.

```
kubectl logs <pod> -f
```
Stream and follow the logs in real-time.

```
kubectl logs <pod> --previous
```
See the log output of the previous container, if still available.

```
kubectl exec <pod> -- curl localhost:8080
```
Execute a command on a pod.

```
kubectl exec <pod> -it -- /bin/sh
```
Start an interactive shell on the pod.

```
kubectl port-forward <pod> 8080:6000
```
Port-forwarding from a pod to your local machine.

```
kubectl get events
```
View cluster/namespace events.

```
kubectl top pods
```
View CPU and memory usage for all pods.

GENERAL OPTIONS

These general options are useful from time to time.

```
kubectl --namespace <namespace>
```
Apply the command to a specific namespace.

```
kubectl --v=<0..9>
```
Increase log output to even see HTTP requests starting at level 7.

EXAMPLE MANIFESTS

Use these example manifests as quick reference.

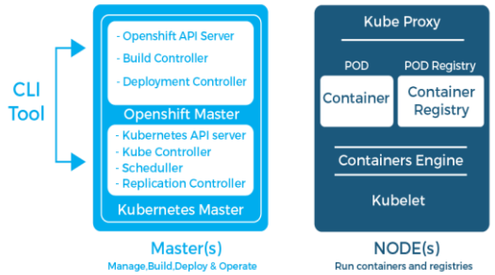
```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mima-deployment
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
  selector:
    matchLabels: {'app': 'mima-app'}
  template:
    metadata:
      labels: {'app': 'mima-app'}
    spec:
      containers:
      - name: mima-app-container
        image: ...
        ports:
        - containerPort: 3000
        resources:
          limits:
            memory: "256M"
            cpu: "500m"
        livenessProbe:
          httpGet:
            port: 3000
            path: '/actuator/info'
```
A deployment manifest with some important properties set.

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: mima-app-service
spec:
  type: ClusterIP
  ports:
  - port: 3000
  selector: {'app': 'mima-app'}
```
A service exposing the above deployment.

KUBERNETES CHEAT SHEET

KUBERNETES

- It is an open source platform for automating deployment and scaling of containers across clusters of hosts providing container centric infrastructure.
- It is a container orchestrator and can run Linux containers:
  - Launch container.
  - Maintain and monitor container site.
  - Performs container-oriented networking



Key Concepts

Now let's discuss the key points of this architecture.

- **Pod:** These are the group of containers.
- **Labels:** These are used to identify the pods.
- **Kubelet:** They are container agents, responsible for maintaining the set of pods.
- **Proxy:** They are the Load balancer for pods, helping in distributing tasks across the pods.
- **ETCD:** A Metadata service.
- **Cadvisor:** For resource usage and performance stats.
- **Replication controller:** It manages pod replication.
- **Scheduler:** Used for pod scheduling in worker nodes.
- **API server:** Kubernetes API server.

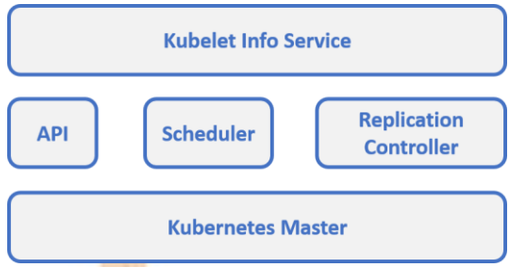
Now let's understand the role Master and Node play in the Kubernetes Architecture.

IntelliPaat

FURTHERMORE: Kubernetes Training

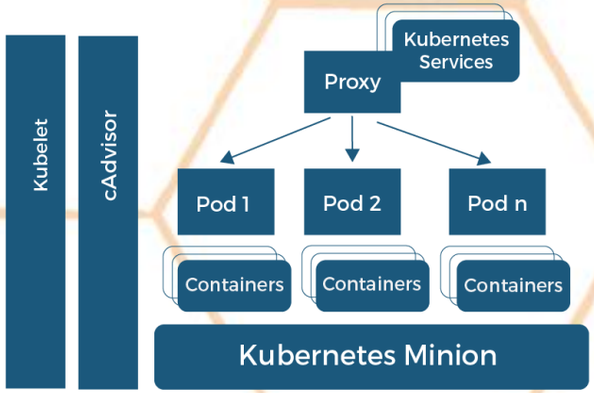
Master

- It is responsible for maintaining the desired state for the cluster you are working on.
- "Master" indicates a set of processes that are used to manage the cluster.
- Contains info, API, scheduler, replication controllers, and master.



Worker Nodes/Minions

- Also called as a minion. It contains the services necessary to run the pods that are managed by the master.
- Some services include: container runtime, Kubelet, kube-proxy.
- Contains: Kubelet, cAdvisor, services, pods and containers.



Features

- **Automated scheduling-** provides an advanced scheduler that helps launch container on cluster nodes
- **Self healing-** reschedule, replace and restart dead containers.
- **Automated rollouts and rollbacks-** supports rollback for systems incase of a failure. Enables rollout and rollback for the desired state.
- **Horizontal scaling-** can scale up and down the app as per required. Can also be automated wrt CPU usage.
- **Service discovery and load balancing-** uses unique ip and dns name to containers. This helps identify them across different containers.

Kubectl Command List

- **Pods and Container Introspection**

COMMANDS	FUNCTION
Kubectl get pods	Lists all current pods
Kubectl describe pod<name>	Describes the pod names
Kubectl get rc	List all replication controllers
Kubectl get rc --namespace="namespace"	Lists replication controllers in namespace
Kubectl describe rc <name>	Shows the replication controller name
Kubectl get cvc	Lists the services
Kubectl describe svc<name>	Shows the service name
Kubectl delete pod<name>	Deletes the pod
Kubectl get nodes -w	Watch nodes continuously

- **Debugging**

FUNCTION	COMMAND
Execute command on service by selecting container.	Kubectl exec<service><commands>[-c< $container>]
Get logs from service for a container	Kubectl logs -f<name>>[-c< $container>]
Watch the kubelet logs	Watch -n 2 cat/var/log/kublet.log
Show metrics for node	Kubectl top node
Show metrics for pods	Kubectl top pod

- **Objects**

All	clusterrolebindings	clusterroles
cm= conf gmaps	controllerrevisions	crd=custom resource definition
Cronjobs	cs=component status	csr= certificate signing requests
Deploy=deployments	ds= daemon sets	ep=end points
ev= events	hpa= autoscaling	ing= ingress
jobs	limits=limit ranges	Netpol- network policies
No = nodes	ns= namespaces	pdb= pod
po= pods	Pod preset	Pod templates
Psp= pod security policies	Pv= persistent volumes	pvc= persistent volume claims
quota= resource quotas	rc= replication controllers	Role bindings
roles	rs= replica sets	sa=service account
sc= storage classes	secrets	sts= stateful sets

- **Cluster Introspection**

FUNCTION	COMMAND
Get version information	Kubectl version
Get cluster information	Kubectl cluster-info
Get the configuration	Kubectl config g view
Output info about a node	Kubectl describe node<node>

Other Quick Commands

Launch a pod with a name an image : Kubectl run<name> --image=<image-name>
Create a service in <manifest.yaml> : Kubectl create -f <manifest.yaml>
Scale replication counter to count the number of instances : Kubectl scale --replicas=<count>
Map external port to internal replication port : Expose rc<name> --port=<external>--target-port=<internal>
To stop all pod in <n> : Kubectl drain<n>-- delete-local-data--force--ignore-daemonset
Allow master nodes to run pods : Kubectltaintnodes --all-node-role.kuernetes.io/master-