<!-- TOC -->

<!-- TOC -->

General
- Use `grep -A2 Mounts` to show two lines after the line matching `Mounts`
- Watch pods / deployments / jobs: `k get pods -w` / `k get deployments -w` / `k get jobs -w`
- Repeat command every n seconds, example: `watch -n 2 kubectl get pods`
- Check all resources at once: `k get all [-A]`
- Select the acgk8s cluster to interact: `k config use-context acgk8s`
- API e.g. for pod manifests : `k explain pods[.child1.child2] | more` OR https://kubernetes.io/docs/reference/kubernetes-api/

Create pods
- Create an nginx pod with `k run my-pod --image=nginx:alpine [--port=80] ['--labels app=my_app]`
- Create a busybox pod with `k run my-pod --image=busybox $do --command -- sh -c "touch /tmp/ready && sleep 1d" > pod6.yml`
- Create a pod with a volume backed by a config map: `k create -f https://kubernetes.io/examples/pods/pod-configmap-volume.yaml $do > pod.yml`
- Create a one-shot pod:
  - to test interactively: `k run my-pod --image busybox --restart=Never --rm -ti`
  - to check a service connection (because the Service is in a different Namespace from the test pod, it is reachable using FQDNs):
```
k run my-pod [--restart=Never] --rm -i --image=nginx:alpine -- curl -m 5 sun-srv.sun:9999
Connecting to sun-srv.sun:9999 (10.23.253.120:9999)
<title>Welcome to nginx!</title>
```

Test a pod
- With a command: `k exec my-pod [-c my-container] (-- env | grep SECRET1 || -- cat /tmp/secret2/key)`
- In interactive mode: `k exec my-pod [-c my-container] -ti -- sh`

Create other resources
- Create a job with `k create job my-job --image=busybox:1.31.0 $do > /opt/course/3/job.yaml -- sh -c "sleep 2 && echo done"` then check the pod execution (no such thing as starting a Job or CronJob!)
- Create a ConfigMap from a file, with a specific key: `k create configmap my-cm --from-file=index.html=/opt/course/15/web-moon.html`
- Create a secret (with implicit base64 encoding): `k create secret generic my-secret --from-literal user=test --from-literal pass=pwd`
- Create an nginx deployment: `k create deployment my-dep --image=nginx:stable $do > my-dep.yml` (deployment name is used as prefix for pods' name)
- Create a Service...
  - ...to expose a given pod `k expose pod my-pod --name my-svc --port 3333 --target-port 80` (much faster than creating a service and editing it to set the correct selector labels)
  - ...for an nginx deployment, which serves on port 80 and connects to the containers on port 8000: `k expose deployment nginx --port=80 --target-port=8000 [--type ClusterIp|NodePort|...] [$do]`
- Note: A NodePort Service kind of lies on top of a ClusterIP one, making the ClusterIP Service reachable on the Node IPs (internal and external).
- Create a quota: `k create quota my-quota --hard=cpu=1,memory=1G,pods=2,services=3,replicationcontrollers=2,resourcequotas=1,secrets=5,persistentvolumeclaims=10 [$do]`

Update resources
- Add / remove / change a label: `k label pods my-pod app=b` / `k label pods my-pod app-` / `k label pods my-pod app=v2 --overwrite`
- Add a new label tier=web to all pods having 'app=v2' or 'app=v1' labels: `k label po -l "app in(v1,v2)" tier=web`
- Change a pod's image (to nginx:1.7.1): `k set image my-pod nginx=nginx:1.7.1`
- Recreate the pods in a deployment: `k rollout restart deploy web-moon`
- Perform a rolling update (e.g. to change an image): `k edit deployment fish` or `k set image deployment/fish nginx=nginx:1.21.5`

- Check rollout status: `k rollout status deployment/rolling-deployment`
- Roll back to the previous version: `k rollout undo deployment/rolling-deployment`
- Autoscale a deployment, pods between 5 and 10, targetting CPU utilization at 80%: `k autoscale deploy nginx --min=5 --max=10 --cpu-percent=80`
  - View the Horizontal Pod Autoscalers (hpa): `k get hpa nginx`

Debugging
- Use `k get pods [-A] [--show-labels]`: check `STATUS`, `READY` and `RESTARTS` attributes.
- Retrieve a pod status: `k get pod <pod_name> -o json | jq .status.phase`
- Retrieve pod / container logs: `k logs <pod_name> [-c <container_name>] [-p]` (if pod crashed and restarted, -p option gets logs about the previous instance)
- List events for a given namespace / all namespaces: `k get events -n <my-namespace>` / `k get events -A`
- Show metrics for pods / pod / nodes: `k top pods` / `k top pod --selector=XXXX=YYYY` / `k top node`

Delete / replace resources
- Force replace a resource: `k replace --force -f ./pod.json`
- Delete pods and services using their label: `k delete pods,services -l app=b $now`

Secrets for ServiceAccount
- If a Secret belongs to a ServiceAccount, it'll have the annotation `kubernetes.io/service-account.name`
- Use `k get secret ...` to get a base64 encoded token
- Use `k describe secret ...` to get a base64 decoded token...or pipe it manually through `echo <token> | base64 -d -`

NetworkPolicy
- Example of egress policy, 1) restricting outgoing tcp connections from frontend to api, 2) still allowing outgoing traffic on UDP/TCP ports 53 for DNS resolution.
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy <...>
spec:
  podSelector:
    matchLabels:
      id: frontend          # label of the pods this policy should be applied on
  policyTypes:
  - Egress                  # we only want to control egress
  egress:
  - to:                     # 1st egress rule
    - podSelector:            # allow egress only to pods with api label
        matchLabels:
          id: api
  - ports:                  # 2nd egress rule
    - port: 53                # allow DNS UDP
      protocol: UDP
    - port: 53                # allow DNS TCP
      protocol: TCP
```

Helm
- List release with `helm [-n my_ns] ls [-a]`
- List pending deployments on all namespaces: `helm list --pending -A`
- List / search repo: `helm repo list` / `helm search repo nginx`
- Download (not install) a chart from a repository: `helm pull [chart URL | repo/chartname] [...] [flags]`
- Untar a chart (after downloading it): `helm pull --untar [rep/chartname]`
- Check customisable values setting for an install, e.g. `helm show values bitnami/apache [| yq e]`
- Custom install example `helm install my-apache bitnami/apache --set replicaCount=2`
- Upgrade a release, e.g. `helm upgrade my-api-v2 bitnami/nginx`
- Undo a helm rollout/upgrade: `helm rollback`
- Delete an installed release with `helm uninstall <release_name>`

```
[//]: # (### Debugging - part 2)
[//]: # (- Check cluster-level logs if you still cannot locate any relevant information.)
[//]: # (  - Check the kube-apiserver logs, e.g.)
[//]: # (    `sudo tail -100f /var/log/containers/kube-apiserver-k8s-control_kube-system_kube-apiserver-<hash>.log`)
[//]: # (  - Check the kubelet status / logs: `sudo systemctl status kubelet` / `sudo journalctl -fu kubelet`)
[//]: # (- More troubleshooting tips...)
[//]: # (  - for pods at https://kubernetes.io/docs/tasks/debug/debug-application/debug-running-pod/)
[//]: # (  - for applications at https://kubernetes.io/docs/tasks/debug/debug-application/)
[//]: # (  - for clusters at https://kubernetes.io/docs/tasks/debug/debug-cluster/)

[//]: # (### Linux)
[//]: # ()
[//]: # (- In vi / vim, to indent multiple lines:)
[//]: # (  - set the shiftwidth using :set shiftwidth=2)
[//]: # (  - mark multiple lines using **Shift v** and the up/down keys)
[//]: # (  - press `>` or `<`)
[//]: # (  - repeat / cancel the action using `.` / `u`)

[//]: # (### YAML templates)
[//]: # ()
[//]: # (- Search YAML templates)
[//]: # (  - in documentation web pages with `kind: <resource_name>`)
[//]: # (  - on disk with `grep -r <search> [directory]`)
[//]: # (- Pod: [Tasks]&#40;https://kubernetes.io/docs/tasks/&#41; > [Configure Pods and Containers]&#40;https://kubernetes.io/docs/tasks/configure-pod-container/&#41;, copy file URL then `wget <file_url>`and modify... )
[//]: # (- Deployment)
[//]: # (- ConfiMap)
[//]: # (- Secret)
[//]: # (- Service)

[//]: # (### References)
[//]: # (- https://kubernetes.io/docs/reference/k/cheatsheet/)
[//]: # (- https://github.com/dennyzhang/cheatsheet-kubernetes-A4)
[//]: # (- https://codefresh.io/blog/kubernetes-cheat-sheet/)
[//]: # (- https://intellipaat.com/blog/tutorial/devops-tutorial/kubernetes-cheat-sheet/)
```