Cluster components

Main components

- Kubernetes proxy
- Kubernetes DNS
- Kubernetes UI

Fun fact: Components in a K8s cluster are deployed using K8s itself.

Kubernetes proxy

- Responsible for routing traffic to load-balanced services.
- The proxy must be present on every node in the cluster.
- `kubectl get daemonSets --namespace=kube-system kube-proxy`

Kubernetes DNS

- Naming and discovery for the services in the cluster.
  `kubectl get deployments --namespace=kube-system core-dns`
- Load balancing service for the DNS server:
  `kubectl get services --namespace=kube-system core-dns`
- In older K8s versions (before 1.12), use `kube-dns` instead!

Kubernetes UI

- `kubectl get deployments --namespace=kube-system kubernetes-dashboard`
  ==> CHECK CHECK CHECK
- If self-hosted, you can access it through `kubectl proxy`
- http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

Kubernetes client

Useful `kubectl` commands

- `kubectl version`
- `kubectl get componentstatuses`
- `kubectl get nodes`
- `kubectl describe nodes <node>`
- `kubectl help <command name>`

Namespaces

- Kubernetes uses *namespaces* to organize objects in the cluster.
- Think of a namespace as a folder that holds a set of objects.
- By default `kubectl` interacts with the `default` namespace.
- `kubectl --namespace=<name>`
- `kubectl get pods --all-namespaces` (all pods in cluster).

Contexts

- Kubernetes uses a YAML file called `kubeconfig` (in `$HOME/.kube/config`) that stores cluster authentication information for `kubectl`.
- A **context** is a group of access parameters. Each context contains:
  - a Kubernetes cluster
  - a user
  - a namespace.
- `kubectl config view` shows the current configuration.
- Switching contexts instead of log off and reconnect.

Contexts (cont.)

- The current context is the cluster that is currently the default for kubectl: all kubectl commands run against that cluster.
- When you create a cluster using `gcloud container clusters create`, an entry is automatically added to the kubeconfig in your environment, and the current context changes to that cluster.
- Sometimes it doesn't! (e.g. if you create a cluster through GCP Console)
- `gcloud container clusters get-credentials <cluster name>`

Contexts (cont.)

- Use a different context:
  - `kubectl config use-context my-context`
- Create a namespace in a context:
  - `kubectl config set-context my-context --namespace=mystuff`

Viewing Kubernetes API objects

- Everything in a Kubernetes cluster is a RESTful resource.
- Each object has a unique HTTP path:
- The `kubectl` command makes HTTP requests to these URLs.

Viewing Kubernetes API objects (cont.)

- By default, this is human-readable. But you can modify it:
  - `-o wide`
  - `-o json`
  - `-o yaml`
- Skip headers to combine with Unix pipes `|`: `--no-headers`
- Extract information (e.g. IP address of a pod) using JSONPath:
- `kubectl get pods my-pod -o jsonpath --template={.status.podIP}`
- `kubectl describe <resource-name> <object-name>`

Viewing Kubernetes API objects (cont.)

- Objects in K8s are either JSON or YAML files, that are either returned or posted.
- You can create or make changes to an object by editing the file:
  - `kubectl apply -f my-object.yaml`
- The `apply` tool only modifies objects that are different from current objects in the cluster.
- You can do dry runs: `--dry-run`

Viewing Kubernetes API objects (cont.)

- Interactive edits:
  - `kubectl edit <resource-name> <object-name>`
    This will download the latest object state, launch an editor and upload back to the cluster after edits are saved.
- Edit history (`view-last-applied`, `edit-last-applied`, `set-last-applied`):
  - `kubectl apply -f my-object.yaml view-last-applied`
- Kill an object (**NO prompts!**):
  - `kubectl delete -f my-object.yaml`

Labels and Annotations

- Labels (`label`) and Annotations (`annotate`) are tags for K8s objects.
  - `kubectl label pods my-pod color=red` will add a `color=red` label to your pod `my-pod` .
- No overwrite by default, a flag is needed (`--overwrite`)
- You can remove a label with `-` :
  - `kubectl label pods my-pod color-` removes the `color` label.

Debugging

- `kubectl logs <pod-name>`
- `kubectl logs <pod-name> -c <container-name>`
- `kubectl logs -f` follows the logs, i.e. continuously streaming to terminal.
- `kubectl exec -it <pod-name> --bash`

Debugging (cont.)

- Copy files:
  - `kubectl cp < pod-name >:</path/to/remote/file> </path/to/ local/file>`
- To access the pod via the network, you can use `port-forward` to forward network traffic. This makes a secure tunnel to containers that might not be exposed in the public network.
  - `kubectl port-forward <pod-name> 8080:80` forwards traffic from the port 8080 in the local machine to the remote container on port 80.

Resource utilization

- `kubectl top nodes`
- `kubectl top pods` (need `--all-namespaces` to see resources in the whole cluster).