# Get information about resources

* List resources of a specific type:
  + In a specific namespace:
    - Kubectl get <resource-type> -n <namespace>
  + In all namespaces:
    - Kubectl get <resource-type> -n --all-namespaces
* Get a specification of a specific resource in YAML format:
  + Kubectl get <resource-type> <resource-name> -o yaml
* Get values for specific keys from the YAML manifest of a resource. In this example we take values corresponding to the spec > tolerations key:
  + Kubectl get … -o jsonpath=’{.spec.tolerations}’

# Create, edit and modify resources

* Delete a resource:
  + Kubectl delete <resource-type> <resource-name>
* Edit a resource:
  + Using a chosen editor in terminal (nano in this example):
    - EDITOR=nano kubectl edit <resource-type> <resource-name>
  + Edit a resource by executing a single command and providing a pattern what to change:
    - Kubectl patch <resource-type> <resource-name> --patch '<patch-data>' [options]
* Deploy a resource using its YAML manifest:
  + kubectl apply -f path\_to\_yaml\_manifest

# Get logs

* Get logs from a resource:
  + Kubectl logs <resource-type> <resource-name>
* Describe a resource (there is for example a field ‘events’ with logs):
  + Kubectl describe <resource-type> <resource-name>
* Get events for a namespace:
  + kubectl get events -n <namespace> --sort-by=.lastTimestamp

# CoreDNS

* iptables – check firewall rules. We can use it to check if rules for CoreDNS and kube-proxy are set up properly.

# Pods

* Get access to a bash session in a running Pod:
  + kubectl exec -it <pod-name> -- /bin/bash
* Create a Pod, get access to a bash session in it and delete the Pod after exiting that bash session:
  + kubectl run -rm -it <pod-name> --image=your-docker-image -- /bin/bash

# Deployments

* Restart deployment (might be needed after making changes):
  + kubectl rollout restart deployment <resource-name>

# Creating a Pod

* Create a Pod using a given Docker image, get access to its terminal and remove it once we exit its terminal:
  + kubectl run pod-name --rm -it --image=container-registry-URL-to-image --bin/bash
* Create a Pod and keep it alive so we can check logs of that Pod:
  + kubectl run pod-name --image= container-registry-URL-to-image --restart=Never --command -- sleep 3600
* We can use the –override option to override some of the specifications of the created Pod. Below example for example specifies what container will be ran: A screen shot of a computer program

  AI-generated content may be incorrect.

# Helm

* Install dependencies from our chart (run it from the chart directory):
  + helm dependency update
* Install helm chart (deploy all the templates). This command will deploy all the resources in the specified namespace and create it if it doesn’t exist yet:
  + Helm install <release-name> ./path\_to\_chart\_directory --namespace=<namespace> --create-namespace
* Once we installed a chart and we made some change in one of the template files, we can update our chart by running:
  + helm upgrade <release-name> ./path\_to\_chart\_directory