Certification Tips - Imperative Commands with Kubectl

While you would be working mostly the declarative way - using definition files, imperative commands can help in getting one time tasks done quickly, as well as generate a definition template easily. This would help save a considerable amount of time during your exams.

Before we begin, familiarize with the two options that can come in handy while working with the below commands:

--dry-run: By default as soon as the command is run, the resource will be created. If you simply want to test your command, use the --dry-run=client option. This will not create the resource, instead, tell you whether the resource can be created and if your command is right.

-o yaml: This will output the resource definition in YAML format on the screen.

Use the above two in combination to generate a resource definition file quickly, that you can then modify and create resources as required, instead of creating the files from scratch.

POD

**Create an NGINX Pod**

kubectl run nginx --image=nginx

**Generate POD Manifest YAML file (-o yaml). Don't create it(--dry-run)**

kubectl run nginx --image=nginx  --dry-run=client -o yaml

Deployment

**Create a deployment**

kubectl create deployment --image=nginx nginx

**Generate Deployment YAML file (-o yaml). Don't create it(--dry-run)**

kubectl create deployment --image=nginx nginx --dry-run=client -o yaml

**IMPORTANT:**

kubectl create deployment does not have a --replicas option. You could first create it and then scale it using the kubectl scale command.

**Save it to a file - (If you need to modify or add some other details)**

kubectl create deployment --image=nginx nginx --dry-run=client -o yaml > nginx-deployment.yaml

You can then update the YAML file with the replicas or any other field before creating the deployment.

Service

**Create a Service named redis-service of type ClusterIP to expose pod redis on port 6379**

kubectl expose pod redis --port=6379 --name redis-service --dry-run=client -o yaml

(This will automatically use the pod's labels as selectors)

Or

kubectl create service clusterip redis --tcp=6379:6379 --dry-run=client -o yaml  (This will not use the pods labels as selectors, instead it will assume selectors as **app=redis.**[You cannot pass in selectors as an option.](https://github.com/kubernetes/kubernetes/issues/46191)So it does not work very well if your pod has a different label set. So generate the file and modify the selectors before creating the service)

**Create a Service named nginx of type NodePort to expose pod nginx's port 80 on port 30080 on the nodes:**

kubectl expose pod nginx --port=80 --name nginx-service --type=NodePort --dry-run=client -o yaml

(This will automatically use the pod's labels as selectors, but you cannot specify the node port. You have to generate a definition file and then add the node port in manually before creating the service with the pod.)

Or

kubectl create service nodeport nginx --tcp=80:80 --node-port=30080 --dry-run=client -o yaml

(This will not use the pods labels as selectors)

Both the above commands have their own challenges. While one of it cannot accept a selector the other cannot accept a node port. I would recommend going with the `kubectl expose` command. If you need to specify a node port, generate a definition file using the same command and manually input the nodeport before creating the service.

**Reference:**

<https://kubernetes.io/docs/reference/kubectl/conventions/>