Kubectl run is used to create pods, don't forget –restart=Never

```
// creating a pod
kubectl run nginx --image=nginx --restart=Never// List the pod
kubectl get po
```

You can create resources with yaml

```
// get the yaml file with --dry-run flag
kubectl run nginx --image=nginx --restart=Never --dry-run -o
yaml > nginx-pod.yaml// cat nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}// create a pod

kubectl create -f nginx-pod.yaml
```

You can output yaml with -o yaml flag

```
kubectl get po nginx -o yaml
```

You can delete based on file or name

```
kubectl delete po nginx
kubectl delete -f nginx-pod.yaml
```

You can set images on object with set image command

```
kubectl set image pod/nginx nginx=nginx:1.17.1

kubectl get po nginx -o
jsonpath='{.spec.containers[].image}{"\n"}'
```

You can exec into pods with kubectl exec

```
kubectl exec -it nginx /bin/sh
```

You can use – separator to run commands inside a pod

```
kubectl run busybox --image=busybox --restart=Never – ls
kubectl logs busybox
```

You can add multiple containers in yaml file

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: busybox
  name: busybox
spec:
  containers:
  - args:
    - bin/sh
    - -c
    - ls; sleep 3600
    image: busybox
    name: busybox1
    resources: {}
  - args:
    - bin/sh
    - -c
    - echo Hello world; sleep 3600
    image: busybox
    name: busybox2
    resources: {}
  - args:
    - bin/sh
    - -c
    - echo this is third container; sleep 3600
    image: busybox
    name: busybox3
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
```

You can use the -c flag to specify container to execute command on

```
kubectl exec busybox –c busybox2 – ls
```

You can mount file through volumemounts

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: multi-cont-pod
  name: multi-cont-pod
spec:
```

```
  volumes:
  - name: var-logs
    emptyDir: {}
  containers:
  - image: busybox
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo 'Hi I am from Main container' >>
/var/log/index.html; sleep 5;done"]
    name: main-container
    resources: {}
    volumeMounts:
    - name: var-logs
      mountPath: /var/log
  - image: nginx
    name: sidecar-container
    resources: {}
    ports:
      - containerPort: 80
    volumeMounts:
    - name: var-logs
      mountPath: /usr/share/nginx/html
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
```

You can create deployments with the create deployment command

```
kubectl create deploy webapp --image=nginx --dry-run -o yaml > webapp.yaml// change the replicas to 5 in the yaml and create it kubectl create -f webapp.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: webapp
  name: webapp
spec:
  replicas: 5
  selector:
    matchLabels:
      app: webapp
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
```

```
        app: webapp
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

You can scale deployment and list status of rollout

```
kubectl scale deploy webapp --replicas=20kubectl get po -l
app=webapp
```

```
kubectl rollout status deploy webapp
```

You can create jobs and cronjobs

```
kubectl create job hello-job --image=busybox --dry-run -o yaml --
- echo "Hello I am from job" > hello-job.yaml// edit the yaml
file to add completions: 10
kubectl create -f hello-job.yaml
```

```
apiVersion: batch/v1
kind: Job
metadata:
  creationTimestamp: null
  name: hello-job
spec:
  completions: 10
  template:
    metadata:
      creationTimestamp: null
    spec:
      containers:
      - command:
        - echo
        - Hello I am from job
        image: busybox
        name: hello-job
        resources: {}
      restartPolicy: Never
status: {}
```

```
kubectl create cronjob date-job --image=busybox --schedule="*/1
* * * *" -- bin/sh -c "date; echo Hello from kubernetes cluster"
```

Volumes and PersistentClaims are create with yaml files.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Configmaps can be created with create cm command and –from-literal

```
// first create a configmap cfgvolume
kubectl create cm cfgvolume --from-literal=var1=val1 --from-
literal=var2=val2// verify the configmap
kubectl describe cm cfgvolume// create the config map
kubectl create -f nginx-volume.yml// exec into the pod
kubectl exec -it nginx -- /bin/sh// check the path
cd /etc/cfg
ls
```

You can mount configmaps into directories

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  volumes:
  - name: nginx-volume
    configMap:
      name: cfgvolume
  containers:
  - image: nginx
    name: nginx
    resources: {}
    volumeMounts:
    - name: nginx-volume
      mountPath: /etc/cfg
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
```