- Command for making a pod and deploy container using nginx docker image
    - Kubectl run nginx –image=nginx

- Command to check the status of pods
    - Kubectl get pods

- Command to get a specific pod
    - Kubectl get pods name_of_pod

- Command for detail information of pods
    - Kubectl describe pod nginx

- Command to check the status of the pod
    - Kubectl get pods -o wide

- Command to make pods from a YAML configuration file.
    - Kubectl create -f “name of YAML file”.yml

      Or

      Kubectl apply -f “name of YAML file”.yml

- Command to check number of nodes in a custer
    - Kubectl get nodes

- Command to check the status of node with slight more detail
    - Kubectl get node -o wide

- Command to check the image used to create a container in pod
    - Kubctl describe pod name_of_pod | grep -i image

- Another way to create a pod using YAML file
    - Kubectl run redis –image=redis123 –dry-run=client -o yaml > pod.yaml

- Command to see the list of replication controllers created
    - Kubectl get replicationcontroller

- Command to get a list of all replica sets
    - Kubectl get replicaset

- Command to execute existing replicaset file after changes
    - Kubectl replace -f “name_of_changed_replicaset”.yml

- Command to delete a replicaset with all its underlying pods
    - Kubectl delete replicaset “name_of_replicaset”

- Command to scale replica’s to 6 if they were 3 in beginning
    - Kubectl scale –replicas=6 -f replicaset-definition.yml

      or

    - Kubectl scale replicaset myapp-replicaset –replicas=6

- Command to edit a replicaset file in terminal
    - Kubectl edit replicaset myapp-replicaset

- Command to get the list of deployments
    - Kubectl get deployments

- Command to get deployment, replicaset and pods list at once or in another words to get all objects in a cluster
    - Kubectl get all

- Command to see information about rollout
    - Kubectl rollout status deployment/mydeployment

- Command to see history of revisions in rollout
    - Kubectl rollout history deployment/mydeployment

- Command to undo rolling updates in other words to rollback current changes
    - Kubectl rollout undo deployment/mydeployment

- Instruction to record the cause of change in rollout
    - Kubectl create -f deployment.yaml –record

- Command to change an image in deployment
    - Kubectl set image deployment myapp-deployment nginx= nginx:1.18-pearl –record

- Command to get the IP of a service in minikube
    - Minikube service myapp-service –url

- Command to make a service from the terminal
    - Kubectl expose deployment simple-webapp-deployment –name=webapp-service –targetPort=8080 –type=NodePort –port=8080 –nodePort=30080 –dry-run=client -o yaml > svc.yaml

- Command to get information regarding clusters
    - Kubectl clusters-info

- Command to get list of nodes in a cluster
    - Kubectl get nodes

- Command to create a deployment from terminal
    - Kubectl create deployment httpd-frontend –image=httpd:2.4-alpine

      or
    - Kubectl create deployment –image=redis redis

- Command to see pods in a specific namespace
    - Kubectl get pods –namespace=”name_of_any_namespace”

      or
    - Kubectl -n “name_of_namespace” get pods

      or
    - Kubectl -n “name_of_namespace” get pods –no-headers

- Command to create a pod in a specific namespace
    - Kubectl create -f “pod_definition_file”.yaml –namespace=”name_of_namespace”

- Command to create a namespace
    - First way is by using commands in the terminal
        - Kubectl create namespace “name_of_namespace”
    - 2nd way is by using namespace definition file
        - Kubectl create -f “yaml_definition_file”.yaml

- Command to set the current default namespace to some other namespace
    - Kubectl config set-context $(kubectl config current-context) –namespace=”name_of_the_namespace”

- Command to see pods in all namespaces
    - Kubectl get pods –all-namespaces

- Command to get list of namespaces
    - Kubectl get namespaces

      or
    - Kubectl get ns

- Command to access a specific service in another namespace
    - Db-service.dev.svc.cluster.local

- Command to get the number of pods in all namespaces without headers
    - Kubectl get ns –no-headers | wc -l

- Command to get a specific pod in any namespace
    - Kubectl get pods –all-namespaces | grep blue

- Command to get services in a specific namespace
    - Kubectl -n “name_of_namespace” get svc

- Command to 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. 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)

- Command to 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 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.

- Command to create a pod using image and labels in terminal
    - kubectl run redis --image=redis-alpine --labels="tier=db"

- Command to create a configmap using imperative way
    - Kubectl create configmap "name_of_configmap" –from-literal=key=value

- Command to create a configmap for multiple key value pairs
    - Kubectl create configmap "name_of_configmap" \
      –from-literal=key=value \
      –from-literal=key=value \
      –from-literal=key=value

- Command to create configmap from a file using imperative way
    - Kubectl create configmap "name_of_configmap" \
      –from-file=app_config.properties

- Command to get the list of configmaps
    - Kubectl get configmaps

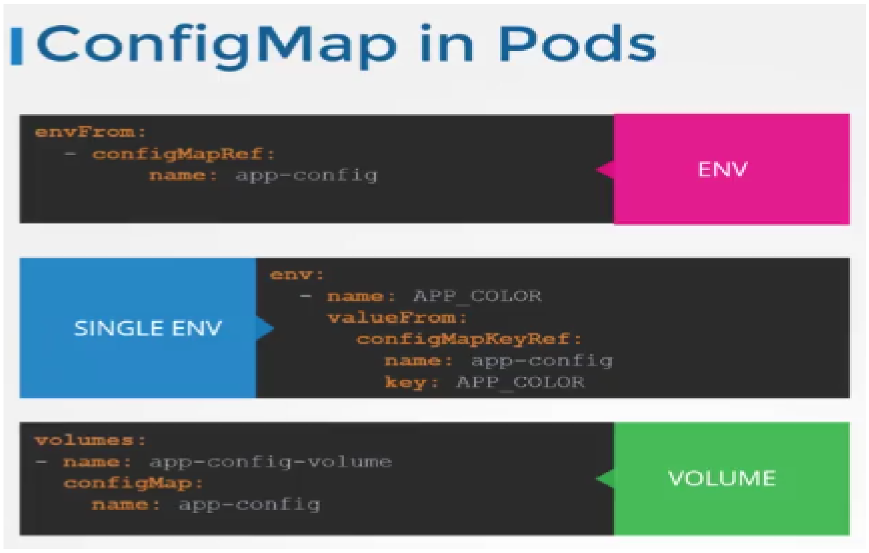
- Command to get extra information regarding configmaps
    - Kubectl describe configmap "name_of_configmap"

- Injecting created configmap file in pod definition file
    - Containers:

```
    - name: nginx
     image: nginx
     ports:
        - containerPort: 8080
      envFrom:
```

                    - configMapRef:
                        name: “name_of_configMap”

- *Different ways to inject configmap data into pods*



ConfigMap in Pods

```
envFrom:
  - configMapRef:
      name: app-config
```

```
env:
  - name: APP_COLOR
    valueFrom:
      configMapKeyRef:
        name: app-config
        key: APP_COLOR
```

```
volumes:
- name: app-config-volume
  configMap:
    name: app-config
```

- Command to get info about some configuration in definition files
    - Kubectl explain pods –recursive | grep envFrom -A3

- Command to create secret in imperative way
    - Kubectl create secret generic “secret_name” –from-literal=key=value
    - Kubectl create secret generic app_secret –from-literal=DB_HOST=mysql

- Command to create a secret from a file in imperative way
    - Kubectl create secret generic “secret_name” –from-file=”path_to_file”
    - Kubectl create secret generic app_secret –from-file=app_secret.properties

- Command to convert plain text to hashed form in linux terminal
    - Echo -n ‘mysql’ | base64

- Command to get list of secrets
    - Kubectl get secrets

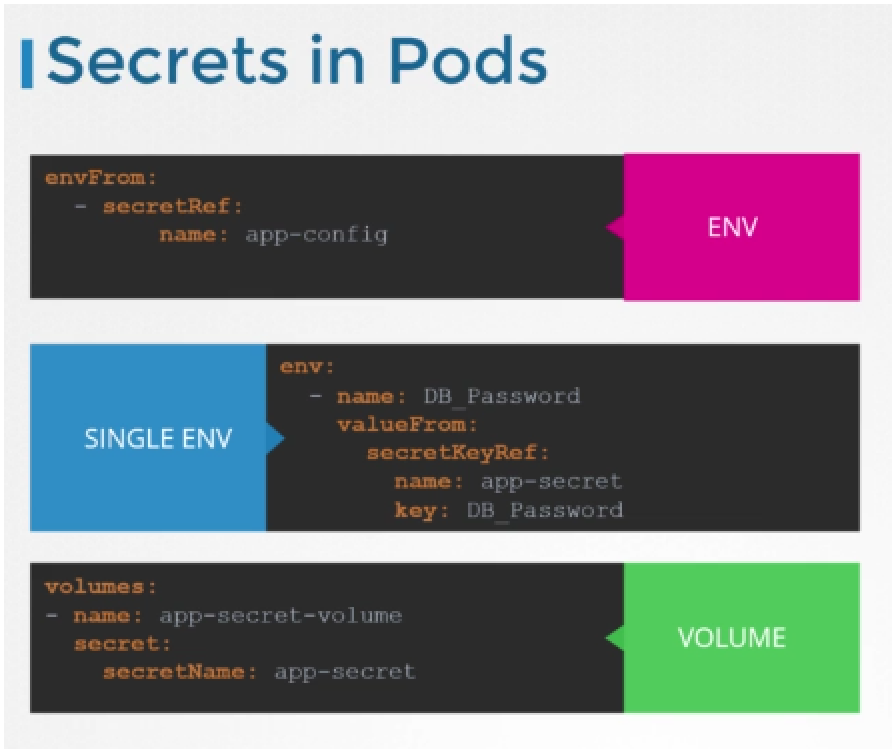
- Command to see additional information about secrets
    - Kubectl describe secret “name_of_secret”

- Command to decode hashed values to a plain text
    - Echo -n “value_to_decode” | base64 –decode

- Injecting created secret file to pod definition file
    - Containers:

```
- name: nginx
  image: nginx
  ports:
     - containerPort: 8080
   envFrom:
     - secretRef:
          name: “name_of_secret”
```

- Different ways to inject secrets into pod definition files



Secrets in Pods

```
envFrom:
  - secretRef:
      name: app-config
```

```
env:
  - name: DB_Password
    valueFrom:
      secretKeyRef:
        name: app-secret
        key: DB_Password
```

```
volumes:
- name: app-secret-volume
  secret:
    secretName: app-secret
```

- Command to find the user of a pod whom runs the pod
    - Kubectl exec "name_of_pod" – whoami

- Instructions to add security context at a pod level
    - spec:

```
containers:
securityContext:
    runAsUser: root
```

- Instructions to add security context at a container level
    - Containers:

```
- name: nginx
  image: nginx
  ports:
     - containerPort: 8080
   securityContext:
   runAsUser: shamsi
   Capabilities:
       Add: ["name_of_capability"]
```

SERVICE ACCOUNT

- Command to create a service account
    - Kubectl create serviceaccount "name_of_service_account"

- Command to get a list of service accounts
    - Kubectl get serviceacount

- Command to get additional information of a service account
    - Kubectl describe serviceaccount "name_of_serviceaccount"

- Command to see a secret token of secretaccount
    - Kubectl describe secret "name_of_token"

- Instructions to add service account to pod definition
    - spec:

```
Containers:
   - name: nginx
     image: nginx
     ports:
        - containerPort: 8080
      serviceAccountName: “name_of_service_account”
```

TAINTS AND TOLERATIONS

- Command to apply taints to the nodes.
  - Kubectl taint nodes “node_name” key=value:taint-effect

  - *Note: values of taint-effect can be*
    - *NoSchedule*
    - *PreferNoSchedule*
    - *NoExecute*
  - *E.g → kubectl taint nodes node1 app=blue: NoSchedule*

- Instructions in Pod definition file

  -
    ```
    spec:
       tolerations:
         -  key: “app”
            operator: “Equal”
            value: “blue”
            effect: “NoSchedule”
    ```

- Command to see the taint of a master node
  - Kubectl describe node kubemaster | grep Taint
- Command to untaint a tainted node
  - Kubectl taint nodes “node_name” “taint”

E.g → kubectl taint nodes controlplane node-role.kubernetes.io/master:NoSchedule-

- Command to check all options available for tolerations
  - Kubectl explain pod –recursive | less

- Command to print the values underneath a certain option
  - Kubectl explain pod –recursive | grep -A5 tolerations

NODE SELECTORS

- Instructions to deploy a pod to a specific node using node selectors

    -
      ```
      spec:
         containers:
           -   name: data-processor
               Image: data-processor

         nodeSelector:
            Size: Large
      ```

- Command to label a node
    - Kubectl label nodes “node_name” key=value
    - *E.g → **kubectl label nodes node01 size=Large***

NODE AFFINITY

- Instructions to deploy a pod to a specific node using node affinity

    -
      ```
      spec:
         containers:
            - name: data-processor
              Image: data-processor

         affinity:
            nodeAffinity:
               requiredDuringSchedulingIgnoredDuringExecution:
                  nodeSelectorTerms:
                     - matchExpressions:
                          - key: size
                            Operator: In
                            Values:
                                - Large
      ```

    Note → the value of operator can be

    - *In*
    - *NotIn*
    - *Exists (when used, no need to specify value)*

- Command to show labels that are on nodes
    - Kubectl get nodes node01 –show-labels

MULTI CONTAINER PODS

- Command to get pods and services in a namespace
    - Kubectl -n elastic-stack get pod,svc

- Command to get logs of an app
    - Kubectl -n elastic-stack logs app

LIVENESS AND READINESS PROBES

- Instructions for liveness and readiness probes

    -
      ```
      spec:
        readinessProbe:
          httpGet:
            Path: /live
            Port: 8080
        initialDelaySeconds: 80
        periodSeconds: 1
        failureThreshold: 8

        livenessProbe:
          httpGet:
             path: /live
             port: 80
         initialDelaySeconds: 80
        periodSeconds: 1
         failureThreshold: 8
      ```

CONTAINER LOGS

- Command to get container logging
    - Kubectl logs -f “name_of_pod” “name_of_container”
    - *E.g → kubectl logs -f event-simulator-pod nginx*

- Command to get logs regarding a specific user
    - Kubectl logs webapp-1 | grep USER5

- Command for running logs on a specific container if a pod has more than 1 container
    - Kubectl logs webapp-2 -c “name_of_container”

KUBERNETES CLUSTER LOGS

- Command to clone metrics server repo for monitoring
    - git clone https://github.com/kodekloudhub/kubernetes-metrics-server.git

- Command to create all the components in this repo → First head into downloaded repo
    - cd kubernetes-master-server
    - Kubectl create -f .

- Command to get metrics data about nodes
    - Kubectl top node

- Command to get metrics data about pods
    - Kubectl top pod

LABELS, SELECTORS AND ANNOTATIONS

- Command to select pods with specific labels
    - Kubectl get pods –selector key=value
    - E.g → kubectl get pods –selector app=App1

- Command to get all objects of a specific label
    - Kubectl get all –selector env=prod

- Command to get objects based on multiple labels
    - Kubectl get all –selector dev=prod,bu=finance,tier=frontend

- Command to show labels of pods on terminal

    - Kubectl get pods –show-labels

- Command to filter pods using a specific label
    - Kubectl get pods -l env=dev

- Command to get a count of pods having a specific label
    - Kubectl get pods -l env=dev –no-headers | wc -l

- Instructions to add labels, selectors and annotations in definition file

    -
      ```
      apiVersion: apps/v1
      kind: ReplicaSet
      metadata:
        name: simple-webapp
        labels:
          app: App1
          functions: Front-end
      spec:
        replicas: 3
        selector:
          matchLabels:
            app: App1
        template:
          metadata:
            labels:
              app: App1
              Function: Front-end
          spec:
            containers:
             -  name: simple-webapp
                Image: simple-webapp
      ```

JOBS AND CRON JOBS

- Instructions to create a job from a definition file

    -
      ```
      apiVersion: batch/v1
      kind: Job
      metadata:
         name: math-add-job
      spec:
         template:
            spec:
               containers:
                  - name: math-add
                    image: ubuntu
                    command: [‘expr’, ‘3’, ‘+’, ‘2’]
               restartPolicy: Never
      ```

- Command to see the list of jobs
    - Kubectl get jobs

- Command to see the output of a computation of a pod
    - Kubectl logs “name_of_pod”
    - *E.g → kubectl logs math-add-job-1d87pn*

- Command to delete a job
    - Kubectl delete job “name_of_job”

- Instructions to create multiple pods through a job

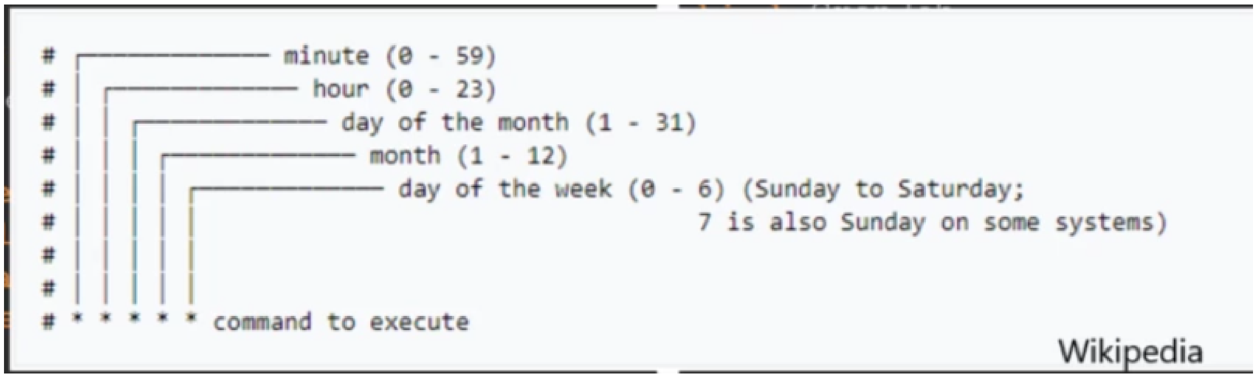
    -
      ```
      apiVersion: batch/v1
      kind: Job
      metadata:
         name: math-add-job
      spec:
         completions: 3
         parallelism: 3
         template:
            spec:
               containers:
                  - name: math-add
      ```

```
                    image: ubuntu
                    command: [‘expr’, ‘3’, ‘+’, ‘2’]
                 restartPolicy: Never
```

- Instructions to create cron job from a definition file

    -
      ```
      apiVersion: batch/v1beta1
      kind: CronJob
      metadata:
        name: reporting-cron-job
      spec:
        Schedule: “*/1 * * * *”
        jobTemplate:
          spec:
            completions: 3
            parallelism: 3
            template:
              spec:
                containers:
                  - name: math-add
                   image: ubuntu

                restartPolicy: Never
      ```



```
# ┌───────────── minute (0 - 59)
# │ ┌───────────── hour (0 - 23)
# │ │ ┌───────────── day of the month (1 - 31)
# │ │ │ ┌───────────── month (1 - 12)
# │ │ │ │ ┌───────────── day of the week (0 - 6) (Sunday to Saturday;
# │ │ │ │ │                                     7 is also Sunday on some systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * command to execute
```

- Command to get list of cron jobs

SERVICES

- Instructions to create a service definition file

    -
      ```
      apiVersion: v1
      kind: Service
      metadata:
        name: myapp-service
      spec:
        type: NodePort
        selector:
          app: myapp
          type: front-end
        port:
          - targetPort: 80
            port: 80
            nodePort: 30008
      ```

- Command to get list of services
    - Kubectl get services

- Command to get extra information about services
    - Kubectl describe service “name_of_service”

INGRESS NETWORKING

- Instructions for ingress controller definition file

    -
      ```
      apiVersion: apps/v1
      kind: Deployment
      metadata:
        name: nginx-ingress-controller
      spec:
      ```

```
replicas: 1
selector:
   matchLabels:
      Name: nginx-ingress
template:
   metadata:
      labels:
         name: nginx-ingress
   spec:
      args:
        - /nginx-ingress-controller
        - –configmap=$ (POD_NAMESPACE)/nginx-configuration
      env:
         - name: POD_NAME
           valueFrom:
              fieldRef:
                 fieldPath: metadata.name

         - name: POD_NAMESPACE
           valueFrom:
              fieldRef:
                 fieldPath: metadata.namespace

      ports:
         - name: http
           containerPort: 80

         - name: https
           containerPort: 443

      containers:
         - name: nginx-ingress-controller
           Image:
quay.io/kubernetes-ingress-controller/nginx-ingress-controller:0.21.0
```

- Instructions for creating service for ingress controller

    -
      ```
      apiVersion: v1
      kind: Service
      metadata:
         name: nginx-ingress
      spec:
      ```

```
type: NodePort
ports:
   - port: 80
     targetPort: 80
     protocol: TCP
     name: http
  - port: 443
    targetPort: 443
    protocol: TCP
    name: https
selector:
   name: nginx-ingress
```

- Instructions for creating configMap for ingress controller

    -
      ```
      apiVersion: v1
      kind: ConfigMap
      metadata:
         name: nginx-configuration
      ```

- Command to get all resources in all namespaces
    - Kubectl get all -A

- Command to get ingress in all namespaces
    - Kubectl get ingress –all-namespaces

-