Q1) Create the necessary roles and role bindings required for the dev-user to create, list and delete pods in the default namespace.

- Role: developer
- Role Resources: pods
- Role Actions: list
- Role Actions: create
- RoleBinding: dev-user-binding
- RoleBinding: Bound to dev-user

Ans:

Create Role:

vim role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["create", "list"]
```

kubectl create -f role.yaml

Create Rolebinding for the role

vim rolebinding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: dev-user-binding
  namespace: default
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: developer
  apiGroup: rbac.authorization.k8s.io
```

kubectl create -f rolebinding.yaml

to check if you can access the pods or not

```
kubectl get pods --as dev-user
```

Q2) Create a blue namespace Grant the **dev-user** permissions to **create deployments** in the role namespace.

Ans:

```
kubectl namespace role
```

Create Role:

```
vim role-1.yaml
```

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: role
  name: deploy-role-1
rules:
- apiGroups: ["apps", "extensions"]
  resources: ["deployments"]
  verbs: ["create"]
```

```
kubectl create -f role-1.yaml
```

Create Rolebinding for the role

```
vim rolebinding-1.yaml
```

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: dev-user-deploy-binding-1
  namespace: role
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: deploy-role-1
  apiGroup: rbac.authorization.k8s.io
```

```
kubectl create -f rolebinding-1.yaml
```

Create Deployment now in role namespace as dev-user

Q3) Create a pod called **secbusybox** with the image busybox which executes command sleep 3600 and makes sure any Containers in the Pod, all processes run with user ID 1000 and with group id 2000 and verify..

Ans:

```
vim secbusy.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: secbusybox
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 2000
  containers:
  command: [“sleep”,”3600”]
    image: busybox
    name: secbusybox
```

```
kubectl create -f secbusy.yaml
```

OR

```
// create yml file with dry-run this will create YAML file without running it
kubectl run secbusybox --image=busybox --dry-run -o yaml -- /bin/sh -c "sleep 3600;" > busybox.yml
//edit YAML file like above and create
kubectl create -f busybox.yml
```

// verify

```
kubectl exec -it secbusybox -- sh
id // it will show the id and group
```

Q4) Create pod with an nginx image and configure the pod with capabilities NET_ADMIN and SYS_TIME verify the capabilities.

Ans:

```
vim nginx.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    securityContext:
      capabilities:
        add: ["SYS_TIME", "NET_ADMIN"]
    name: nginx
```

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

OR

```
// create yml file with dry-run this will create YAML file without running it
kubectl run nginx --image=nginx --dry-run -o yaml > nginx.yml
//edit YAML file like above and create
kubectl create -f nginx.yml
```

// verify

```
kubectl exec -it nginx -- sh
cd /proc/1
cat status
// you should see these values
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb
```

Q5) Create a Pod nginx and specify a memory request and a memory limit of 100Mi and 200Mi respectively.

Ans:

```
vim pod-limit.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
```

```
  - image: nginx
    name: nginx
    resources:
      requests:
        memory: "100Mi"
      limits:
        memory: "200Mi"
```

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

OR

```
// create a yml file
kubectl run nginx --image=nginx --dry-run -o yaml > pod-limit.yml
// add the resources section and create
kubectl create -f pod-limit.yml
```

verify

```
kubectl top pod
```

Q6) Create a Pod nginx and specify a CPU request and a CPU limit of 0.5 and 1 respectively.

Ans:

```
vim pod-request.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources:
      requests:
        cpu: "0.5"
      limits:
        cpu: "1"
```

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

OR

```
// create a yml file
kubectl run nginx --image=nginx --dry-run -o yaml > pod-request.yml
// add the resources section and create
kubectl create -f pod-request.yml
```

verify

```
kubectl top pod
```

Q7) Create a NetworkPolicy which denies all ingress traffic

Ans:

```
vim policy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
  - Ingress
kubectl create -f policy.yaml
```

Q8) List all the configmaps in the cluster

Ans:

```
kubectl get cm
    or
kubectl get configmap
```

Q9) Create a configmap called myconfigmap with literal value appname=myapp

Ans:

```
kubectl create cm myconfigmap --from-literal=appname=myapp
```

Q10) Verify the configmap we just created has this data

Ans:

```
kubectl describe cm
```

Q11) delete the configmap myconfigmap we just created

Ans:

```
kubectl delete cm myconfigmap
```

Q12) Create a file called config.txt with two values key1=value1 and key2=value2 and verify the file

Ans:

```
cat >> config.txt << EOF
key1=value1
key2=value2
EOF
```

Q13) Create a configmap named keyvalcfgmap and read data from the file config.txt and verify that configmap is created correctly

Ans:

```
kubectl create cm keyvalcfgmap --from-file=config.txt
kubectl get cm keyvalcfgmap -o yaml
```

Q14) Create an nginx pod and load environment values from the above configmap keyvalcfgmap and exec into the pod and verify the environment variables and delete the pod

Ans:

```
vim config.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
```

```
  envFrom:
  - configMapRef:
      name: keyvalcfgmap
```

```
kubectl create -f config.yaml
```

```
// first run this command to save the pod yml
kubectl run nginx --image=nginx --dry-run -o yaml > config.yml
// edit the yml to below file and create
kubectl create -f config.yml
```

// verify

```
kubectl exec -it nginx -- env
kubectl delete pod nginx
```