# Solution M3: Security and Policies

For this challenge we will assume that we are working as the **root** user on the control plane node of a three node **Kubernetes** cluster with **Calico** installed as a pod network plugin. We will execute all tasks in order

# Task 1

**Challenge:**

Create and register two Kubernetes uses – **Ivan** (**ivan**) and **Mariana** (**mariana**) who are part of the **Gurus** (**gurus**) group

**Solution:**

Create the two users

**useradd -m -c 'Ivan' -s /bin/bash ivan**

**useradd -m -c 'Mariana' -s /bin/bash mariana**

Prepare the subfolders for both users

**mkdir -p /home/{ivan,mariana}/{.certs,.kube}**

Create the private keys for both users

**openssl genrsa -out /home/ivan/.certs/ivan.key 2048**

**openssl genrsa -out /home/mariana/.certs/mariana.key 2048**

Create a certificate signing request (**CSR**) for every user

**openssl req -new -key /home/ivan/.certs/ivan.key -out /home/ivan/.certs/ivan.csr -subj "/CN=ivan/O=gurus"**

**openssl req -new -key /home/mariana/.certs/mariana.key -out /home/mariana/.certs/mariana.csr -subj "/CN=mariana/O=gurus"**

Sign both **CSRs** with the **Kubernetes CA** certificate

**openssl x509 -req -in /home/ivan/.certs/ivan.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out /home/ivan/.certs/ivan.crt -days 365**

**openssl x509 -req -in /home/mariana/.certs/mariana.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out /home/mariana/.certs/mariana.crt -days 365**

Create the credentials for the two users

**kubectl config set-credentials ivan --client-certificate=/home/ivan/.certs/ivan.crt --client-key=/home/ivan/.certs/ivan.key**

**kubectl config set-credentials mariana --client-certificate=/home/mariana/.certs/mariana.crt --client-key=/home/mariana/.certs/mariana.key**

Create the contexts for the two users

**kubectl config set-context ivan-context --cluster=kubernetes --user=ivan**

**kubectl config set-context mariana-context --cluster=kubernetes --user=mariana**

Create configuration file for **ivan** at **/home/ivan/.kube/config** with the following content:

apiVersion: v1

clusters:

- cluster:

    certificate-authority: /etc/kubernetes/pki/ca.crt

    server: https://192.168.81.221:6443

  name: kubernetes

contexts:

- context:

    cluster: kubernetes

    user: ivan

  name: ivan-context

current-context: ivan-context

kind: Config

preferences: {}

users:

- name: ivan

  user:

    client-certificate: /home/ivan/.certs/ivan.crt

    client-key: /home/ivan/.certs/ivan.key

*Please note that you should adjust some parameters (cluster name, server URL, etc.) to match yours*

Create configuration file for **mariana** at **/home/mariana/.kube/config** with the following content:

apiVersion: v1i

clusters:

- cluster:

    certificate-authority: /etc/kubernetes/pki/ca.crt

    server: https://192.168.81.221:6443

  name: kubernetes

contexts:

- context:

    cluster: kubernetes

    user: mariana

  name: mariana-context

current-context: mariana-context

kind: Config

preferences: {}

users:

- name: mariana

  user:

    client-certificate: /home/mariana/.certs/mariana.crt

    client-key: /home/mariana/.certs/mariana.key

*Please note that you should adjust some parameters (cluster name, server URL, etc.) to match yours*

Now, change the ownership of the respective folders and files

**chown -R ivan: /home/ivan**

**chown -R mariana: /home/mariana**

We are done for now. Each one of them will be able to use **kubectl** but without much success. We are about to change this

# Task 2

**Challenge:**

Create a **namespace** named **projectx**

**Solution:**

This one is as simple as executing this command (with user that has the right)

**kubectl create ns projectx**

# Task 3

**Challenge:**

Create a **LimitRange** for the namespace to set **defaults**, **minimum** and **maximum** both for **CPU** and **memory**

**Solution:**

Create a **limits.yaml** file with the following (sample) content

apiVersion: v1

kind: LimitRange

metadata:

  name: projectx-limits

spec:

  limits:

  - max:

      cpu: 500m

      memory: 500Mi

    min:

      cpu: 50m

      memory: 50Mi

    default:

      cpu: 100m

      memory: 100Mi

    type: Container

Deploy it to the cluster with

**kubectl apply -f limits.yaml -n projectx**

And check its status

**kubectl describe limitrange -n projectx**

# Task 4

**Challenge:**

Create a **ResourceQuota** for the namespace to set **requests** and **limits** both for **CPU** and **memory** (use values that you consider suitable). In addition, add limits for **pods**, **services**, **deployments**, and **replicasets** (again, use values that you find appropriate)

**Solution:**

Create a **quotas.yaml** file with the following (sample) content

apiVersion: v1

kind: ResourceQuota

metadata:

  name: projectx-quotas

spec:

  hard:

    requests.cpu: 1

    requests.memory: 1Gi

    limits.cpu: 2

    limits.memory: 2Gi

    pods: 10

    services: 5

    count/deployments.apps: 2

    count/replicasets.apps: 2

As you can see, there two “non-standard” lines – starting with **count/**

They should have the form **count/<resource>.<group>**

Deploy the manifest to the cluster

**kubectl apply -f quotas.yaml -n projectx**

And check its status

**kubectl describe quota -n projectx**

# Task 5

**Challenge:**

Create a custom role (**devguru**) which will allow the one that has it to do anything with any of the following resources **pods**, **services**, **deployments**, and **replicasets**. Grant the role to **ivan** and **mariana** (or to the group they belong to) for the namespace created earlier

**Solution:**

We can create the required role with

**kubectl create role role-hw -n projectx --verb="\*" --resource=pods,services,deployments.apps,replicasets.apps**

Then check what we have created

**kubectl describe role role-hw -n projectx**

Now, we can bind the role either to each user individually *(use either this one)*

**kubectl create rolebinding roleb-hw-ivan -n projectx --role role-hw --user ivan**

**kubectl create rolebinding roleb-hw-mariana -n projectx --role role-hw --user mariana**

And then check what we can do as **ivan**

**kubectl auth can-i --list --namespace projectx --as ivan**

Or create a role binding to the group *(or this one)*

**kubectl create rolebinding roleb-hw-gurus -n projectx --role role-hw --group gurus**

And then check what we can do as **ivan** being a member of the **gurus** group

**kubectl auth can-i --list --namespace projectx --as ivan --as-group gurus**

# Task 6

**Challenge:**

Using one of the two users, deploy the **producer-consumer** application (use the attached files – you may need to modify them a bit). Deploy one additional pod that will act as a (curl) **client**

**Solution:**

Switch the session as **ivan**

**su ivan**

Then deploy the provided manifests

**kubectl apply -n projectx -f producer-deployment.yml**

**kubectl apply -n projectx -f producer-svc.yml**

**kubectl apply -n projectx -f consumer-deployment.yml**

**kubectl apply -n projectx -f consumer-svc.yml**

Then, deploy a simple pod to act as a client for testing purposes

**kubectl run client -n projectx --image alpine -- sleep 1d**

Finally, check how the deployment went

**kubectl get pods,svc,rs,deployments -n projectx**

# Task 7

**Challenge:**

Create one or more **NetworkPolicy** resources in order to

1. Allow communication to the **producer** only from the **consumer**
2. Allow communication to the **consumer** only from the **client**

**Solution:**

As our two users (**ivan** and **mariana**) does not have rights to create network policies, next part we must execute as the **admin** user

First, let’s connect to the client pod

**kubectl exec -n projectx -it client -- sh**

And update and install the curl package

**apk update**

**apk add curl**

Then, try to talk to the consumer and then the producer services

**curl http://consumer:5000**

**curl http://producer:5000**

Now, close the session

**exit**

And then create the first network policy manifest **np-prod.yaml** with the following content

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

  name: np-prod

spec:

  podSelector:

    matchLabels:

      role: producer

  ingress:

  - from:

    - podSelector:

        matchLabels:

          role: consumer

Deploy it with

**kubectl apply -n projectx -f np-prod.yaml**

Then create the second network policy manifest **np-cons.yaml** with the following content

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

  name: np-cons

spec:

  podSelector:

    matchLabels:

      role: consumer

  ingress:

  - from:

    - podSelector:

        matchLabels:

          run: client

And deploy it with

**kubectl apply -n projectx -f np-cons.yaml**

Finally, connect to the client pod

**kubectl exec -n projectx -it client – sh**

And test again

**curl --connect-timeout 5 http://consumer:5000**

**curl --connect-timeout 5 http://producer:5000**

We manage to do all the tasks 😊