🔬 Lab: Create a Kubernetes User and Generate `kubeconfig`

🎯 Goal

Create a user `developer` using a client certificate and give them access to a specific namespace using RBAC.

🧰 Prerequisites

- A working Kubernetes cluster (e.g. Minikube, Kind)
- `openssl`, `kubectl`, and `cfssl` (optional) installed
- Cluster admin privileges (to create certs, roles, and secrets)

📁 Lab Structure

```
developer-lab/
├── certs/
│   ├── developer-csr.conf
│   ├── developer.csr
│   ├── developer.key
│   └── developer.crt
├── kubeconfig/
│   └── developer.kubeconfig
└── rbac/
    └── developer-role.yaml
```

1️⃣ Generate Certificate for the User

```
mkdir -p developer-lab/certs && cd developer-lab/certs
```

📄 `developer-csr.conf`

```
[req]
default_bits = 2048
prompt = no
default_md = sha256
distinguished_name = dn

[dn]
CN = developer
O = dev-team
```

Now generate:

```
openssl genrsa -out developer.key 2048

openssl req -new -key developer.key \
  -out developer.csr -config developer-csr.conf

# Approve and sign with cluster CA (update path as needed)
openssl x509 -req -in developer.csr \
  -CA /etc/kubernetes/pki/ca.crt \
  -CAkey /etc/kubernetes/pki/ca.key \
  -CAcreateserial \
  -out developer.crt -days 365
```

> 🔐 You now have `developer.crt` and `developer.key`.

2️⃣ Create a `kubeconfig` for the Developer

```
cd ../kubeconfig
CLUSTER_NAME=$(kubectl config view --minify -o jsonpath='{.clusters[0].name}')
CLUSTER_SERVER=$(kubectl config view --minify -o
jsonpath='{.clusters[0].cluster.server}')
CA_CERT=$(kubectl config view --raw --minify -o
jsonpath='{.clusters[0].cluster.certificate-authority-data}')

kubectl config set-cluster "$CLUSTER_NAME" \
  --certificate-authority=/etc/kubernetes/pki/ca.crt \
  --embed-certs=true \
  --server="$CLUSTER_SERVER" \
  --kubeconfig=developer.kubeconfig

kubectl config set-credentials developer \
  --client-certificate=../certs/developer.crt \
  --client-key=../certs/developer.key \
  --embed-certs=true \
  --kubeconfig=developer.kubeconfig

kubectl config set-context developer-context \
  --cluster="$CLUSTER_NAME" \
  --namespace=dev \
  --user=developer \
  --kubeconfig=developer.kubeconfig

kubectl config use-context developer-context --kubeconfig=developer.kubeconfig
```

✅ `developer.kubeconfig` is now ready.

3️⃣ Create Namespace and RBAC for the User

```
kubectl create namespace dev
```

📄 `rbac/developer-role.yaml`

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

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

Apply:

```
kubectl apply -f rbac/developer-role.yaml
```

4️⃣ Test as Developer

Use the new `kubeconfig`:

```
kubectl --kubeconfig=developer.kubeconfig get pods
```

Try creating a pod:

```
kubectl --kubeconfig=developer.kubeconfig run nginx --image=nginx
```

✅ Success! You're using Kubernetes as the `developer` user.

🧹 Cleanup

```
kubectl delete ns dev
kubectl delete -f rbac/developer-role.yaml
```

August 23, 2025