Use `openssl`, and create a private key named `emp.key`.

openssl genrsa -out emp.key 2048

Create a certificate sign request named `emp.csr` using the private key generated earlier.
Use the following information:

```
name :emp
group: dev
```

openssl req -new -key emp.key -out emp.csr -subj "/CN=emp/O=dev"

Generate `emp.crt` by approving the request created earlier.

openssl x509 -req -in emp.csr -CA CA_FOLDER/ca.crt -CAkey CA_FOLDER/ca.key -CAcreateserial -out emp.crt -days 500
Note: Change CA_FOLDER value by either ~/.minikube/. or /etc/kubernetes/pki/. based on your CA files location

Create a namespace named `dev`.

kubectl create namespace dev

Create a new context pointing to the cluster `minikube`, and name it `dev-ctx`. It should point to the namespace `dev`, and the user should be `emp`.

kubectl config set-context dev-ctx --cluster=minikube --namespace=dev --user=emp

Set credentials for `emp`.
Use `emp.key` and `emp.crt` created earlier.

kubectl config set-credentials emp --client-certificate=./emp.crt --client-key=./emp.key
Run cat ~/.kube/config. You should see above added information in that config file.

Create a role named `emp-role`, and assign `get`, `list` access on `pods` and `deployments` (use `dev` namespace).

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  namespace: dev
  name: emp-role
rules:
- apiGroups: ["", "extensions", "apps"]
  resources: ["deployments",  "pods"]
  verbs: ["get", "list"]
```

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  namespace: office
  name: ops-role
rules:
- apiGroups: ["", "extensions", "apps"]
  resources: ["deployments",  "pods"]
  verbs: ["get", "list", "watch"]
```

> Bind `emp` to the role `emp-role` created earlier, and name it `emp-bind`.

kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: emp-bind
  namespace: dev
subjects:
- kind: User
  name: emp
  apiGroup: ""
roleRef:
  kind: Role
  name: emp-role
  apiGroup: ""

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: ops-binding
  namespace: office
subjects:
- kind: User
  name: employee
  apiGroup: ""
roleRef:
  kind: Role
  name: ops-role
  apiGroup: ""
```

Testing

Testing

So far we created a user(`employee`) and add **certificates** to **kube-config**, gave `employee` read permission on **pods** and **deployments**.

Now, let's test what our `employee` can and cannot do.

- Run the following command,

```
kubectl --context=employee-context get pods
```

You should be able to retrieve the pods because `employee` is given `get` right on resource `pod` .

- Now let's run,

```
kubectl --context=employee-context run --image=nginx
```

You should see **access denied** error because `employee` doesn't have `run` right on any resource.

```
kubectl --context=employee-context get pods
```

```
kubectl --context=employee-context run --image=nginx
```