# Lab 6a - Manage authorization of your AKS cluster

## Introduction

In this Lab you configure the YADA application to ensure API developers can access the API namespace with read/write permissions and can also have read-only permission on the WEB namespace.

You will do that using 2 methods so you know how to do it but in real life you'll focus on only one:

- Azure RBAC
- Kubernetes RBAC
- Expected Lab duration: 30 minutes

## Challenge yourself (OPTIONAL)

If you want to challenge yourself, try to perform the following actions on your own:

1. Using Azure CLI, add `Azure Kubernetes Service RBAC Writer` Azure role to the API Student Group provided by the trainer (or just add one of your colleague)
2. Ask one colleague member of the group to access your cluster and verify it's possible to create a pod in the `api` namespace
3. Using a `RoleBinding` resource, add read only permission to the `web` namespace to the previous group
4. Ask one colleague member of the group to access your cluster and verify it's possible to list pods in the `web` namespace (and not create pod)

## Exercise 1: Use Azure RBAC to add permissions

In this exercise you will use Azure RBAC to add read/write permission to an Azure AD Group on the `api` namespace.

1. Open **Azure Cloud Shell**

2. Execute the following command to create a role assignment to the corresponding namespace (note the `scope` parameter in last command):

```
API_GROUP_NAME="${PREFIX} Students API"
API_GROUP_ID=$(az ad group show --group "${API_GROUP_NAME}" --query "id" --
output tsv)
CLUSTER_ID=$(az aks list -g ${RESOURCE_GROUP} --query "[].id" -o tsv)
echo "Your Cluster ID is ${CLUSTER_ID}"
echo "The ID of ${API_GROUP_NAME} is ${API_GROUP_ID}"
az role assignment create --assignee $API_GROUP_ID --role "Azure Kubernetes
Service RBAC Writer" --scope "${CLUSTER_ID}/namespaces/api"
```

3. OPTIONAL from here (if you want to validate): Execute the following command to find a colleague member of the group

```
az ad group member list -g ${API_GROUP_NAME} -o table --query "
[].displayName"
```

4. **Ask a colleague** from the previous user list to connect to your cluster using its credentials (ask him/her to replace X with your number) and perform some actions on your API resources

```
#
az aks get-credentials --resource-group rg-akstrainingX --name PREFIXaksX
kubectl get pod -n api # The pod list should appear
kubectl scale -n api deployment api --replicas 2
```

5. Now, ask your colleague to get pods of your web frontend, he/she should get a deny message

```
kubectl get pod -n web # An forbidden message should appear
```

## Exercise 2: Use Kubernetes RBAC to add permissions

In this exercise you will use Kubernetes RBAC to add read only permission to an Azure AD Group on the `web` namespace so API developers can have a look at what's going on on your deployment.

1. Open **Azure Cloud Shell**

2. Change your context back to your cluster in case you connected to your colleague's cluster:

```
kubectl config use-context "${PREFIX}aks${STUDENT_NB}"
```

3. Execute the following command to download the `RoleBinding` resource:

```
wget https://raw.githubusercontent.com/jbpaux/aks-infra-
training/main/yaml/lab6a/web-rolebinding.yml
```

4. Get the `id` of the group and replace it in the previously downloaded file. Have a look on the resource definition:

```
API_GROUP_NAME="${PREFIX} Students API"
API_GROUP_ID=$(az ad group show --group "${API_GROUP_NAME}" --query "id" --
output tsv)
echo "The ID of ${API_GROUP_NAME} is ${API_GROUP_ID}"
code web-rolebinding.yml #Replace APISTUDENTSGROUP in last line with the
previous ID
```

5. Apply the modified `RoleBinding` resource:

   ```
   kubectl apply -n web -f web-rolebinding.yml
   ```

6. OPTIONAL from here (if you want to validate): Execute the following command to find a colleague member of the group

   ```
   az ad group member list -g ${API_GROUP_NAME} -o table --query "
   [].displayName"
   ```

7. **Ask a colleague** to list the pods of your web frontend, he/she should get the list

   ```
   kubectl get pod -n web
   ```