Kubernetes AKS RBAC

Kubernetes RBAC stands for Role Based Access Control. It is used when multiple teams have the access of Kubernetes cluster, in those scenarios a user will get the restricted access by the Kubernetes Administrator using RBAC.

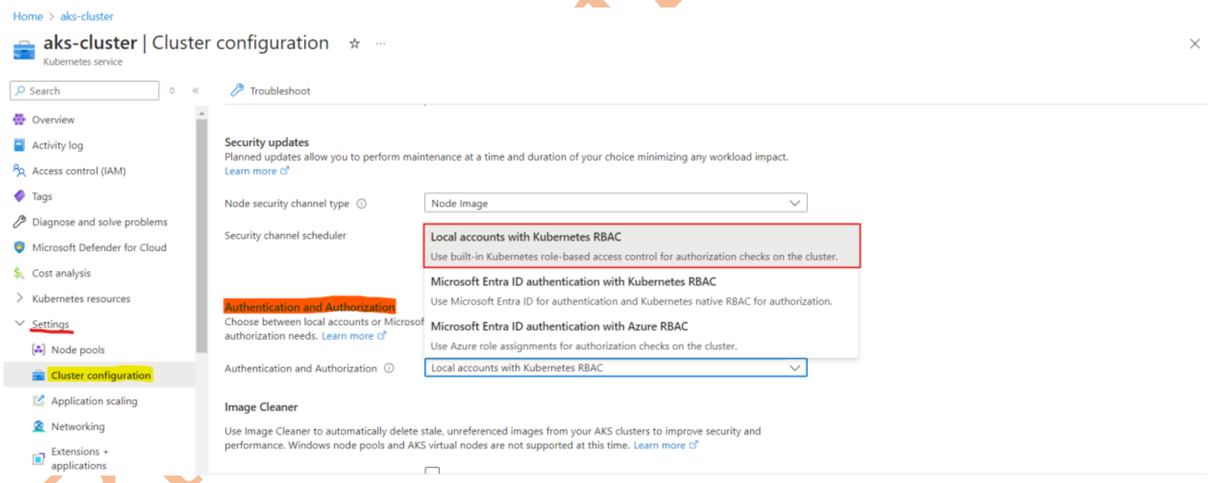
Here Kubernetes Cluster has been created in Azure using AKS (Azure Kubernetes Services), a managed Kubernetes Service. Azure AKS supports three ways for Authentication and Authorization of users as mentioned below.

(a) Local accounts with Kubernetes RBAC
(b) Microsoft Entra ID authentication with Kubernetes RBAC
(c) Microsoft Entra ID authentication with Azure RBAC

(a) Local accounts with Kubernetes RBAC

In this method of Authentication and Authorization in Azure we need to create the certificate for user and set the context for user modify the kubeconfig file and provide to user along with the created certificate.

(a) For demonstration I had created AKS cluster with the Authentication and Authorization option of **Local accounts with Kubernetes RBAC** and will change it if needed. Below screenshot shows the Authentication and Authorization options for Azure AKS.


Home > aks-cluster

aks-cluster | Cluster configuration

Kubernetes service

Security updates

Planned updates allow you to perform maintenance at a time and duration of your choice minimizing any workload impact. Learn more

Node security channel type: Node Image

Security channel scheduler

Local accounts with Kubernetes RBAC
Use built-in Kubernetes role-based access control for authorization checks on the cluster.

Microsoft Entra ID authentication with Kubernetes RBAC
Use Microsoft Entra ID for authentication and Kubernetes native RBAC for authorization.

Microsoft Entra ID authentication with Azure RBAC
Use Azure role assignments for authorization checks on the cluster.

Authentication and Authorization

Choose between local accounts or Microsof authorization needs. Learn more

Authentication and Authorization: Local accounts with Kubernetes RBAC

Image Cleaner

Use Image Cleaner to automatically delete stale, unreferenced images from your AKS clusters to improve security and performance. Windows node pools and AKS virtual nodes are not supported at this time. Learn more

1. I will create a user adam and will provide him only reader access to pods, services, and deployments in the namespace maxo.

Create the namespace in Kubernetes using the command **kubectl create ns maxo** as shown in the screenshot attached below.

```
[root@Terraform-Server ~]# kubectl create ns maxo
namespace/maxo created
```

Create Role with the name of reader and RoleBinding with the name of reader-binding as shown in the screenshot attached below.

vim role-rolebinding.yaml

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: reader
  namespace: maxo
rules:
- apiGroups: [""]
  resources: ["pods", "services"]
  verbs: ["get", "watch", "list"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "watch", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: reader-binding
  namespace: maxo
subjects:
- kind: User
  name: adam
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: reader
  apiGroup: rbac.authorization.k8s.io
```

```
[root@Terraform-Server ~]# kubectl apply -f role-rolebinding.yaml
role.rbac.authorization.k8s.io/reader created
rolebinding.rbac.authorization.k8s.io/reader-binding created
[root@Terraform-Server ~]# cat role-rolebinding.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: reader
  namespace: maxo
rules:
- apiGroups: [""]
  resources: ["pods", "services"]
  verbs: ["get", "watch", "list"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "watch", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: reader-binding
  namespace: maxo
subjects:
- kind: User
  name: adam
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: reader
  apiGroup: rbac.authorization.k8s.io
```

Create certificates so that user can access the cluster

mkdir cred && cd cred

openssl genrsa -out adam.key 2048

```
[root@Terraform-Server ~]# mkdir cred && cd cred
[root@Terraform-Server cred]# openssl genrsa -out adam.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
........................................................+++++
...............................+++++
e is 65537 (0x010001)
```

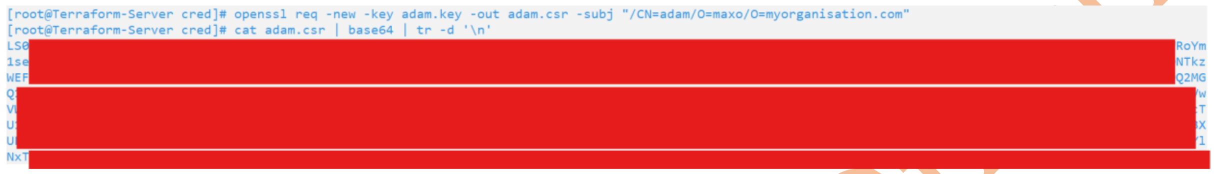
CN: This will be set as username

O: Org name. This is used as a group by Kubernetes while authenticating/authorizing users. You could add as many as you need

openssl req -new -key adam.key -out adam.csr -subj "/CN=adam/O=maxo/O=myorganisation.com"

Convert the CSR to Base64

cat adam.csr | base64 | tr -d '\n'

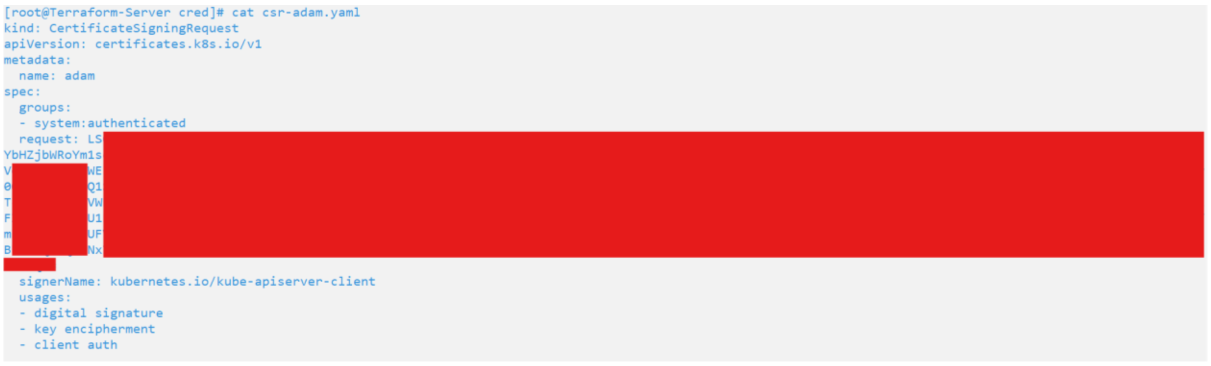


```
[root@Terraform-Server cred]# openssl req -new -key adam.key -out adam.csr -subj "/CN=adam/O=maxo/O=myorganisation.com"
[root@Terraform-Server cred]# cat adam.csr | base64 | tr -d '\n'
```
[redacted]

The CSR file after encoding to base64 we need to send to Kubernetes and ask Kubernetes to register it. Finally, we need to approve this request.

cat csr-adam.yaml

```
kind: CertificateSigningRequest
apiVersion: certificates.k8s.io/v1
metadata:
  name: adam
spec:
  groups:
  - system:authenticated
  request: <paste base64 string from above step>
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - digital signature
  - key encipherment
  - client auth
```


```
[root@Terraform-Server cred]# cat csr-adam.yaml
kind: CertificateSigningRequest
apiVersion: certificates.k8s.io/v1
metadata:
  name: adam
spec:
  groups:
  - system:authenticated
  request: LS
YbHZjbWRoYm1s[redacted]
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - digital signature
  - key encipherment
  - client auth
```

Approve certificate signing request

kubectl certificate approve adam

kubectl get csr

create .crt extension certificate file from the token

kubectl get csr adam -o jsonpath='{.status.certificate}' | base64 --decode > adam.crt

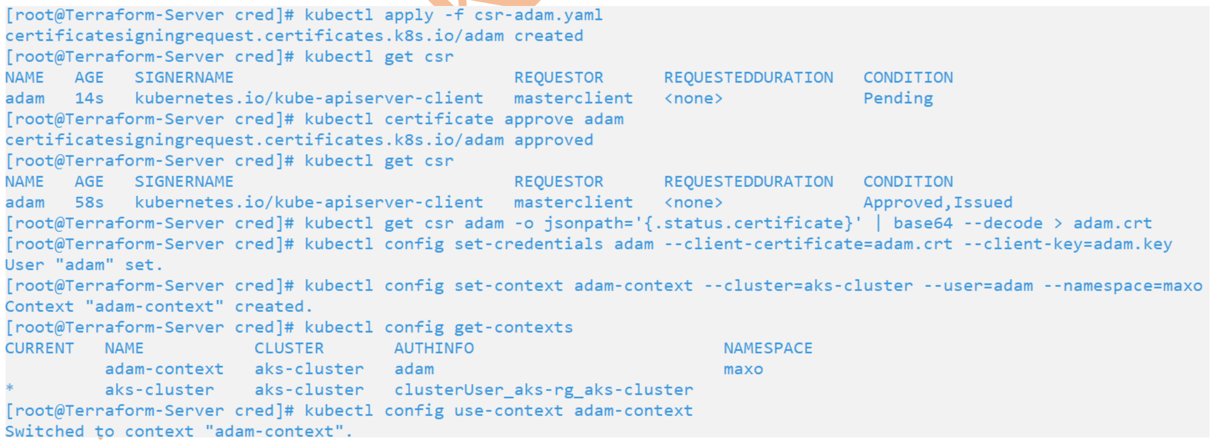
configure these details in kubeconfig file

kubectl config set-credentials adam --client-certificate=adam.crt --client-key=adam.key

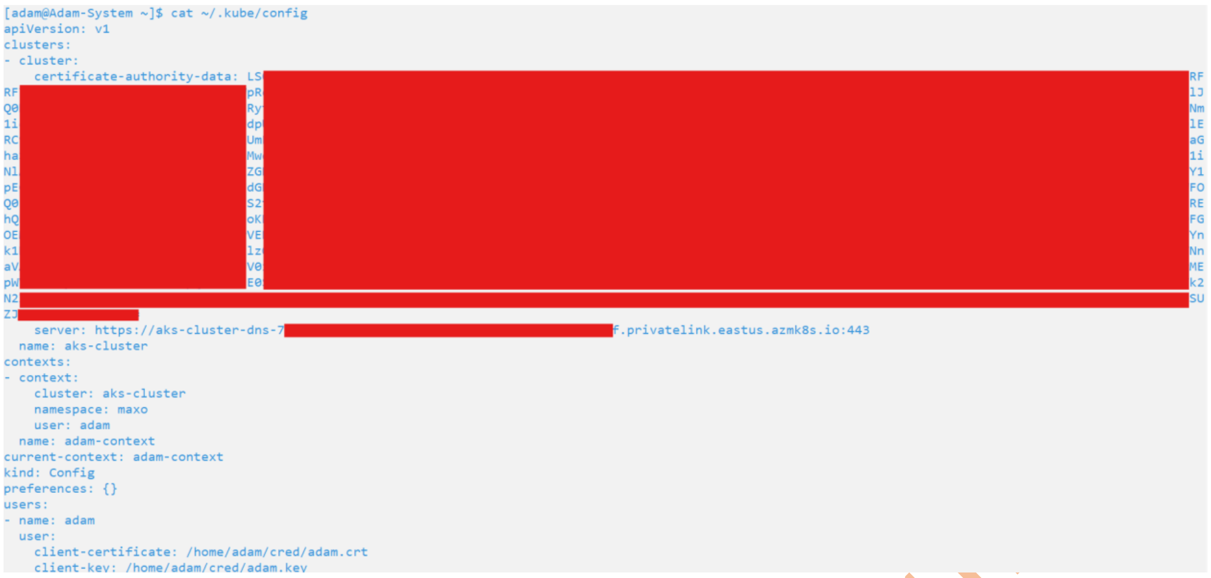
kubectl config set-context adam-context --cluster=aks-cluster --user=adam --namespace=maxo

kubectl config get-contexts

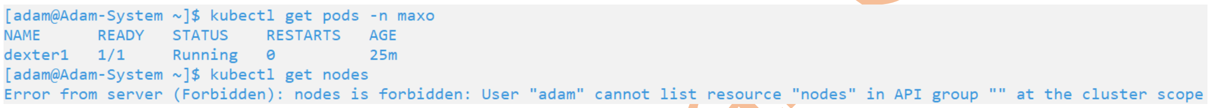
kubectl config use-context adam-context


```
[root@Terraform-Server cred]# kubectl apply -f csr-adam.yaml
certificatesigningrequest.certificates.k8s.io/adam created
[root@Terraform-Server cred]# kubectl get csr
NAME   AGE   SIGNERNAME                            REQUESTOR      REQUESTEDDURATION   CONDITION
adam   14s   kubernetes.io/kube-apiserver-client   masterclient   <none>              Pending
[root@Terraform-Server cred]# kubectl certificate approve adam
certificatesigningrequest.certificates.k8s.io/adam approved
[root@Terraform-Server cred]# kubectl get csr
NAME   AGE   SIGNERNAME                            REQUESTOR      REQUESTEDDURATION   CONDITION
adam   58s   kubernetes.io/kube-apiserver-client   masterclient   <none>              Approved,Issued
[root@Terraform-Server cred]# kubectl get csr adam -o jsonpath='{.status.certificate}' | base64 --decode > adam.crt
[root@Terraform-Server cred]# kubectl config set-credentials adam --client-certificate=adam.crt --client-key=adam.key
User "adam" set.
[root@Terraform-Server cred]# kubectl config set-context adam-context --cluster=aks-cluster --user=adam --namespace=maxo
Context "adam-context" created.
[root@Terraform-Server cred]# kubectl config get-contexts
CURRENT   NAME           CLUSTER       AUTHINFO                          NAMESPACE
          adam-context   aks-cluster   adam                              maxo
*         aks-cluster    aks-cluster   clusterUser_aks-rg_aks-cluster
[root@Terraform-Server cred]# kubectl config use-context adam-context
Switched to context "adam-context".
```

Share this kubeconfig file with user adam so that user can access the namespace within the cluster. Below screenshot show the kubeconfig file which I had shared with the user.


```
[adam@Adam-System ~]$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS[redacted]
[redacted]
    server: https://aks-cluster-dns-7[redacted]f.privatelink.eastus.azmk8s.io:443
  name: aks-cluster
contexts:
- context:
    cluster: aks-cluster
    namespace: maxo
    user: adam
  name: adam-context
current-context: adam-context
kind: Config
preferences: {}
users:
- name: adam
  user:
    client-certificate: /home/adam/cred/adam.crt
    client-key: /home/adam/cred/adam.key
```

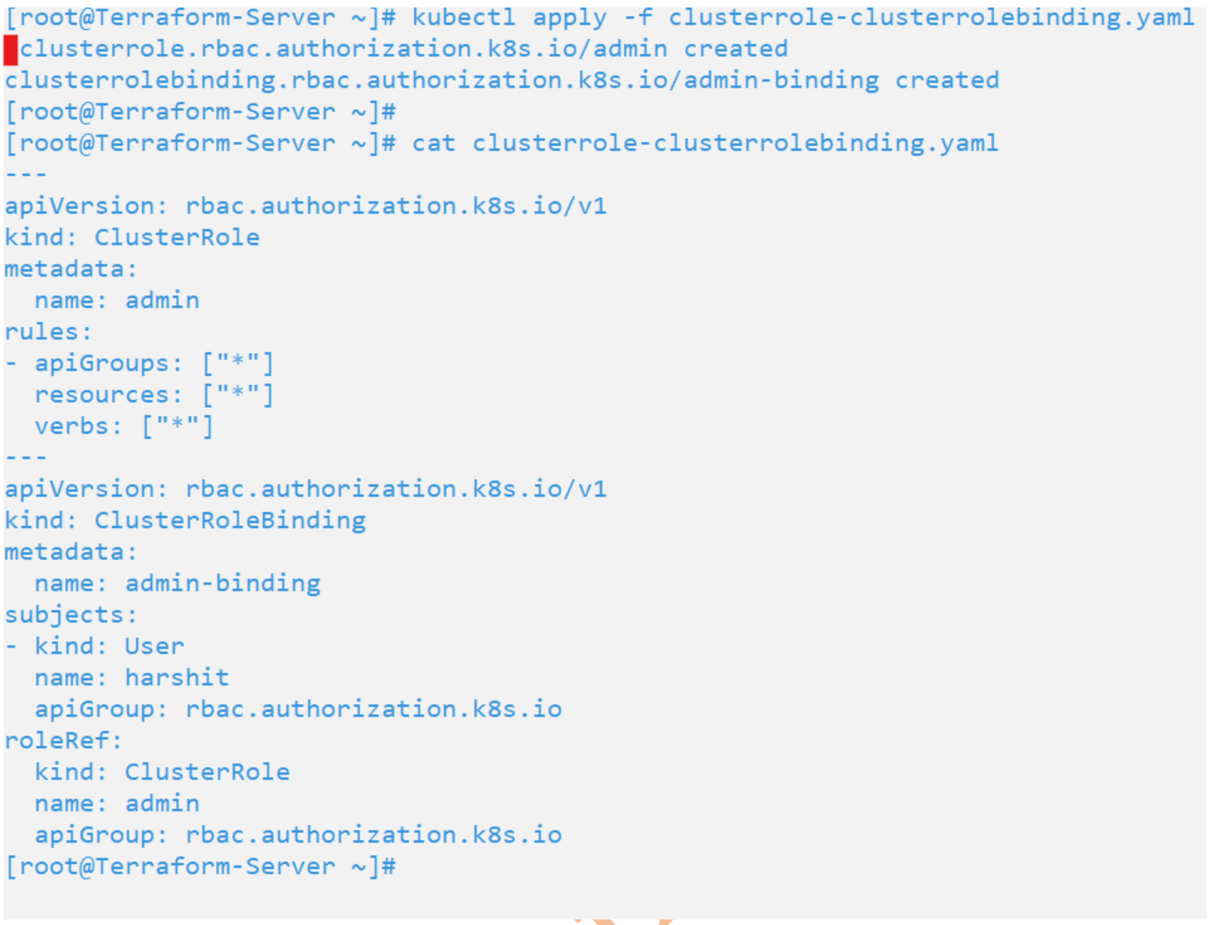
As shown in the screenshot attached below the user adam cannot do much activity. He can list the pods but unable to list the nodes.


```
[adam@Adam-System ~]$ kubectl get pods -n maxo
NAME      READY    STATUS     RESTARTS    AGE
dexter1   1/1      Running    0           25m
[adam@Adam-System ~]$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "adam" cannot list resource "nodes" in API group "" at the cluster scope
```

2. I will create a user harshit and will provide him administrator access to the entire Kubernetes cluster. To achieve this, I had created a ClusterRole with the name of admin and ClusterRoleBinding with the name of admin-binding.

vim clusterrole-clusterrolebinding.yaml

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: admin
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-binding
subjects:
- kind: User
  name: harshit
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: admin
  apiGroup: rbac.authorization.k8s.io
```



```
[root@Terraform-Server ~]# kubectl apply -f clusterrole-clusterrolebinding.yaml
clusterrole.rbac.authorization.k8s.io/admin created
clusterrolebinding.rbac.authorization.k8s.io/admin-binding created
[root@Terraform-Server ~]#
[root@Terraform-Server ~]# cat clusterrole-clusterrolebinding.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: admin
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-binding
subjects:
- kind: User
  name: harshit
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: admin
  apiGroup: rbac.authorization.k8s.io
[root@Terraform-Server ~]#
```

Create certificates so that user can access the cluster

mkdir cred && cd cred

openssl genrsa -out harshit.key 2048

CN: This will be set as username

O: Org name. This is used as a group by Kubernetes while authenticating/authorizing users. You could add as many as you need

openssl req -new -key harshit.key -out harshit.csr -subj "/CN=harshit/O=administrator/O=myorganisation.com"

Convert the CSR to Base64

cat harshit.csr | base64 | tr -d '\n'

The CSR file after encoding to base64 we need to send to Kubernetes and ask Kubernetes to register it. Finally, we need to approve this request.

```
cat csr-harshit.yaml


kind: CertificateSigningRequest
apiVersion: certificates.k8s.io/v1
metadata:
  name: harshit
spec:
  groups:
  - system:authenticated
  request: <paste base64 string from above step>
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - digital signature
  - key encipherment
  - client auth
```

Approve certificate signing request

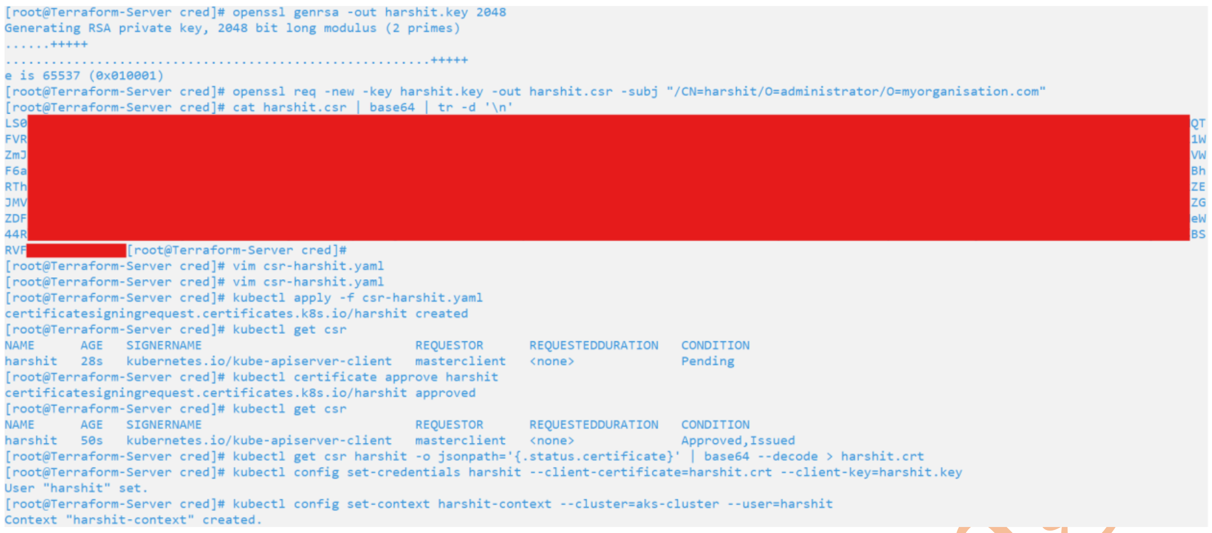
```
kubectl certificate approve harshit
kubectl get csr
```

create .crt extension certificate file from the token

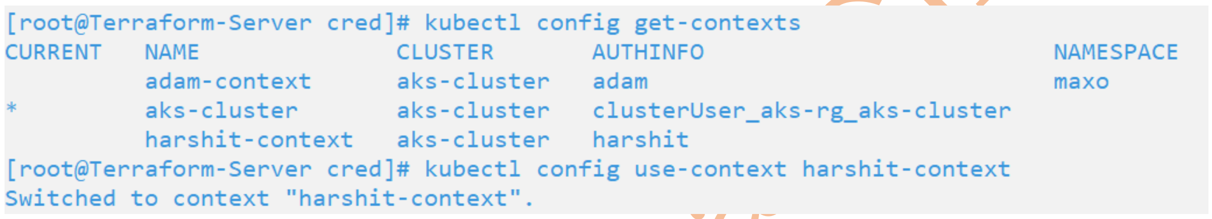
```
kubectl get csr harshit -o jsonpath='{.status.certificate}' | base64 --decode > harshit.crt
```

configure these details in kubeconfig file

```
kubectl config set-credentials harshit --client-certificate=harshit.crt --client-key=harshit.key
kubectl config set-context harshit-context --cluster=aks-cluster --user=harshit
kubectl config get-contexts
kubectl config use-context harshit-context
```

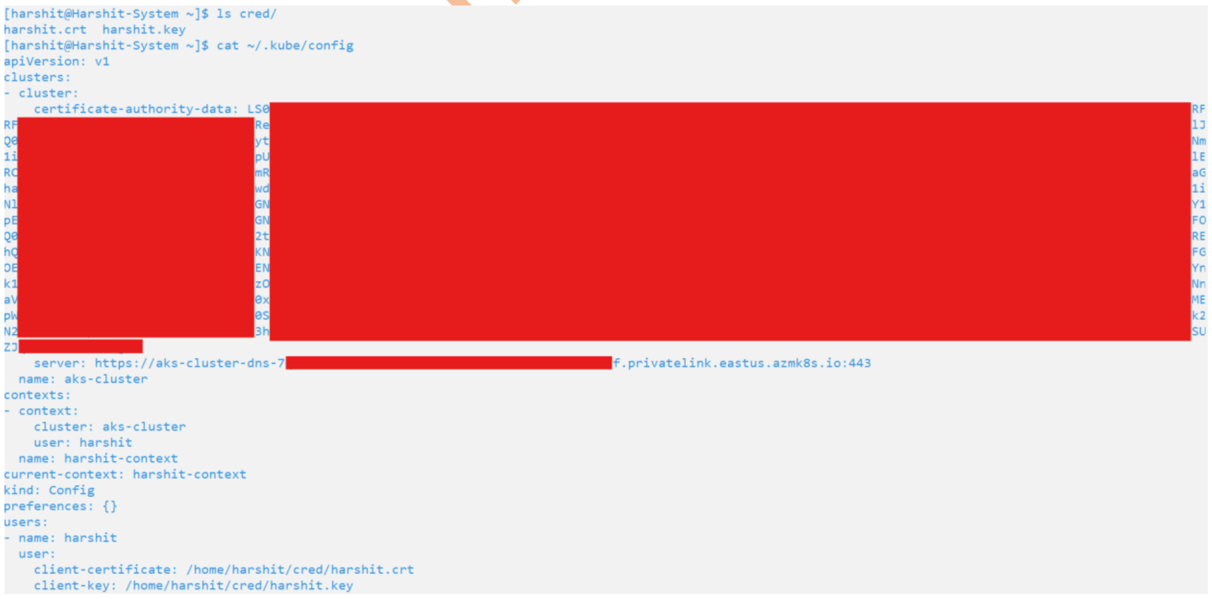


```
[root@Terraform-Server cred]# openssl genrsa -out harshit.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
......+++++
.......................................................+++++
e is 65537 (0x010001)
[root@Terraform-Server cred]# openssl req -new -key harshit.key -out harshit.csr -subj "/CN=harshit/O=administrator/O=myorganisation.com"
[root@Terraform-Server cred]# cat harshit.csr | base64 | tr -d '\n'
LS0[redacted]QT
FVR   1W
ZmJ   VW
F6a   Bh
RTh   ZE
JMV   ZG
ZDF   eW
44R   BS
RVF[redacted][root@Terraform-Server cred]#
[root@Terraform-Server cred]# vim csr-harshit.yaml
[root@Terraform-Server cred]# vim csr-harshit.yaml
[root@Terraform-Server cred]# kubectl apply -f csr-harshit.yaml
certificatesigningrequest.certificates.k8s.io/harshit created
[root@Terraform-Server cred]# kubectl get csr
NAME      AGE   SIGNERNAME                            REQUESTOR      REQUESTEDDURATION   CONDITION
harshit   28s   kubernetes.io/kube-apiserver-client   masterclient   <none>              Pending
[root@Terraform-Server cred]# kubectl certificate approve harshit
certificatesigningrequest.certificates.k8s.io/harshit approved
[root@Terraform-Server cred]# kubectl get csr
NAME      AGE   SIGNERNAME                            REQUESTOR      REQUESTEDDURATION   CONDITION
harshit   50s   kubernetes.io/kube-apiserver-client   masterclient   <none>              Approved,Issued
[root@Terraform-Server cred]# kubectl get csr harshit -o jsonpath='{.status.certificate}' | base64 --decode > harshit.crt
[root@Terraform-Server cred]# kubectl config set-credentials harshit --client-certificate=harshit.crt --client-key=harshit.key
User "harshit" set.
[root@Terraform-Server cred]# kubectl config set-context harshit-context --cluster=aks-cluster --user=harshit
Context "harshit-context" created.
```



```
[root@Terraform-Server cred]# kubectl config get-contexts
CURRENT   NAME              CLUSTER       AUTHINFO                          NAMESPACE
          adam-context      aks-cluster   adam                              maxo
*         aks-cluster       aks-cluster   clusterUser_aks-rg_aks-cluster
          harshit-context   aks-cluster   harshit
[root@Terraform-Server cred]# kubectl config use-context harshit-context
Switched to context "harshit-context".
```

Share this kubeconfig file with user harshit so that user can access the Kubernetes cluster. The kubeconfig file which I had shared with the user harshit is shown in the screenshot attached below. I shared the kubeconfig file along with harshit.key and harshit.crt with the user so that user can be able to access the Kubernetes cluster with administrator privilege.



```
[harshit@Harshit-System ~]$ ls cred/
harshit.crt  harshit.key
[harshit@Harshit-System ~]$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0[redacted]RF
RF[redacted]Re   1J
Q0   yt   Nm
1i   pU   1E
RC   mR   aG
ha   wd   1i
N1   GN   Y1
pE   GN   FO
Q0   2t   RE
hQ   KN   FG
OE   EN   Yn
k1   zO   Nn
aV   0x   ME
pW   0S   k2
N2   3h   SU
ZJ[redacted]
    server: https://aks-cluster-dns-7[redacted]f.privatelink.eastus.azmk8s.io:443
  name: aks-cluster
contexts:
- context:
    cluster: aks-cluster
    user: harshit
  name: harshit-context
current-context: harshit-context
kind: Config
preferences: {}
users:
- name: harshit
  user:
    client-certificate: /home/harshit/cred/harshit.crt
    client-key: /home/harshit/cred/harshit.key
```

```
[harshit@Harshit-System ~]$ kubectl get nodes
NAME                                STATUS   ROLES   AGE     VERSION
aks-agentpool-2[redacted]22-vmss000000   Ready    agent   4h13m   v1.28.5
aks-userpool-2[redacted]13-vmss000000    Ready    agent   4h5m    v1.28.5
```

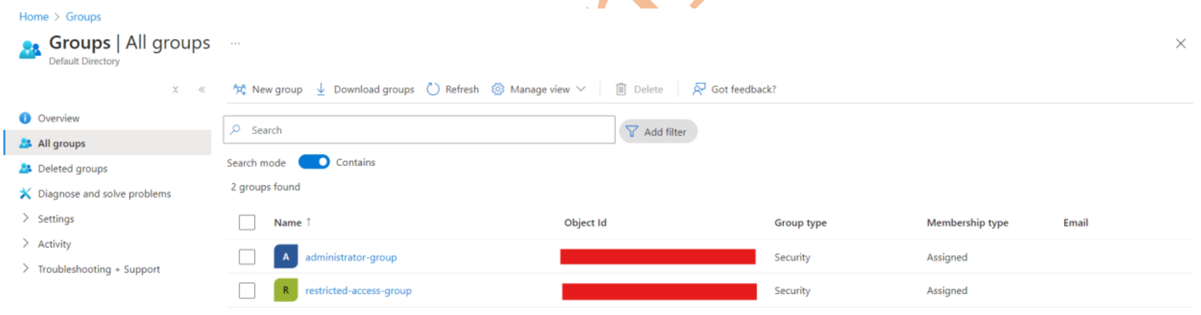
(b) Microsoft Entra ID authentication with Kubernetes RBAC

With this option of Authentication and Authorization in Azure AKS the user or user group will be created in Azure Entra ID (for authentication) and authorize this user/user group with Kubernetes Role/RoleBinding or ClusterRole/ClusterRoleBinding.

Here I had created two groups and two users in Azure Entra ID named as administrator-group, restricted-access-group and kunal and administrator (user kunal is member of restricted-access-group and user administrator is member of administrator-group) as shown in the screenshot attached below.

I will provide reader access (can only read pod, service, and deployment) to the user kunal who is the member of group restricted-access-group in the namespace zokomo. Administrator access had been provided to the user administrator (can access entire Kubernetes cluster as an administrator) who is a member of group administrator-group.

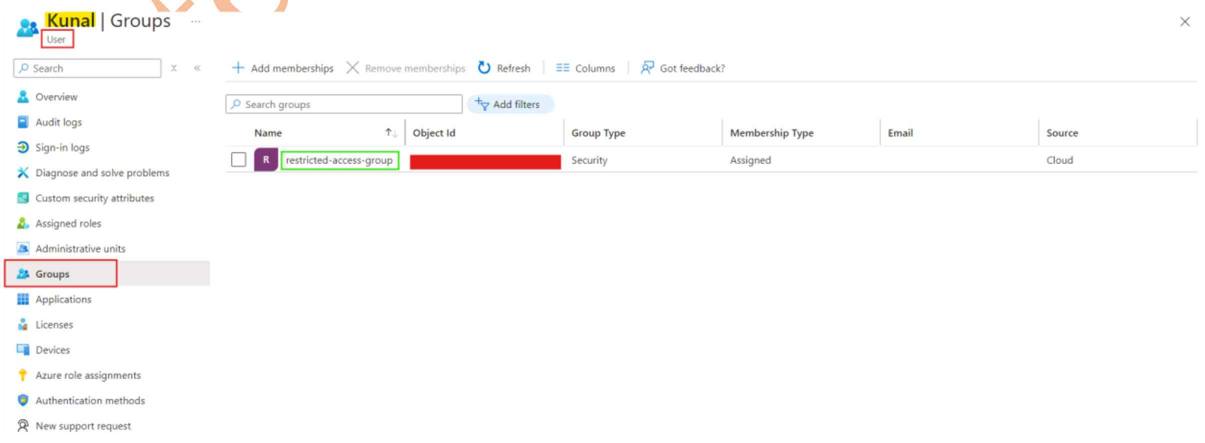
Below screenshot shows the two Azure Entra ID groups administrator-group and restricted-access-group.



2 groups found

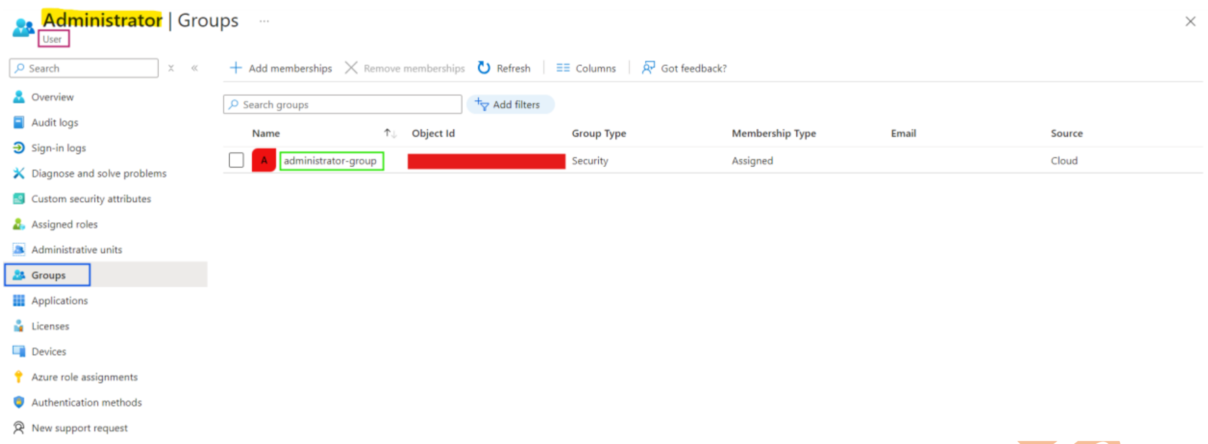
Name ↑	Object Id	Group type	Membership type	Email
administrator-group	[redacted]	Security	Assigned	
restricted-access-group	[redacted]	Security	Assigned	

User kunal is the member of restricted-access-group.



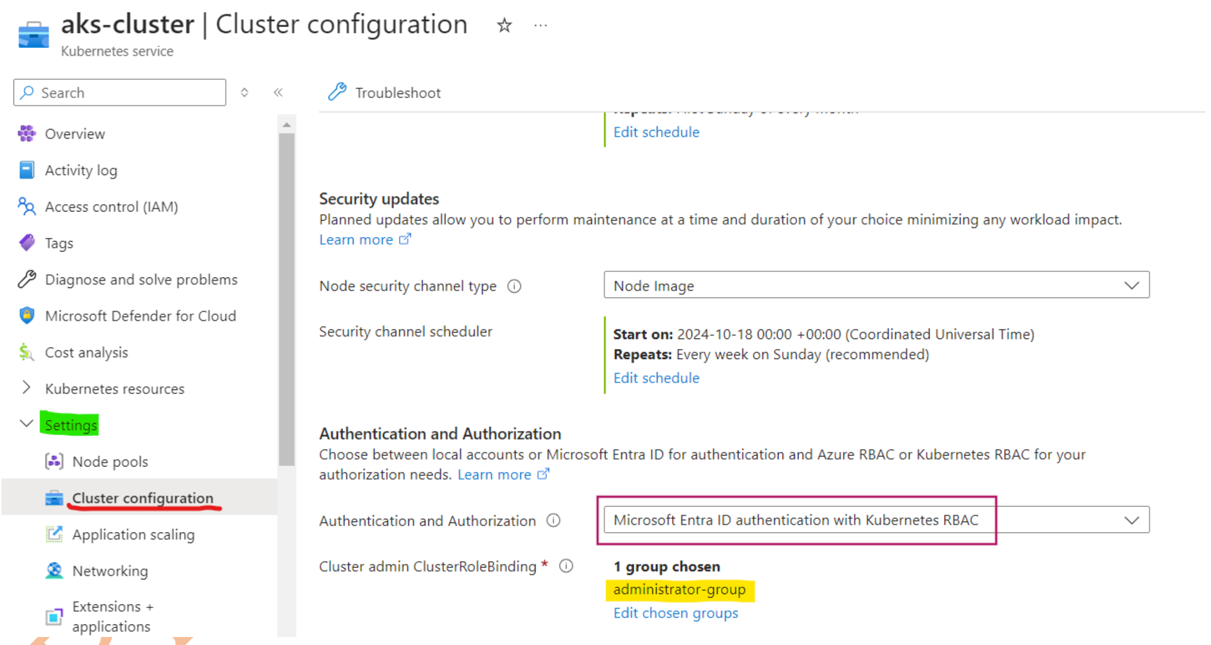
Name	Object Id	Group Type	Membership Type	Email	Source
restricted-access-group	[redacted]	Security	Assigned		Cloud

User administrator is a member of administrator-group.


Administrator | Groups
User

Name: administrator-group | Object Id: [redacted] | Group Type: Security | Membership Type: Assigned | Email: | Source: Cloud

Authentication and Authorization strategy used is **Microsoft Entra ID authentication with Kubernetes RBAC** and for **Cluster admin ClusterRoleBinding** option an Azure AD (Azure Entra ID) Group should be attached. The **Users within this Azure AD group will get admin access within the cluster**.


aks-cluster | Cluster configuration
Kubernetes service

Edit schedule

Security updates
Planned updates allow you to perform maintenance at a time and duration of your choice minimizing any workload impact.
Learn more

Node security channel type: Node Image

Security channel scheduler:
Start on: 2024-10-18 00:00 +00:00 (Coordinated Universal Time)
Repeats: Every week on Sunday (recommended)
Edit schedule

Authentication and Authorization
Choose between local accounts or Microsoft Entra ID for authentication and Azure RBAC or Kubernetes RBAC for your authorization needs. Learn more

Authentication and Authorization: Microsoft Entra ID authentication with Kubernetes RBAC

Cluster admin ClusterRoleBinding *:
1 group chosen
administrator-group
Edit chosen groups

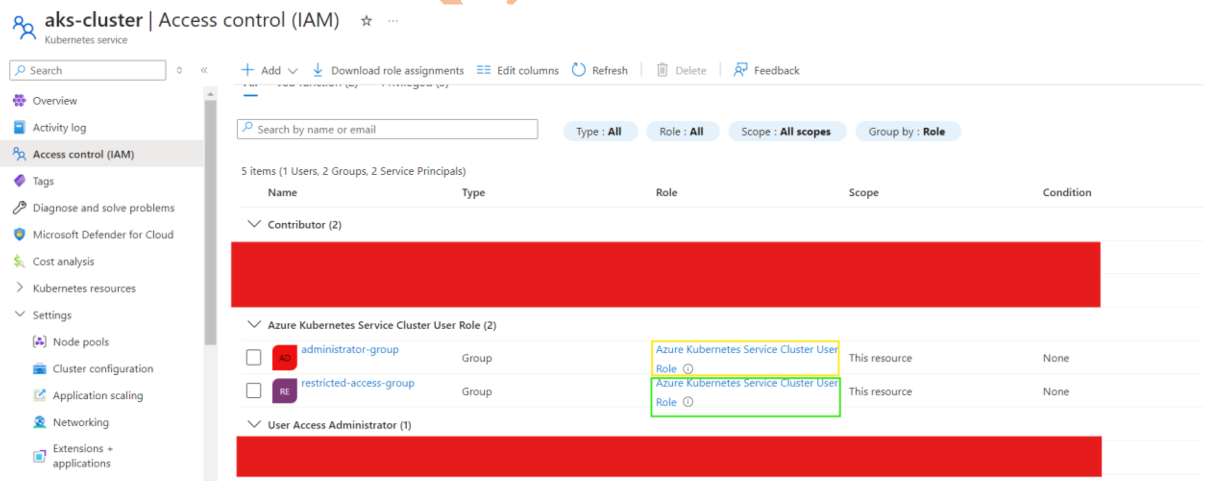
The Role and RoleBinding used to provide the restricted access to the user kunal through the Object ID of the Azure AD Group whose member Kunal was had been shown in the screenshot attached below.

vim role-rolebinding.yaml

```yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: reader
  namespace: zokomo
rules:
- apiGroups: [""]
  resources: ["pods", "services"]
  verbs: ["get", "watch", "list"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "watch", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: reader-binding
  namespace: zokomo
subjects:
- kind: Group
  name: aXXXXXX8-XXXX-XXXX-XXXX-0XXXXXXXXXX5  # Use Object ID of the Azure AD User Group
  namespace: zokomo
roleRef:
  kind: Role
  name: reader
  apiGroup: rbac.authorization.k8s.io
```

```
[root@Terraform-Server ~]# kubectl apply -f role-rolebinding.yaml
role.rbac.authorization.k8s.io/reader created
rolebinding.rbac.authorization.k8s.io/reader-binding created
[root@Terraform-Server ~]# cat role-rolebinding.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: reader
  namespace: zokomo
rules:
- apiGroups: [""]
  resources: ["pods", "services"]
  verbs: ["get", "watch", "list"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "watch", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: reader-binding
  namespace: zokomo
subjects:
- kind: Group
  name: a[redacted]5   # Use Object ID of the Azure AD User Group
  namespace: zokomo
roleRef:
  kind: Role
  name: reader
  apiGroup: rbac.authorization.k8s.io
```

Provide at least **Azure Kubernetes Service Cluster User Role** to the two Azure AD (Azure Entra ID) Groups restricted-access-group and administrator-group for the Azure AKS Cluster as shown in the screenshot attached below. As this Role is needed to generate the kubeconfig file.

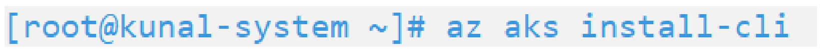


5 items (1 Users, 2 Groups, 2 Service Principals)

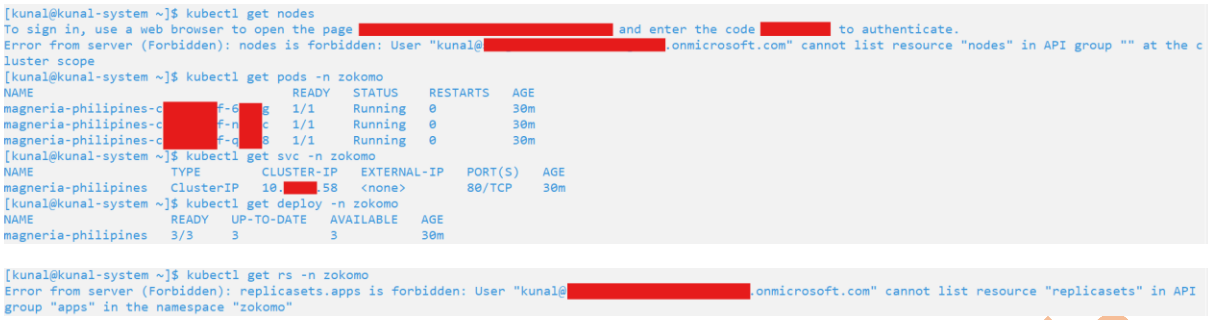
Name	Type	Role	Scope	Condition
Contributor (2)				
[redacted]				
Azure Kubernetes Service Cluster User Role (2)				
administrator-group	Group	Azure Kubernetes Service Cluster User Role	This resource	None
restricted-access-group	Group	Azure Kubernetes Service Cluster User Role	This resource	None
User Access Administrator (1)				
[redacted]				

Using az login, login with user kunal and try to list the nodes and list the pods, service, and deployment in the namespace zokomo and see the result as shown in the screenshot attached below.

```
[kunal@kunal-system ~]$ az aks get-credentials --resource-group aks-rg --name aks-cluster
Merged "aks-cluster" as current context in /home/kunal/.kube/config
```



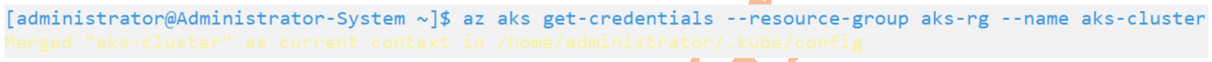
```
[root@kunal-system ~]# az aks install-cli
```



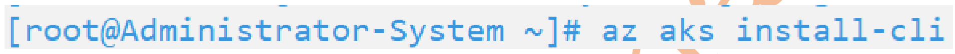
```
[kunal@kunal-system ~]$ kubectl get nodes
To sign in, use a web browser to open the page [redacted] and enter the code [redacted] to authenticate.
Error from server (Forbidden): nodes is forbidden: User "kunal@[redacted].onmicrosoft.com" cannot list resource "nodes" in API group "" at the c
luster scope
[kunal@kunal-system ~]$ kubectl get pods -n zokomo
NAME                                  READY   STATUS    RESTARTS   AGE
magneria-philipines-c[redacted]f-6[redacted]g   1/1     Running   0          30m
magneria-philipines-c[redacted]f-n[redacted]c   1/1     Running   0          30m
magneria-philipines-c[redacted]f-q[redacted]8   1/1     Running   0          30m
[kunal@kunal-system ~]$ kubectl get svc -n zokomo
NAME                  TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
magneria-philipines   ClusterIP   10.[redacted].58   <none>        80/TCP    30m
[kunal@kunal-system ~]$ kubectl get deploy -n zokomo
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
magneria-philipines   3/3     3            3           30m

[kunal@kunal-system ~]$ kubectl get rs -n zokomo
Error from server (Forbidden): replicasets.apps is forbidden: User "kunal@[redacted].onmicrosoft.com" cannot list resource "replicasets" in API
group "apps" in the namespace "zokomo"
```

The user kunal does not have overall access but will have limited access in zokomo namespace.

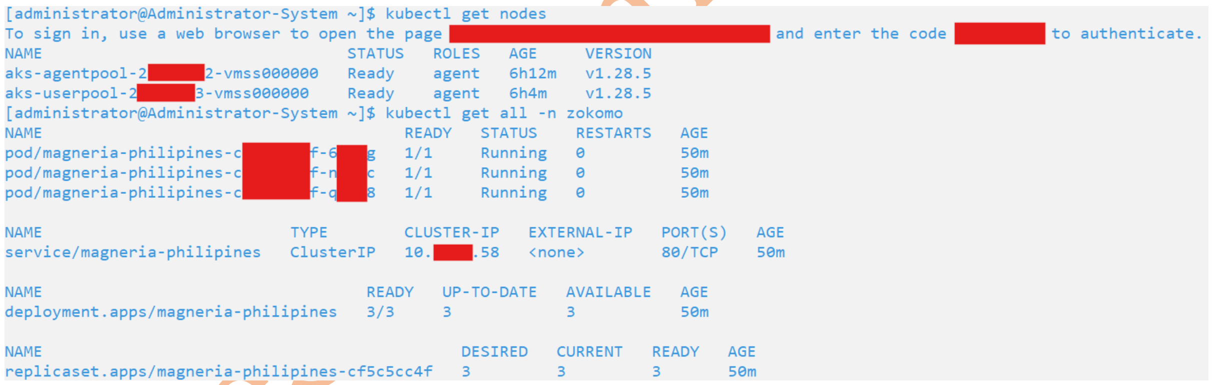
Now login with User administrator with the help of az login and then try to access all the resources within the cluster and you will find that user administrator has all the access within the cluster as the user is a member of Azure AD Group (Azure Entra ID Group) administrator-group and this group is attached to **Cluster admin ClusterRoleBinding** option as discussed above.



```
[administrator@Administrator-System ~]$ az aks get-credentials --resource-group aks-rg --name aks-cluster
Merged "aks-cluster" as current context in /home/administrator/.kube/config
```



```
[root@Administrator-System ~]# az aks install-cli
```



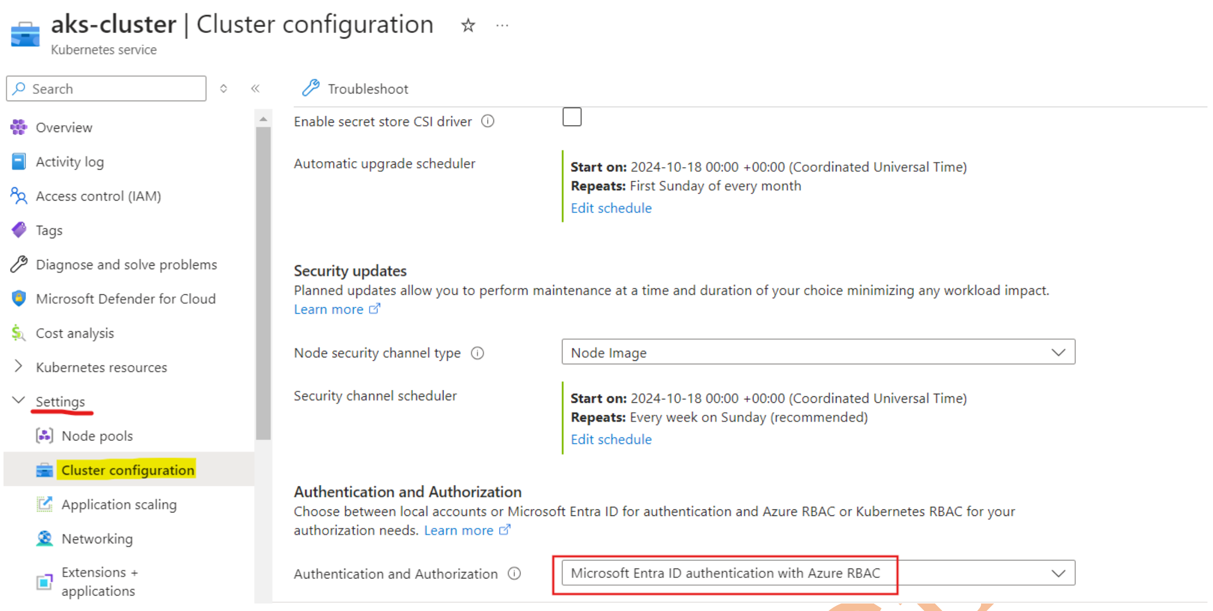
```
[administrator@Administrator-System ~]$ kubectl get nodes
To sign in, use a web browser to open the page [redacted] and enter the code [redacted] to authenticate.
NAME                                STATUS   ROLES   AGE     VERSION
aks-agentpool-2[redacted]2-vmss000000   Ready    agent   6h12m   v1.28.5
aks-userpool-2[redacted]3-vmss000000    Ready    agent   6h4m    v1.28.5
[administrator@Administrator-System ~]$ kubectl get all -n zokomo
NAME                                      READY   STATUS    RESTARTS   AGE
pod/magneria-philipines-c[redacted]f-6[redacted]g   1/1     Running   0          50m
pod/magneria-philipines-c[redacted]f-n[redacted]c   1/1     Running   0          50m
pod/magneria-philipines-c[redacted]f-q[redacted]8   1/1     Running   0          50m

NAME                          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/magneria-philipines   ClusterIP   10.[redacted].58   <none>        80/TCP    50m

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/magneria-philipines   3/3     3            3           50m

NAME                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/magneria-philipines-cf5c5cc4f   3         3         3       50m
```

(c) Microsoft Entra ID authentication with Azure RBAC

With this Authorization and Authentication strategy in Azure AKS the user and user group will be created in Azure Entra ID for Authentication and Authorization will be provided to these users through the Azure Role.

For this demonstration I had created two users and two groups named as deendayal, admin who are the member of groups restrictedaccess-group and administrator-group respectively as shown in the screenshot attached below.

I had created an Azure Role using which I will give restricted access (read access to pod, service, and deployment) to the user deendayal for the scope of namespace dexter through the Azure Entra ID Group Object ID. It means user deendayal is a member of Azure Entra ID Group restrictedaccess-group who had read access to pod, service, and deployment in the namespace dexter.


aks-cluster | Cluster configuration
Kubernetes service

Enable secret store CSI driver: ☐

Automatic upgrade scheduler
Start on: 2024-10-18 00:00 +00:00 (Coordinated Universal Time)
Repeats: First Sunday of every month
Edit schedule

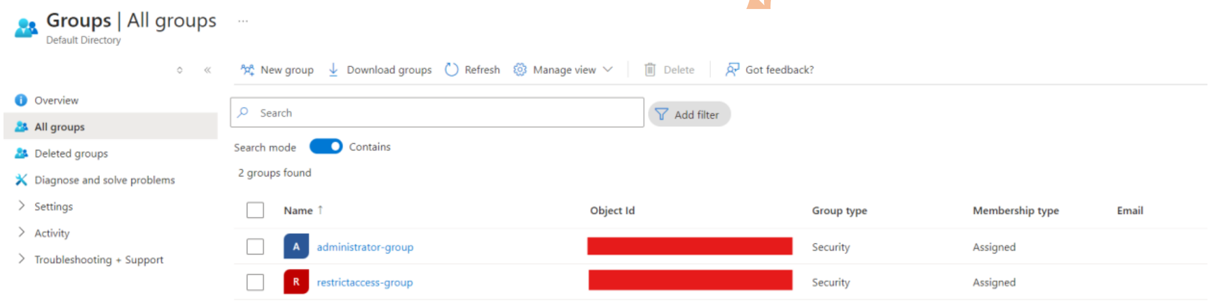
Security updates
Planned updates allow you to perform maintenance at a time and duration of your choice minimizing any workload impact. Learn more

Node security channel type: Node Image

Security channel scheduler
Start on: 2024-10-18 00:00 +00:00 (Coordinated Universal Time)
Repeats: Every week on Sunday (recommended)
Edit schedule

Authentication and Authorization
Choose between local accounts or Microsoft Entra ID for authentication and Azure RBAC or Kubernetes RBAC for your authorization needs. Learn more

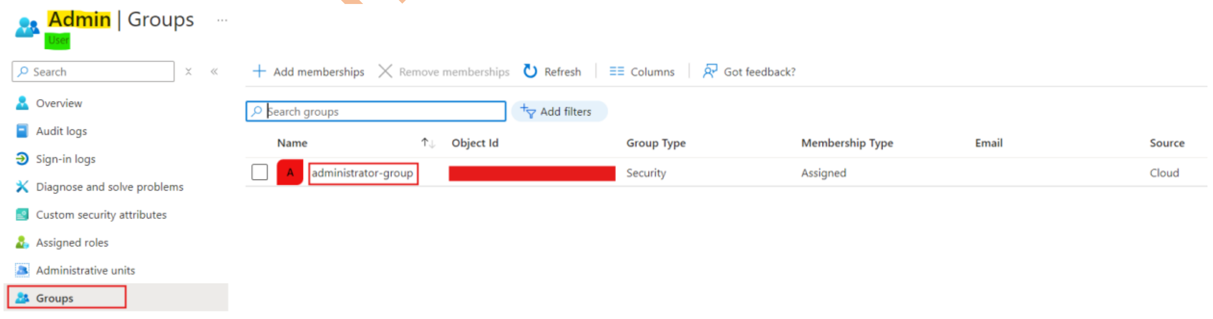
Authentication and Authorization: Microsoft Entra ID authentication with Azure RBAC


Groups | All groups
Default Directory

Search mode: Contains
2 groups found

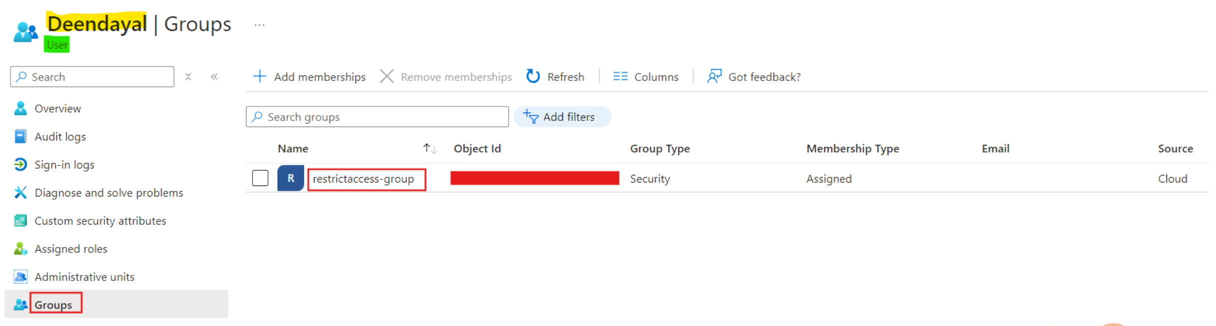
Name	Object Id	Group type	Membership type	Email
administrator-group	[redacted]	Security	Assigned	
restrictaccess-group	[redacted]	Security	Assigned	

Azure Entra ID User admin is the member of Azure Entra ID Group administrator-group.


Admin | Groups
User

Name	Object Id	Group Type	Membership Type	Email	Source
administrator-group	[redacted]	Security	Assigned		Cloud

Azure Entra ID User deendayal is the member of Azure Entra ID Group restrictedaccess-group.


Deendayal | Groups ...
User

Name	Object Id	Group Type	Membership Type	Email	Source
restrictaccess-group	[redacted]	Security	Assigned		Cloud

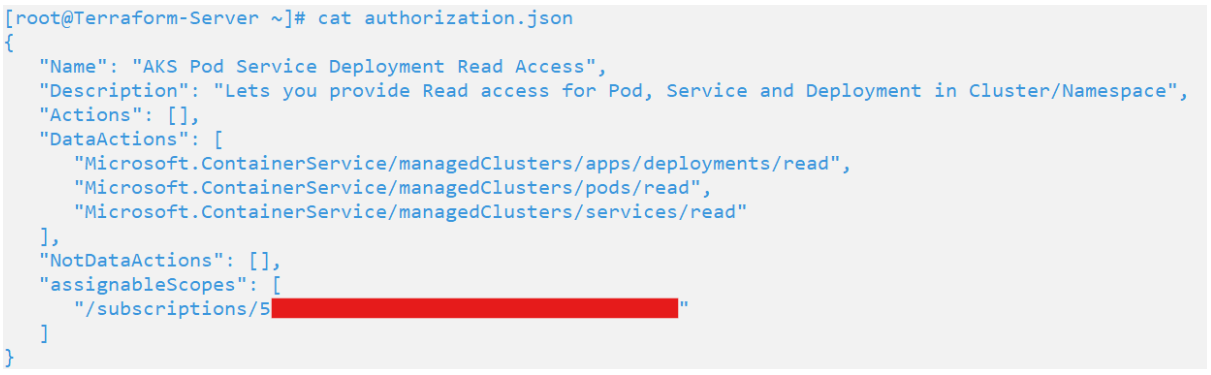
(1) Create a Custom Role to get Read Access for Pods, Services and Deployments (When you create a Custom Role in Azure the Assignable scopes can be ManagementGroup, Subscription or ResourceGroup).

(2) Assign this Azure Role to Azure AKS (You can assign an Azure Role to User, Group, Service Principle or Managed Identity for the scope of Azure Resource).

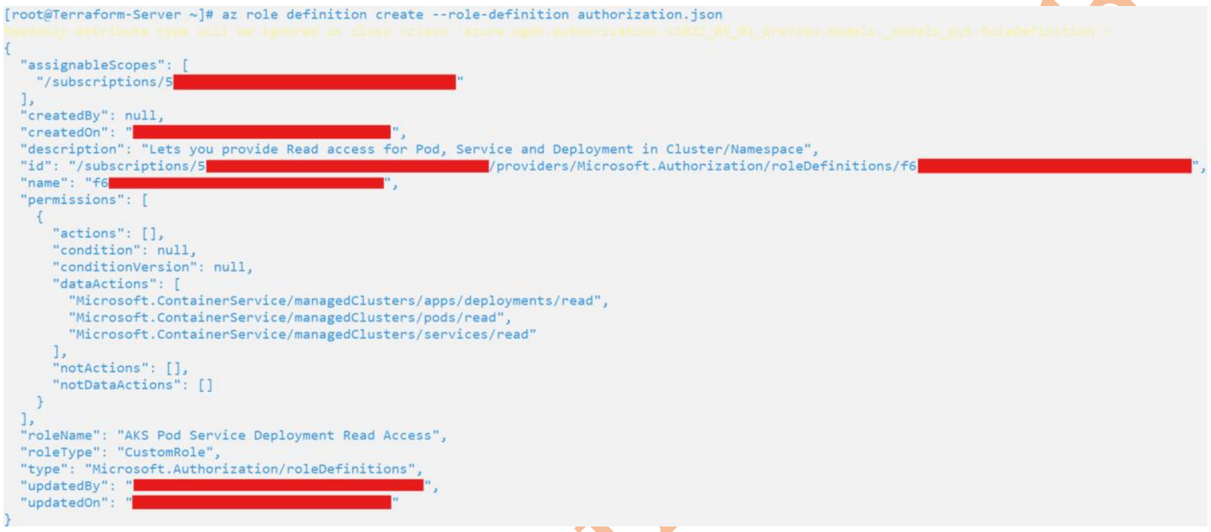
Here I have assigned this Azure Role to Azure AD Group for the Scope as Azure Resource AKS.

vim authorization.json

```
{
  "Name": "AKS Pod Service Deployment Read Access",
  "Description": "Lets you provide Read access for Pod, Service and Deployment in
Cluster/Namespace",
  "Actions": [],
  "DataActions": [
    "Microsoft.ContainerService/managedClusters/apps/deployments/read",
    "Microsoft.ContainerService/managedClusters/pods/read",
    "Microsoft.ContainerService/managedClusters/services/read"
  ],
  "NotDataActions": [],
  "assignableScopes": [
    "/subscriptions/5XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
  ]
}
```

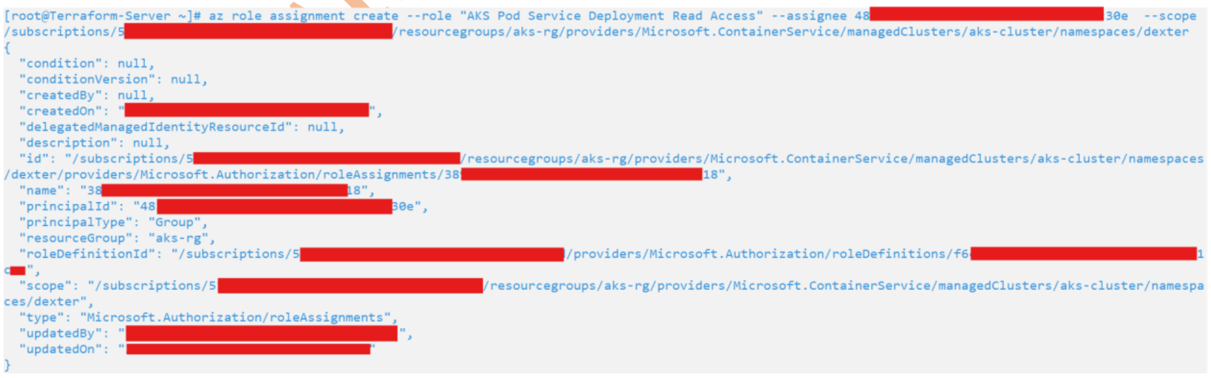


```
[root@Terraform-Server ~]# cat authorization.json
{
   "Name": "AKS Pod Service Deployment Read Access",
   "Description": "Lets you provide Read access for Pod, Service and Deployment in Cluster/Namespace",
   "Actions": [],
   "DataActions": [
      "Microsoft.ContainerService/managedClusters/apps/deployments/read",
      "Microsoft.ContainerService/managedClusters/pods/read",
      "Microsoft.ContainerService/managedClusters/services/read"
   ],
   "NotDataActions": [],
   "assignableScopes": [
      "/subscriptions/5[redacted]"
   ]
}
```



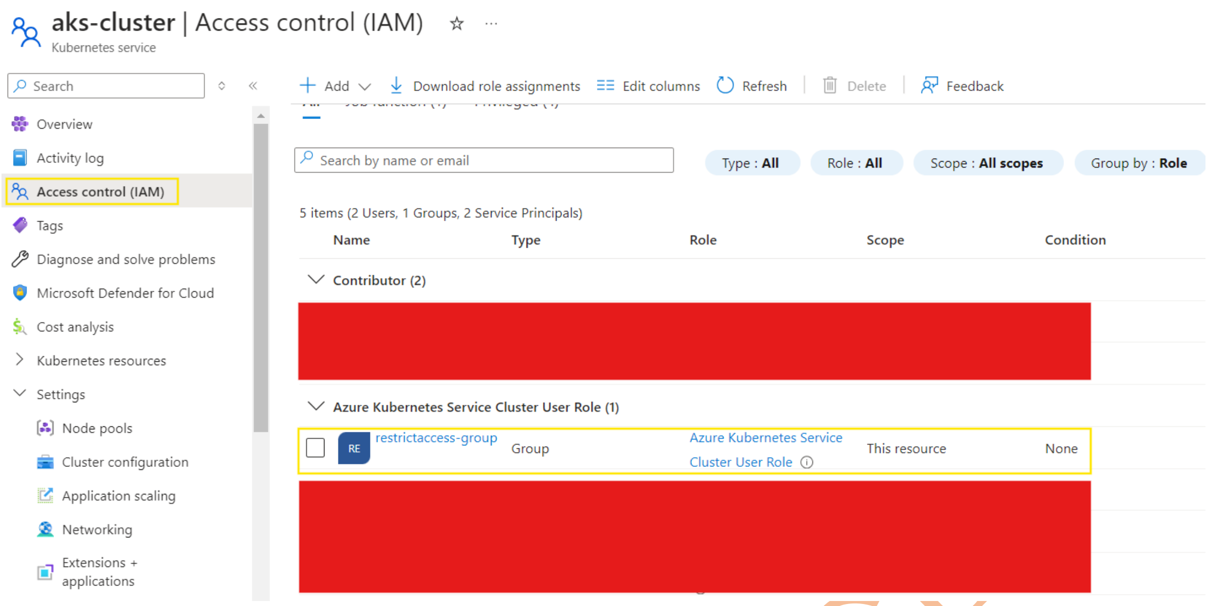
```
[root@Terraform-Server ~]# az role definition create --role-definition authorization.json
Readonly attribute type will be ignored in class <class 'azure.mgmt.authorization.v2022_05_01_preview.models._models_py3.RoleDefinition'>
{
  "assignableScopes": [
    "/subscriptions/5[redacted]"
  ],
  "createdBy": null,
  "createdOn": "[redacted]",
  "description": "Lets you provide Read access for Pod, Service and Deployment in Cluster/Namespace",
  "id": "/subscriptions/5[redacted]/providers/Microsoft.Authorization/roleDefinitions/f6[redacted]",
  "name": "f6[redacted]",
  "permissions": [
    {
      "actions": [],
      "condition": null,
      "conditionVersion": null,
      "dataActions": [
        "Microsoft.ContainerService/managedClusters/apps/deployments/read",
        "Microsoft.ContainerService/managedClusters/pods/read",
        "Microsoft.ContainerService/managedClusters/services/read"
      ],
      "notActions": [],
      "notDataActions": []
    }
  ],
  "roleName": "AKS Pod Service Deployment Read Access",
  "roleType": "CustomRole",
  "type": "Microsoft.Authorization/roleDefinitions",
  "updatedBy": "[redacted]",
  "updatedOn": "[redacted]"
}
```

az role assignment create --role <Name_of_the_Azure_Role as mentioned in the Azure Role definition> --assignee <Object_ID of Group> --scope <Resource_ID of Azure Resource which you can get from Properties tab of Azure Resource>



```
[root@Terraform-Server ~]# az role assignment create --role "AKS Pod Service Deployment Read Access" --assignee 48[redacted]30e  --scope
/subscriptions/5[redacted]/resourcegroups/aks-rg/providers/Microsoft.ContainerService/managedClusters/aks-cluster/namespaces/dexter
{
  "condition": null,
  "conditionVersion": null,
  "createdBy": null,
  "createdOn": "[redacted]",
  "delegatedManagedIdentityResourceId": null,
  "description": null,
  "id": "/subscriptions/5[redacted]/resourcegroups/aks-rg/providers/Microsoft.ContainerService/managedClusters/aks-cluster/namespaces
/dexter/providers/Microsoft.Authorization/roleAssignments/38[redacted]18",
  "name": "38[redacted]18",
  "principalId": "48[redacted]30e",
  "principalType": "Group",
  "resourceGroup": "aks-rg",
  "roleDefinitionId": "/subscriptions/5[redacted]/providers/Microsoft.Authorization/roleDefinitions/f6[redacted]1
c[redacted]",
  "scope": "/subscriptions/5[redacted]/resourcegroups/aks-rg/providers/Microsoft.ContainerService/managedClusters/aks-cluster/namespa
ces/dexter",
  "type": "Microsoft.Authorization/roleAssignments",
  "updatedBy": "[redacted]",
  "updatedOn": "[redacted]"
}
```

To generate kubeconfig file Azure Role **Azure Kubernetes Service Cluster User Role** was assigned to Azure AD Group restrictaccess-group for Azure AKS cluster named as aks-cluster.



aks-cluster | Access control (IAM)

Kubernetes service

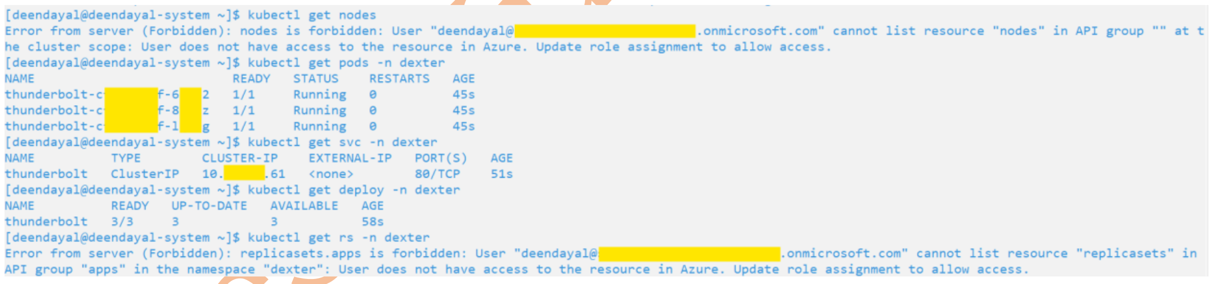
5 items (2 Users, 1 Groups, 2 Service Principals)

Name	Type	Role	Scope	Condition
Contributor (2)				
[redacted]				
Azure Kubernetes Service Cluster User Role (1)				
restrictaccess-group	Group	Azure Kubernetes Service Cluster User Role	This resource	None
[redacted]				

As per the Azure Role the deendayal is only authorize for Read access of Pods, Services and Deployments in namespace dexter which can be verified with the screenshots attached below.

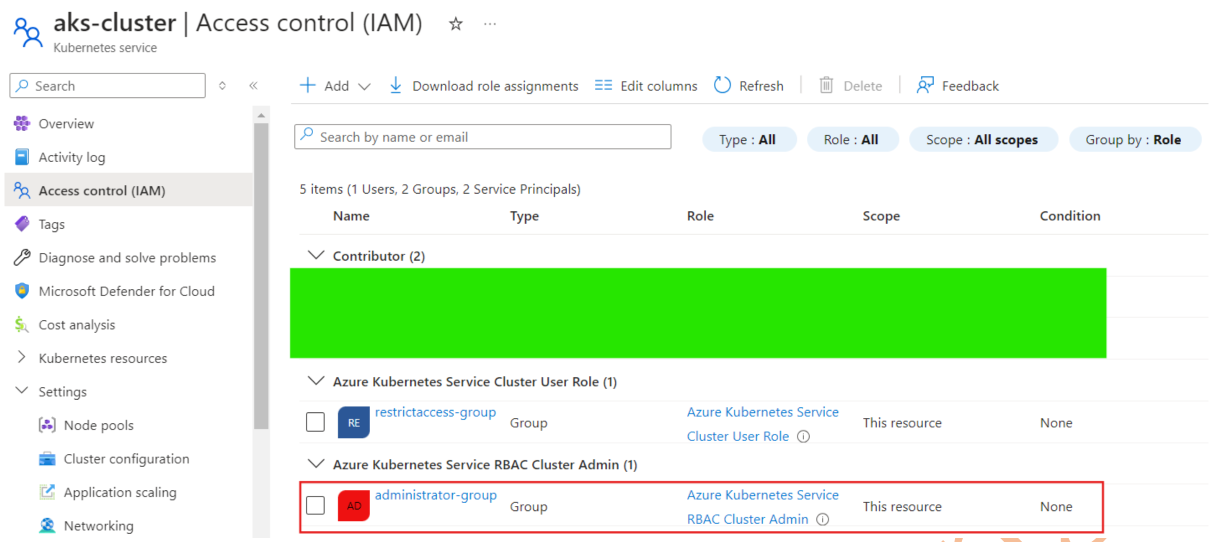
```
[root@deendayal-system ~]# az aks install-cli
```

```
[deendayal@deendayal-system ~]$ az aks get-credentials --resource-group aks-rg --name aks-cluster
Merged "aks-cluster" as current context in /home/deendayal/.kube/config
```



```
[deendayal@deendayal-system ~]$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "deendayal@[redacted].onmicrosoft.com" cannot list resource "nodes" in API group "" at t
he cluster scope: User does not have access to the resource in Azure. Update role assignment to allow access.
[deendayal@deendayal-system ~]$ kubectl get pods -n dexter
NAME                              READY   STATUS    RESTARTS   AGE
thunderbolt-c[redacted]f-6[redacted]2   1/1     Running   0          45s
thunderbolt-c[redacted]f-8[redacted]z   1/1     Running   0          45s
thunderbolt-c[redacted]f-1[redacted]g   1/1     Running   0          45s
[deendayal@deendayal-system ~]$ kubectl get svc -n dexter
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
thunderbolt   ClusterIP   10.[redacted].61     <none>        80/TCP    51s
[deendayal@deendayal-system ~]$ kubectl get deploy -n dexter
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
thunderbolt   3/3     3            3           58s
[deendayal@deendayal-system ~]$ kubectl get rs -n dexter
Error from server (Forbidden): replicasets.apps is forbidden: User "deendayal@[redacted].onmicrosoft.com" cannot list resource "replicasets" in
API group "apps" in the namespace "dexter": User does not have access to the resource in Azure. Update role assignment to allow access.
```

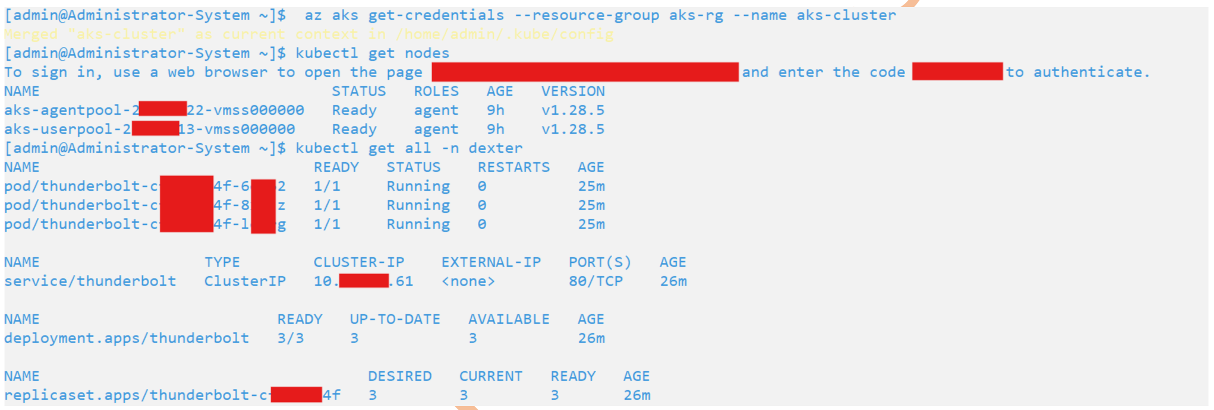
The user admin which belongs to the group administrator-group will have admin privileges as the Role attached with it is **Azure Kubernetes Service RBAC Cluster Admin** and it can be verified with the attached screenshots.



aks-cluster | Access control (IAM)
Kubernetes service

5 items (1 Users, 2 Groups, 2 Service Principals)

Name	Type	Role	Scope	Condition
Contributor (2)				
[redacted]				
Azure Kubernetes Service Cluster User Role (1)				
restrictaccess-group	Group	Azure Kubernetes Service Cluster User Role	This resource	None
Azure Kubernetes Service RBAC Cluster Admin (1)				
administrator-group	Group	Azure Kubernetes Service RBAC Cluster Admin	This resource	None



```
[admin@Administrator-System ~]$  az aks get-credentials --resource-group aks-rg --name aks-cluster
Merged "aks-cluster" as current context in /home/admin/.kube/config
[admin@Administrator-System ~]$ kubectl get nodes
To sign in, use a web browser to open the page [redacted] and enter the code [redacted] to authenticate.
NAME                                    STATUS   ROLES   AGE   VERSION
aks-agentpool-2[redacted]22-vmss000000       Ready    agent   9h    v1.28.5
aks-userpool-2[redacted]13-vmss000000        Ready    agent   9h    v1.28.5
[admin@Administrator-System ~]$ kubectl get all -n dexter
NAME                                  READY   STATUS    RESTARTS   AGE
pod/thunderbolt-c[redacted]4f-6[redacted]2   1/1     Running   0          25m
pod/thunderbolt-c[redacted]4f-8[redacted]z   1/1     Running   0          25m
pod/thunderbolt-c[redacted]4f-1[redacted]g   1/1     Running   0          25m

NAME                  TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/thunderbolt   ClusterIP   10.[redacted].61   <none>        80/TCP    26m

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/thunderbolt   3/3     3            3           26m

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/thunderbolt-c[redacted]4f   3         3         3       26m
```