

AKS Azure AD Integration

Last updated by | Tony Wecker | Feb 18, 2021 at 9:56 AM GMT+1


AKS Azure AD Integration

There are several aspects and terminologies to consider when it comes to integrating Azure AD with Azure Kubernetes Services. This page is intended to provide clarity on the various terms and their respective scopes.

Contents

- [AKS Azure AD Integration](#)
 - [Kubernetes Authentication](#)
 - [Kubernetes Authorization \(RBAC\)](#)
 - [Terraform configuration](#)
 - [Azure Authentication](#)
 - [AKS-managed Azure Active Directory integration](#)
 - [Terraform configuration](#)
 - [Example scenario](#)
 - [Authentication](#)
 - [Authorization](#)
 - [Further details](#)
 - [Howto connect](#)
 - [Non-interactive Logins \(CI/CD\)](#)
 - [Service-principal](#)
 - [Others](#)
 - [Azure RBAC for Kubernetes Authorization \(preview\)](#)

Kubernetes Authentication

Money quote (Source: [Kubernetes docs](#) 

All Kubernetes clusters have two categories of users: service accounts managed by Kubernetes, and normal users.

It is assumed that a cluster-independent service manages normal users in the following ways:

- an administrator distributing private keys
- a user store like Keystone or Google Accounts
- a file with a list of usernames and passwords

In this regard, Kubernetes does not have objects which represent normal user accounts. Normal users cannot be added to a cluster through an API call.

In the context of AKS, this means that Azure also manage the credentials required to connect the authorized user to the cluster. A service account is transparently created within Kubernetes for AKS users, in background, which is used when an Azure user or SP accesses the cluster using the corresponding AKS

credentials. In Kubernetes, users can also create their own service accounts that can operate within the cluster.

Kubernetes Authorization (RBAC)

[Kubernetes Role-based access control](#) ☐, if activated, regulates access to resources based on the roles of individual users within the cluster.

Roles are used to grant permissions within a namespace whereas ClusterRoles grant permissions across the entire Cluster. When Roles are created, they must be assigned to user entities with RoleBindings or ClusterRoleBindings. ClusterRoleBindings can, contrary to the name, apply ClusterRoles to namespaces as well.

Within AKS, it is possible to enable RBAC during creation, but only enables the cluster to use this model, and first has no effect on the permissions of Azure users within Kubernetes without the usage of Azure Active Directory Integration.

Terraform configuration

```
resource "azurerm_kubernetes_cluster" "aks" {
  role_based_access_control {
    enabled = true
  }
}
```

Azure Authentication

Azure Authentication is already in place when using Azure, Azure Active Directory and AKS and is using AAD identities.

Following default roles exist within Azure:

Role	Description
Azure Kubernetes Service Contributor Role	Grants access to read and write Azure Kubernetes Service clusters
Azure Kubernetes Service Cluster Admin Role	List cluster admin credential action.
Azure Kubernetes Service Cluster User Role	List cluster user credential action.

AKS-managed Azure Active Directory integration

[This feature](#) ☐ enables the use of Azure AD Users or Groups within Kubernetes RBAC, so that an cluster administrator can configure RoleBindings or ClusterRoleBindings based on a user’s identity or directory group membership. Azure AD authentication is provided to AKS clusters with OpenID Connect.

Kubernetes RBAC must be activated in order to use AKS-managed AAD. To learn about how to use these integration within AKS, refer to:

Terraform configuration

```
resource "azurerm_kubernetes_cluster" "aks" {  
  role_based_access_control {  
    enabled = true  
    azure_active_directory {  
      managed = true  
      admin_group_object_ids = [ "<add admin group object id>" ]  
    }  
  }  
}
```

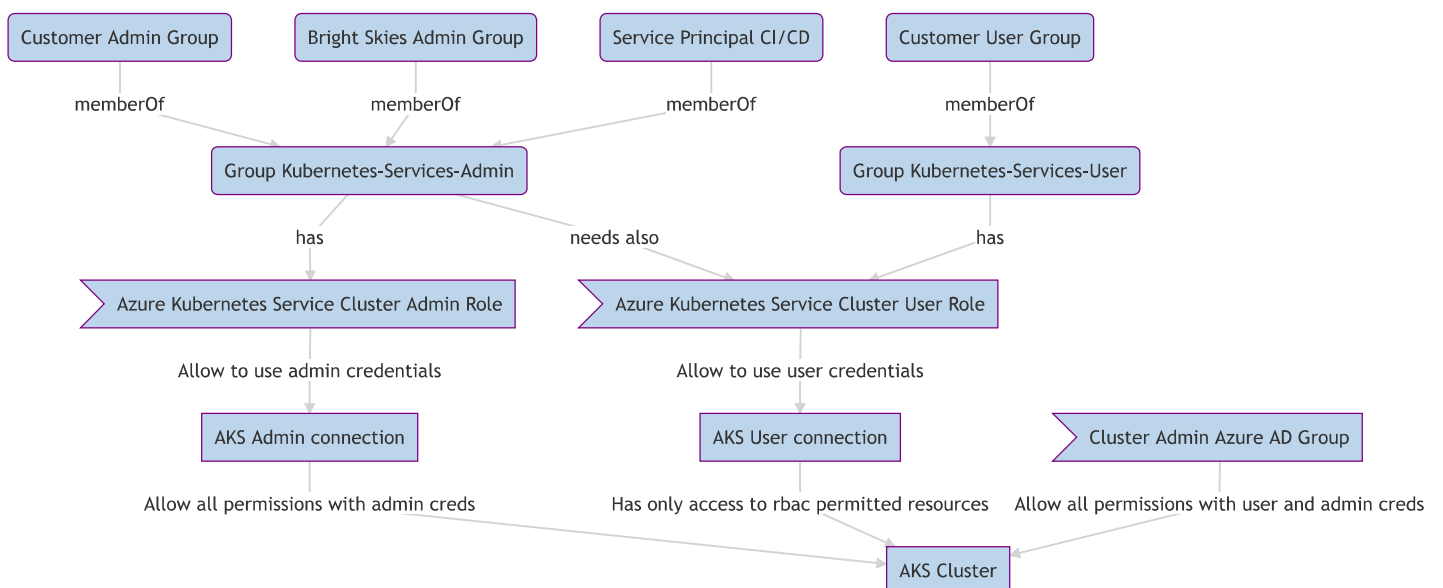
⚠ With activated AAD integration, Cluster User permissions fall back to a minimum set of permissions, which just allows to connect to AKS. To provide further permissions, appropriate Kubernetes RBAC Roles has to be assigned to the User or Group.

🔗 There is also the option to define an admin group for each Kubernetes cluster. Members of this group automatically have admin privileges without having to download separate admin credentials.

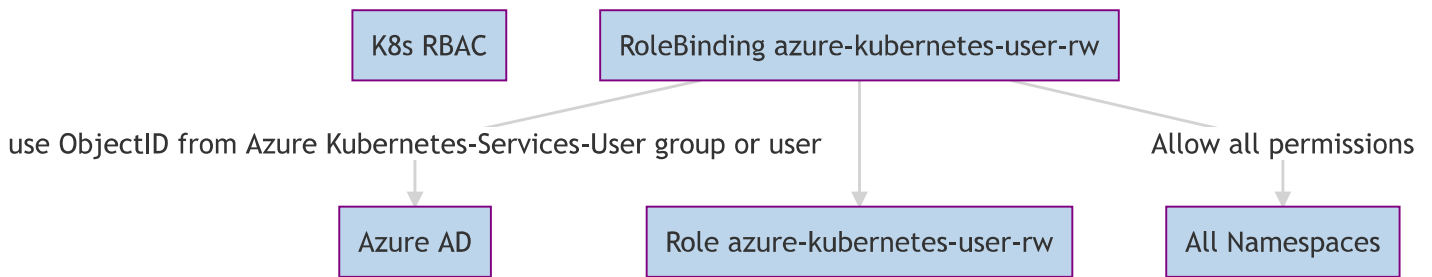
Example scenario

Authentication

- The Customer is using one developer group which should have non-administrative access to AKS
- The Customer is using one admin group which should have administrative access to AKS
- Bright Skies should be able to administer the cluster, for what the BSkies Group also needs admin access
- Jenkins will be used as CI/CD Tool, which should have access to cluster-wide resources



Authorization



An Kubernetes RBAC ClusterRole and RoleBinding for Read/Write Access could look like as follows. Please note the Azure AD Group Object ID at the `subjects` definition.

```

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: azure-kubernetes-user-rw
rules:
- apiGroups: ["", "extensions", "apps"]
  resources: ["*"]
  verbs: ["*"]
- apiGroups: ["batch"]
  resources:
  - jobs
  - cronjobs
  verbs: ["*"]
- apiGroups:
  - traefik.containo.us
  - cert-manager.io
  - coreos.com
  resources: ["*"]
  verbs: ["*"]
- apiGroups: ["apiextensions.k8s.io"]
  resources: ["customresourcedefinitions"]
  verbs: ["*"]
- apiGroups:
  - rbac.authorization.k8s.io
  - authorization.k8s.io
  resources: ["*"]
  verbs: ["get", "list", "watch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: azure-kubernetes-user-rw
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: azure-kubernetes-user-rw
subjects:
- kind: Group
  name: 82455555-555a-444b-333c-98101f1f1337

```

Further details

- It is possible to assign roles in nested group scenarios
- Subscription/AKS Contributors lose access to cluster resources, if they are not part of the Cluster Admin group, nor has any appropriate roles assigned
- Azure Kubernetes Service RBAC Roles have no effect, if the preview feature for Azure RBAC is not enabled

Howto connect

Azure Managed Identities are used for login, therefore login via Azure CLI and then again via kubectl is required for access.

⚠ Each time the user/group mappings are changed, the login via Azure CLI and the AKS credentials must be renewed.

The following commands can be used to download the KUBECONFIG, alternatively as usual via the "Connect" link of the Kubernetes cluster in the Azure Portal.

```
az login
az account set --subscription <subscriptionid>
az aks get-credentials --resource-group <resource group name> --name <aks cluster name>
kubectl get pods
> To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code CT9Gk
```

Non-interactive Logins (CI/CD)




Howto connect to the cluster, using Jenkins as an example.

Service-principal

Requirements:

- Service Principals has password/secret configured
- Reader permissions on subscription
- Permissions to connect to the cluster via Azure Kubernetes Service roles

Requirements Jenkins CI/CD:

- [Kubernetes CLI Plugin](#) 
- [az cli](#) 
- [kubelogin](#) 

One-time download of Kubeconfig and converting is necessary:

```
az login --service-principal --username <sp application/client id> --password "<sp password/secret>" --tenant <tenant id>

# ggf. wegsichern der aktuellen Config
mv ~/.kube/config{,.backup}

az aks get-credentials --resource-group <resource group name> --name <aks cluster name> # --admin if needed

export KUBECONFIG=~/.kube/config
kubelogin convert-kubeconfig -l spn

export AAD_SERVICE_PRINCIPAL_CLIENT_ID="<sp application/client id>"
export AAD_SERVICE_PRINCIPAL_CLIENT_SECRET="<sp password/secret>"

kubectl -n default get all
```

In **Jenkins**, the resulting KUBECONFIG can be stored as a credential, as well as the service principal information.

As "Secret File" Credential:
Content of ~/.kube/config
(Make sure it's containing only the needed context)

As "Username with password" Credential:
AAD_SERVICE_PRINCIPAL_CLIENT_ID=<sp application/client id>
AAD_SERVICE_PRINCIPAL_CLIENT_SECRET=<sp password/secret>

Example:

```
pipeline {
  agent any

  stages {
    stage('Azure AKS RBAC') {
      steps {
        withCredentials([usernamePassword(
          credentialsId: 'jenkinscid-service-principal',
          passwordVariable: 'AAD_SERVICE_PRINCIPAL_CLIENT_SECRET',
          usernameVariable: 'AAD_SERVICE_PRINCIPAL_CLIENT_ID')]) {
          withKubeConfig(credentialsId: 'jenkinscid-kubernetes-config') {
            script {
              sh "kubectl -n default get pods"
              sh "helm -n default list"
            }
          }
        }
      }
    }
  }
}
```

Others

Further methods are described on the [kubelogin](#) page.

Azure RBAC for Kubernetes Authorization (preview)

Using [Azure RBAC for Kubernetes Authorization](#) is a preview feature, that builds on top of the authentication of AKS-managed AAD, and allows administrators to use Azure RBAC for authorization within Kubernetes, in addition to Kubernetes RBAC. For this purpose, there are built-in Azure roles which manages Kubernetes permissions:

Role	Description
Azure Kubernetes Service RBAC Reader	Allows read-only access to see most objects in a namespace.
Azure Kubernetes Service RBAC Writer	Allows read/write access to most objects in a namespace.
Azure Kubernetes Service RBAC Admin	Allows admin access, intended to be granted within a namespace.
Azure Kubernetes Service RBAC Cluster Admin	Allows super-user access to perform any action on any resource.