Secure AKS at deployment

FLORENT APPOINTAIRE

DAVID FRAPPART

Secure AKS at deployment

- Azure Kubernetes service offers a managed service for container orchestration
- But how to leverage a secure AKS at deployment ?
- In this session, we will cover
  - how to deploy an AKS cluster in IaC
  - How to integrate authentication and RBAC with Azure AD
  - How to secure network traffic between pods with Kubernetes network policies
  - And a brief overview of features to come in AKS
  - Integration with Azure Security Center

Agenda

- The speakers IDs, in a few words
- AKS reminder (and K8S ?)
- Securing AKS at Deployment, what does it mean ?
- AAD integration
- RBAC binding in AKS
- Using Network Policies for more security
- Azure Security Center integration
- And then ? AKS features roadmap

David Frappart

- Cloud Architect @devoteam
- Agile IT core team Cloud
- ~ 15 years of experiences in IT
- A few cloud certifications:
  - Azure
  - AWS
  - GCP
- Recently nominated MVP, because I speak a lot
- Fonds of Terraform as a IaC tool
- Currently decrypting the complexity of the K8S for my clients, thus the talk tonight

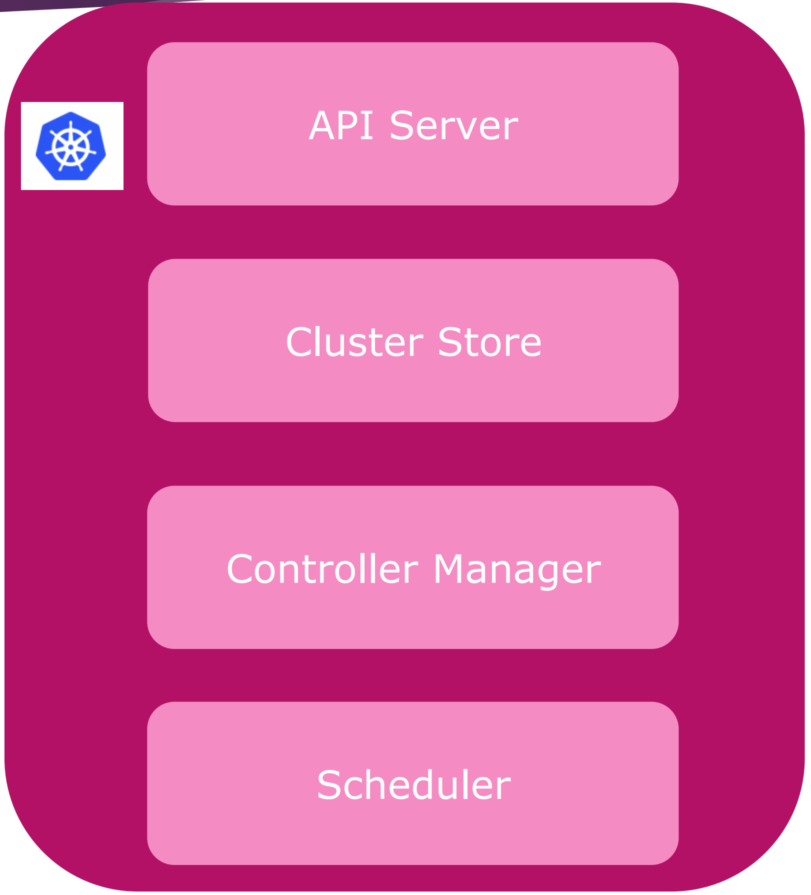
Florent Appointaire

- Microsoft MVP Azure (4 times)
- Azure Solution Architect certified
- Membre aOS/SCUGBE
- Freelance Cloud Architect
  - florent@falaconsulting.be
- CSP Tier 2
- +7 ans d'expérience
- Speaker (MMSMOA, ELEU, ELNL, aOS, etc.)
- Blog: https://cloudyjourney.fr
- Twitter: @florent_app

AKS Reminder

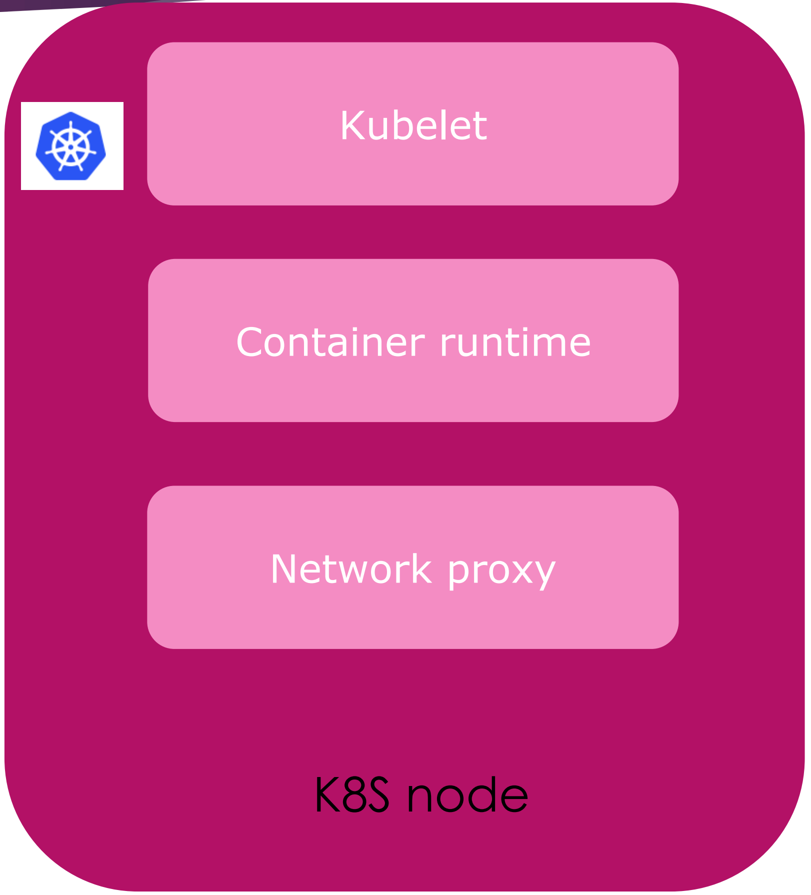
Kubernetes Architecture – The control plane

- API Server – the brain of the cluster
  - Front End into the K8S control plane.
  - Exposes a RESTful API.
  - Manifest files are posted there and the work they define gets deployed to the cluster
- Cluster store – the memory of the cluster
  - Config and state gets persistently stored in the cluster store
  - The only stateful component
  - Based on etcd
- Controller Manager
  - Implement functions such as node controller, namespace controller…
  - Watch for change and ensure that the current state matchthe desired state
- Scheduler
  - Assign new workload to nodes
  - Evaluate affinity and anti affinity, constraints and resource mgmt



Kubernetes Architecture – The nodes

- Kubelet
  - Main agent running on nodes
  - Watch the API server for new work of assignment
  - Exposes an endpoint on TCP 10255
- Container runtime
  - Container related actions such as pull, start and stop
  - Can be docker or any CRI compatible engine
- Kube proxy
  - Network brain of the node
  - Make sure that every pod gets its own unique IP
  - Light weight load balancing



Azure Kubernetes reminder

- The managed kubernetes offer from Azure
  - RBAC with Azure AD capabilities
  - Managed control plane
  - An Azure customised CNI
  - And other stuff
- A real K8S following (relatively) closely the community release
- Very easy to deploy…

Azure Kubernetes – some constraint to be aware of

- Managed Master
- An Azure customized CNI
- Possible lag in new K8S features
- Still a quite young service in the cloud point of view
- Platform as a service => meaning public endpoint for control plane (for now)

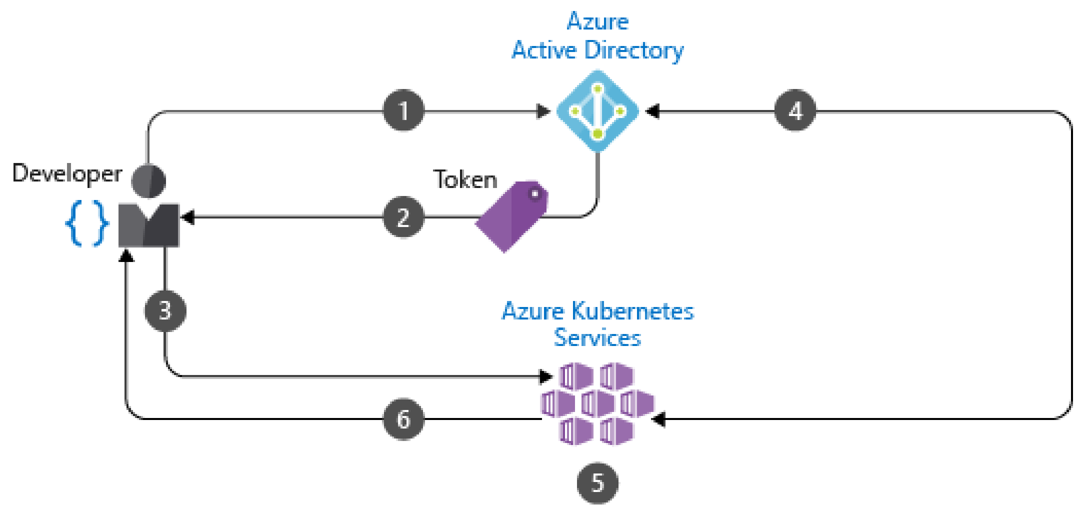
Securing AKS at Deployment, what does it mean ?

Securing AKS

- Authentication
  - Use RBAC
  - Delegate Identity to 3rd party provider
  - Implement MFA
- Network access
  - Filter traffic between pods
  - Filter traffic to control plane

AAD Integration

RBAC in AKS with Azure AD in picture



1. User authenticate with Azure AD
2. The Azure AD token issuance endpoint issues the access token.
3. The user performs an action using the Azure AD token, such as kubectl create pod
4. K8S validates the token with AAD & fetches the user’s group memberships.
5. K8S RBAC and cluster policies are applied.
6. User’s request is successful or not based on previous validation of AAD group membership and K8S RBAC and policies.

AKS integration with AAD under the hood

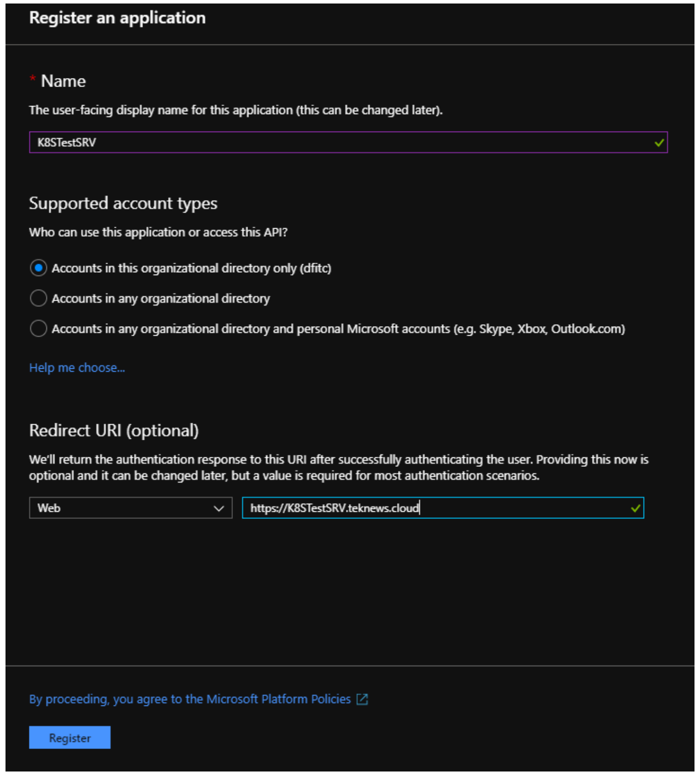
- AAD authentication relies on an OpenId Connect layer in AKS
- It works with 2 AAD Applications
  - A server application

    The server component that will provides user authentication
    AKS is able to check the user information through this couple AAD App / AAD SP
    The App needs to be able to read Directory data
    No access on AKS is configured yet !

  - A client application

    A native app with access to the server app
    This is the app that is referred to when a user authenticate on AKS
- Granting access to the application requires an admin validation

Create the Azure AD Server app- Portal



Register an application

* Name

The user-facing display name for this application (this can be changed later).

K8STestSRV

Supported account types

Who can use this application or access this API?

- [✓ Accounts in this organizational directory only (dfitc)]
- Accounts in any organizational directory
- Accounts in any organizational directory and personal Microsoft accounts (e.g. Skype, Xbox, Outlook.com)

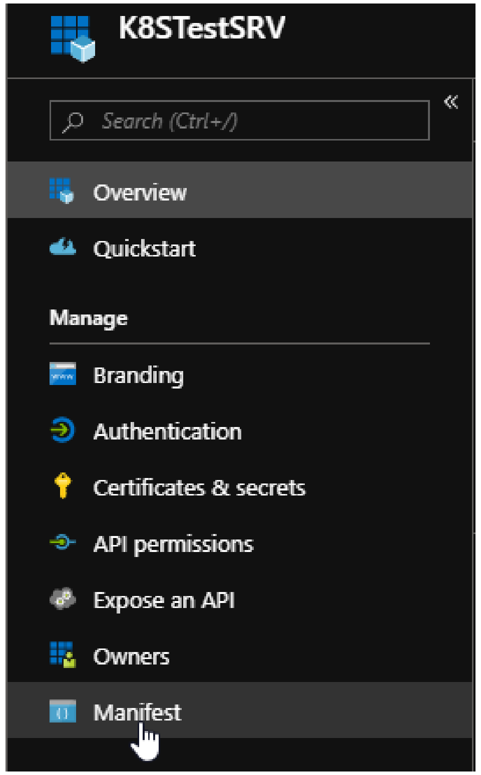
Help me choose...

Redirect URI (optional)

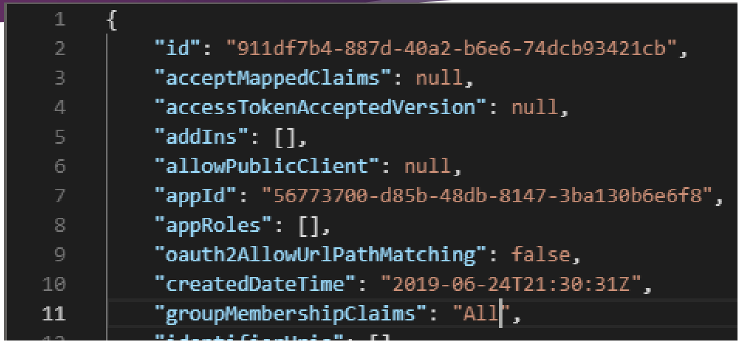
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web | https://K8STestSRV.teknews.cloud

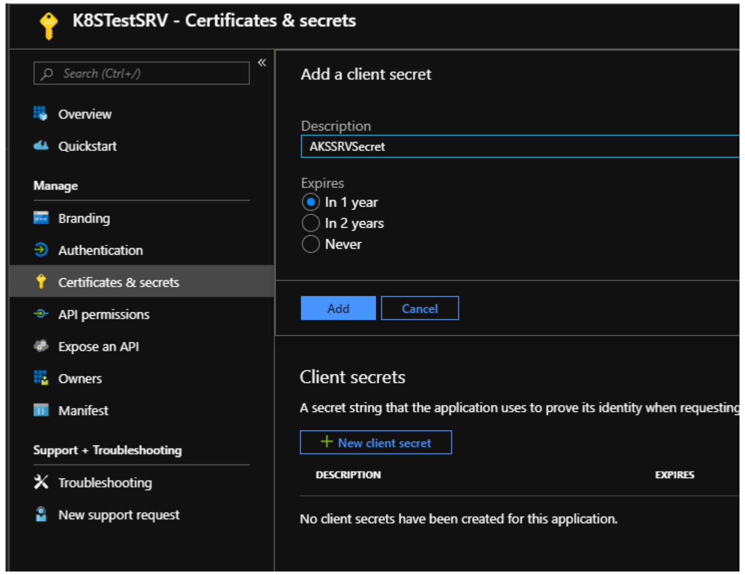
By proceeding, you agree to the Microsoft Platform Policies



K8STestSRV



```
{
    "id": "911df7b4-887d-40a2-b6e6-74dcb93421cb",
    "acceptMappedClaims": null,
    "accessTokenAcceptedVersion": null,
    "addIns": [],
    "allowPublicClient": null,
    "appId": "56773700-d85b-48db-8147-3ba130b6e6f8",
    "appRoles": [],
    "oauth2AllowUrlPathMatching": false,
    "createdDateTime": "2019-06-24T21:30:31Z",
    "groupMembershipClaims": "All",
```



K8STestSRV - Certificates & secrets

Add a client secret

Description: AKSSRVSecret

Expires

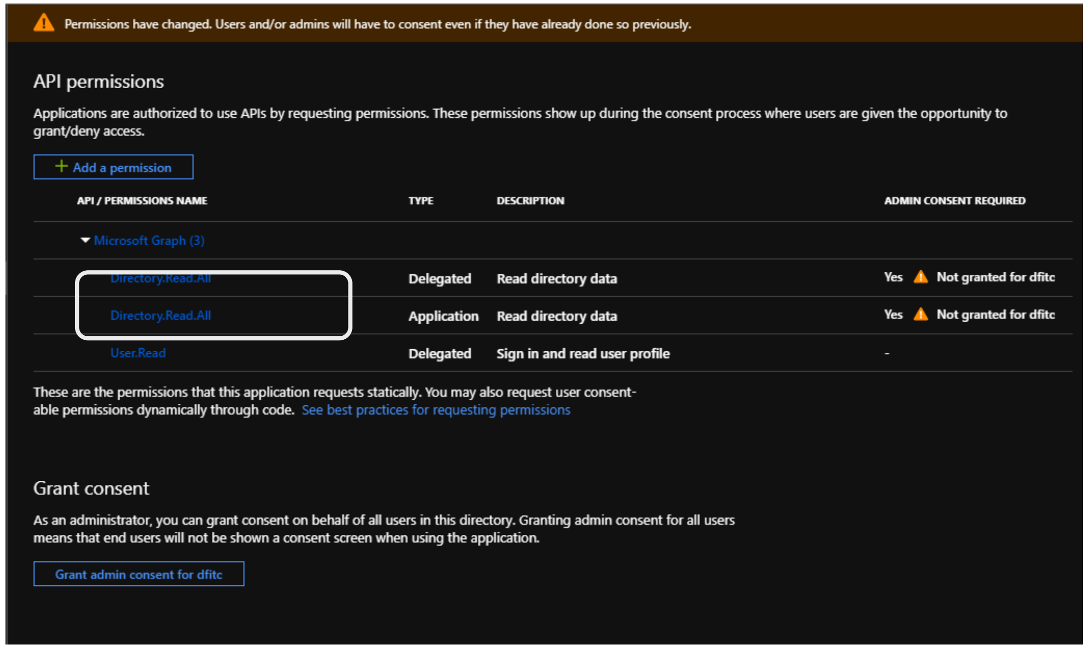
- [✓ In 1 year]
- In 2 years
- Never

Client secrets

A secret string that the application uses to prove its identity when requesting

DESCRIPTION	EXPIRES
No client secrets have been created for this application.	

Create the Azure AD Server app- Portal



Permissions have changed. Users and/or admins will have to consent even if they have already done so previously.

API permissions

Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.

+ Add a permission

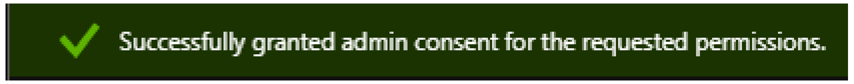
API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED	
▾ Microsoft Graph (3)				
Directory.Read.All	Delegated	Read directory data	Yes	⚠ Not granted for dfitc
Directory.Read.All	Application	Read directory data	Yes	⚠ Not granted for dfitc
User.Read	Delegated	Sign in and read user profile	-	

These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. See best practices for requesting permissions

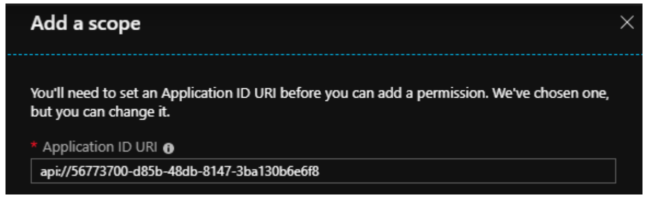
Grant consent

As an administrator, you can grant consent on behalf of all users in this directory. Granting admin consent for all users means that end users will not be shown a consent screen when using the application.

Grant admin consent for dfitc



✓ Successfully granted admin consent for the requested permissions.

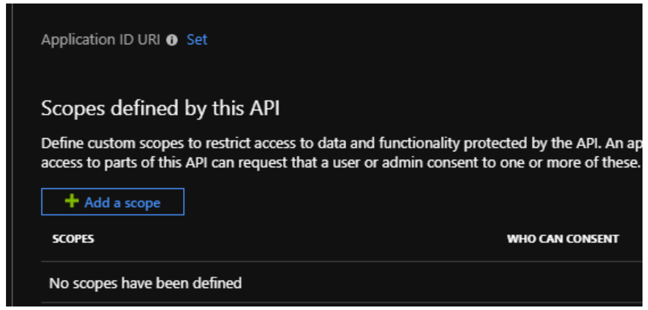


Add a scope

You'll need to set an Application ID URI before you can add a permission. We've chosen one, but you can change it.

* Application ID URI

api://56773700-d85b-48db-8147-3ba130b6e6f8



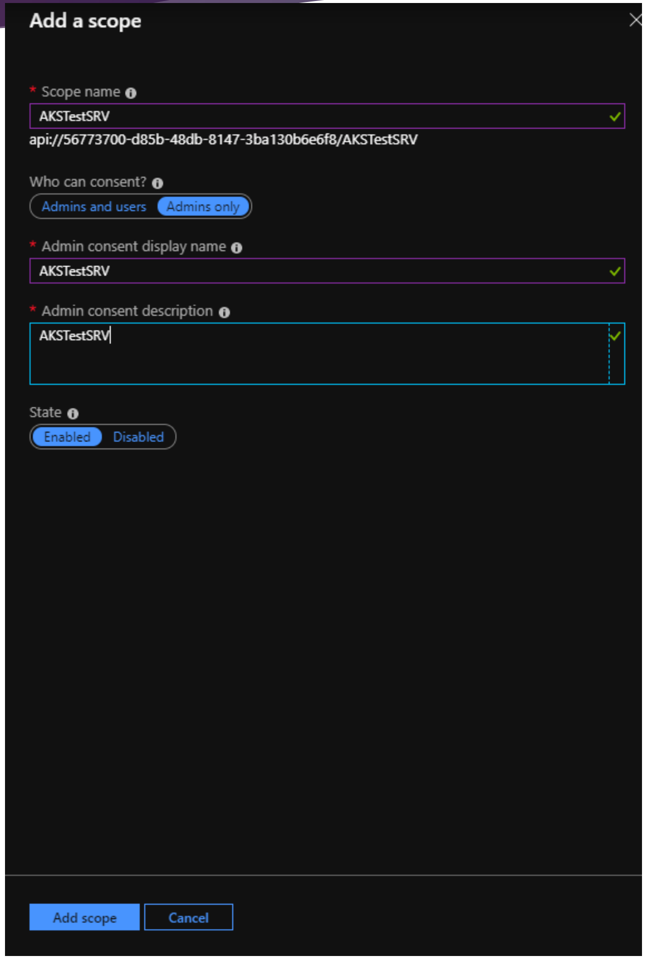
Application ID URI Set

Scopes defined by this API

Define custom scopes to restrict access to data and functionality protected by the API. An ap access to parts of this API can request that a user or admin consent to one or more of these.

+ Add a scope

SCOPES	WHO CAN CONSENT
No scopes have been defined	



Add a scope

* Scope name

AKSTestSRV

api://56773700-d85b-48db-8147-3ba130b6e6f8/AKSTestSRV

Who can consent?

Admins and users [✓ Admins only]

* Admin consent display name

AKSTestSRV

* Admin consent description

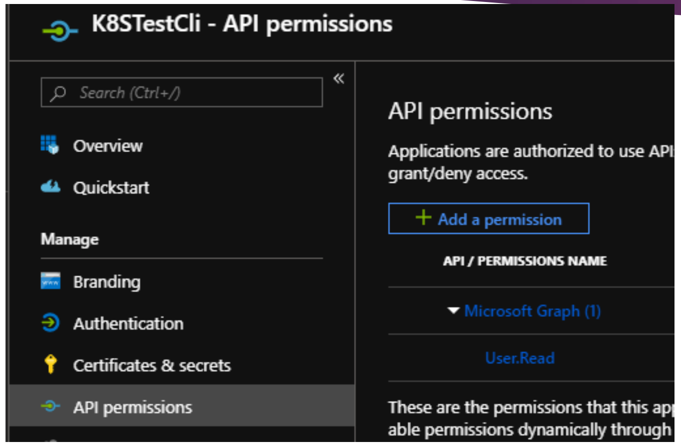
AKSTestSRV

State

[✓ Enabled] Disabled

Add scope Cancel

Create the Azure AD Cli app- Portal


K8STestCli - API permissions

Search (Ctrl+/)

Overview
Quickstart

Manage
Branding
Authentication
Certificates & secrets
API permissions

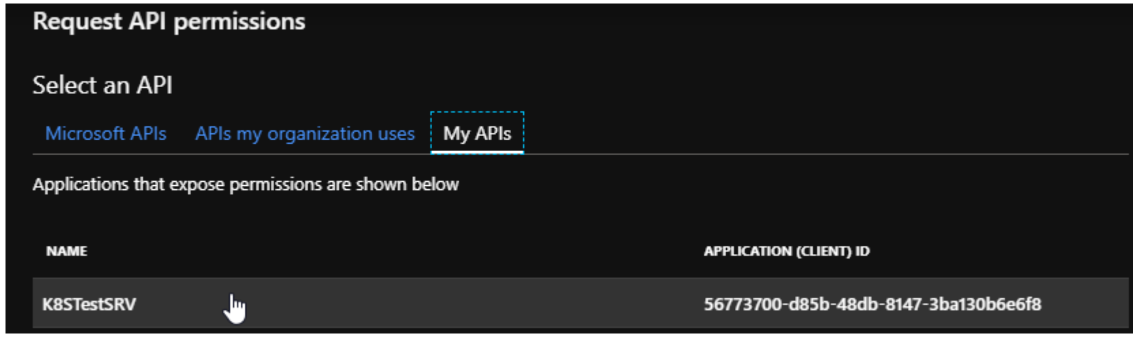
API permissions

Applications are authorized to use API
grant/deny access.

+ Add a permission

API / PERMISSIONS NAME
Microsoft Graph (1)
User.Read

These are the permissions that this ap
able permissions dynamically through

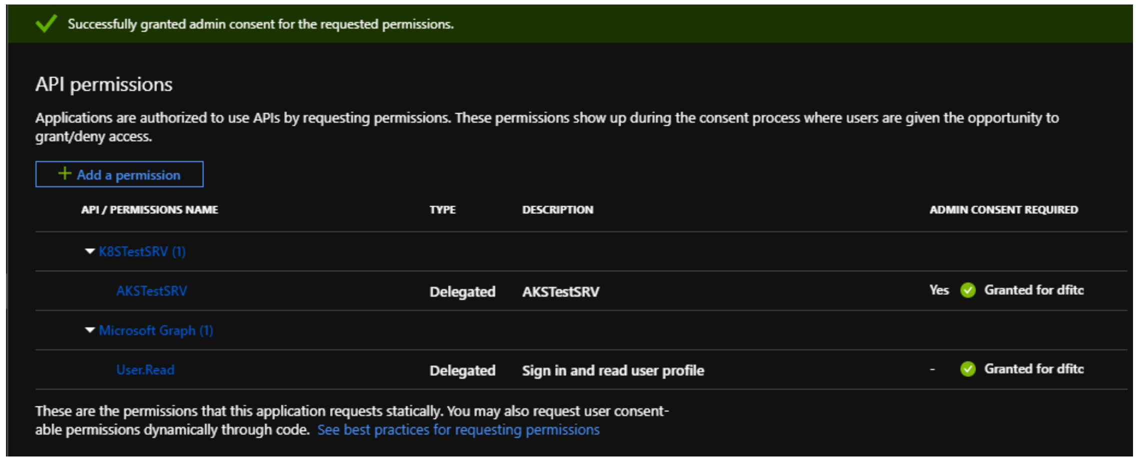

Request API permissions

Select an API

Microsoft APIs | APIs my organization uses | My APIs

Applications that expose permissions are shown below

NAME	APPLICATION (CLIENT) ID
K8STestSRV	56773700-d85b-48db-8147-3ba130b6e6f8


✓ Successfully granted admin consent for the requested permissions.

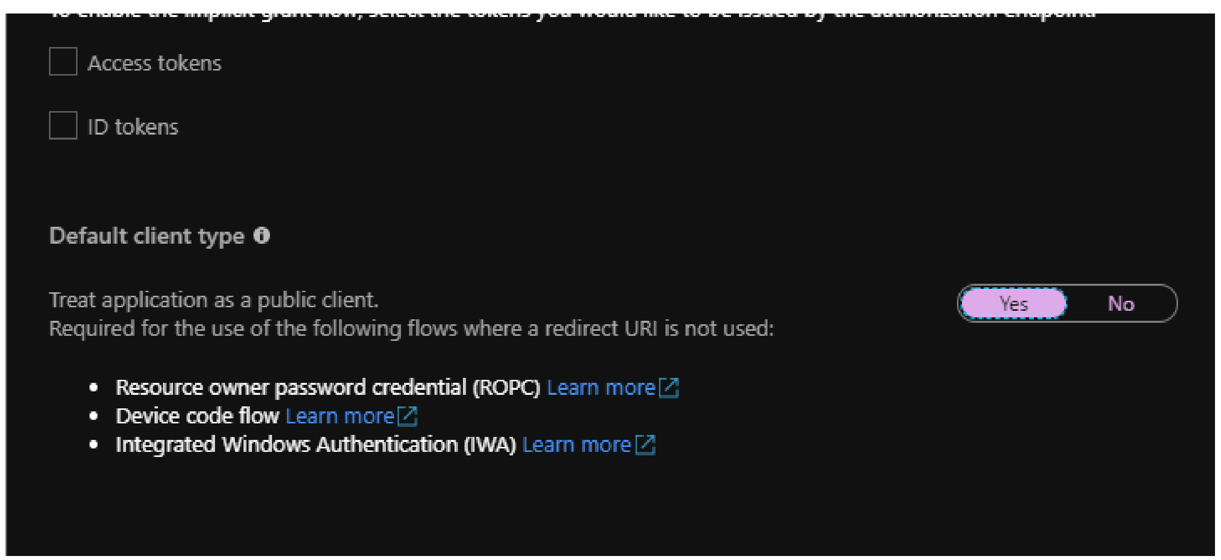
API permissions

Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.

+ Add a permission

API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED	
K8STestSRV (1)				
AKSTestSRV	Delegated	AKSTestSRV	Yes	✓ Granted for dfitc
Microsoft Graph (1)				
User.Read	Delegated	Sign in and read user profile	-	✓ Granted for dfitc

These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. See best practices for requesting permissions


To enable the implicit grant flow, select the tokens you would like to be issued by the authorization endpoint:

☐ Access tokens

☐ ID tokens

Default client type

Treat application as a public client. [✓ Yes] No
Required for the use of the following flows where a redirect URI is not used:

- Resource owner password credential (ROPC) Learn more
- Device code flow Learn more
- Integrated Windows Authentication (IWA) Learn more

Create the Azure AD Server app - Az Cli

- Az Cli command

```
$serverApplicationId=$(az ad app create --display-name K8SSRV --identifier-uris
"https://K8SSRV.teknews.cloud" --query appId -o tsv)
```

```
az ad app update --id $serverApplicationId --set groupMembershipClaims=All
```

```
$serverApplicationSecret=$(az ad sp credential reset --name $serverApplicationId
--credential-description K8SSRVPwd --query password -o tsv)
```

```
az ad app permission add --id $serverApplicationId --api 00000003-0000-0000-c000-000000000000
--api-permissions e1fe6dd8-ba31-4d61-89e7-88639da4683d=Scope
06da0dbc-49e2-44d2-8312-53f166ab848a=Scope 7ab1d382-f21e-4acd-a863-ba3e13f7da61=Role

Invoking "az ad app permission grant --id xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx
--api 00000003-0000-0000-c000-000000000000" is needed to make the change effective
```

```
az ad app permission grant --id $serverApplicationId --api 00000003-0000-0000-c000-000000000000
```

```
az ad app permission admin-consent --id $serverApplicationId
```

Create the Azure AD Client app - Az Cli

- Az Cli command

```
$clientApplicationId=$(az ad app create --display-name K8SCki --native-app --
reply-urls "https://K8SCli.teknews.cloud" --query appId -o tsv)
```

```
az ad sp create --id $clientApplicationId
```

```
$serverApplicationSecret=$(az ad sp credential reset --name $serverApplicationId
--credential-description K8SSRVPwd --query password -o tsv)
```

```
$oAuthPermissionId=(az ad app show --id $serverApplicationId --query oauth2Permissions[0].id)
```

Store the credentials securely for future deployment

- AAD app interact with AAD tenant
- Secure the environment by avoiding the hard coded key in the Infra code
- Leverage Azure Key Vault to secure storage of secret

Deploy AKS with AAD integration through Terraform

```
role_based_access_control {
   enabled                = true

   azure_active_directory {
     client_app_id         = "${var.AADCliAppId}"
     server_app_id         = "${var.AADServerAppId}"
     server_app_secret     = "${var.AADServerAppSecret}"
     tenant_id             = "${var.AADTenantId}"
   }

 }
```

- Dedicated Block for RBAC
- Enabled
- AAD information block uses AAD app ID and secret, Tenant Id

RBAC binding in AKS

RBAC in K8S

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: contoso-cluster-admins
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: "david@teknews.cloud"
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: contoso-cluster-admins
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"
```

- Managed through roles and Role Bindings
- Cluster Roles span the whole cluster
- Roles can be associated to namespace
- 2 roles used here
  - Cluster-Admins on the Cluster
  - Admins on a target namespace

Create Binding for AKS

- With Az Cli and Kubectl

```
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: contoso-cluster-admins
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: "david@teknews.cloud"
```

Create Binding for AKS

- Directly with terraform, mixing AzureRM provider and Kubernetes Provider

```
provider "kubernetes" {
    host                        = "${module.AKSClus.KubeAdminCFG_HostName}"
    client_certificate          = "${base64decode(module.AKSClus.KubeAdminCFG_ClientCertificate)}"
    client_key                  = "${base64decode(module.AKSClus.KubeAdminCFG_ClientKey)}"
    cluster_ca_certificate      = "${base64decode(module.AKSClus.KubeAdminCFG_ClusCACert)}"
}
```

```
##################################################################
# associate user & groups to cluster admin role

resource "kubernetes_cluster_role_binding" "Terra_builtin_clubsteradmin_binding_user" {

    metadata {
        name        = "terracreated-clusteradminrole-binding-user"
    }
    role_ref {
        api_group   = "rbac.authorization.k8s.io"
        kind        = "ClusterRole"
        name        = "cluster-admin"
    }
    subject {
        api_group   = "rbac.authorization.k8s.io"
        kind        = "User"
        name        = "${var.AKSClusterAdminUSer}"
    }
}
```

Demo

CHECK AAD AUTHENTICATION

CHECK IF MFA IS WORKING

Using Network Policies for more security

Network Policies

```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny-all
  namespace: terra-test-namespace
spec:
  podSelector: {}
  ingress: []
```

```hcl
resource "kubernetes_network_policy" "terra_defaultnp_denyallin_ns_terra-test-namespace" {
  metadata {
    name         = "defaultnp-denyall-in"
    namespace    = "${kubernetes_namespace.terra_test_namespace.metadata.0.name}"
  }
  spec {
    pod_selector {}
    ingress = []
    policy_types = ["Ingress"]
  }
}
```

- Network policies filter traffic between pods
- Secure AKS with Default deny all ingress

Demo

CHECK THE DEFAULT NETPOL

ADD A SPECIFIC NETPOL FOR ALLOWING TRAFFIC

Azure Security Center integration

Azure Security Center

- Threat Detection
- Scan container images in ACR
- Currently in Preview
- Powered by Qualys
- Equivalent:
  - Aqua
  - Twistlock
  - Neuvector
  - Etc.
- Based on CIS Docker Benchmarks
- https://www.cisecurity.org/benchmark/docker/

Demo

SCAN REGISTRY

And then ?
AKS other features

AKS Features, preview and in development

- Preview: Allowed range on API Server
- Preview: Pod Security Policy
- In development: Pod Identity
- In development: Private Cluster
- Available: Availability Zone
- Available: Calico Net Pol with Kubenet

THANKS TO

ONIRYX
Make IT happen

CORILUS
Connecting Care

digital
wallonia
.be

GENESIS CONSULT
IT Consultancy Services

Microsoft

Références

Satellit
DEDICATED EXPERTS DELIVER

ingenico
ePayments