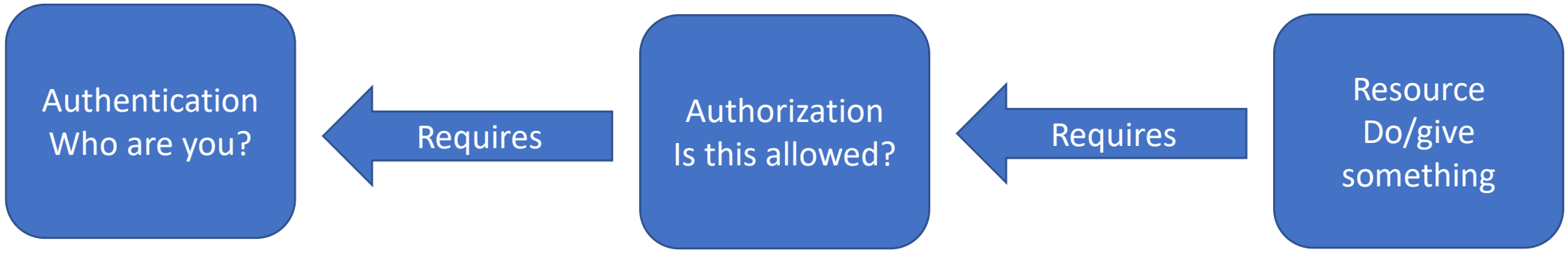
Kubernetes and AAD Workload Identity

Marius Rochon

Issue

Control application access to APIs

Authentication and authorization



Critical: protection of a secret/unique material

Critical: consistent control surface

Critical: secure validation

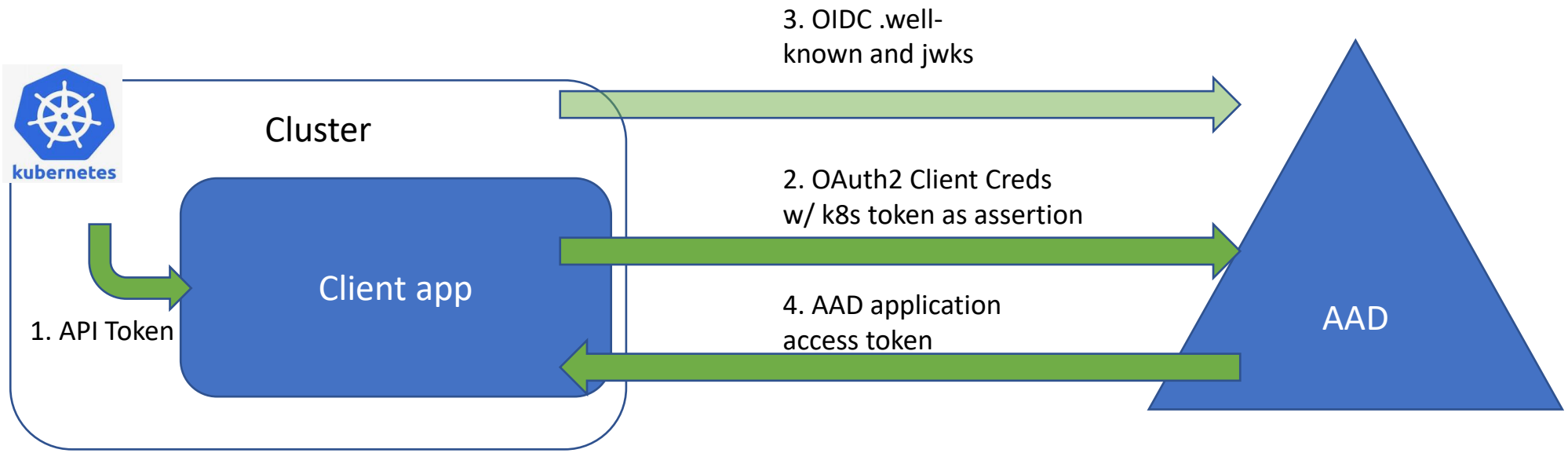
Two scenarios

- Cluster as consumer
- Cluster as resource

Cluster as consumer

- Use existing control surface (Azure AD) when available
- Use OAuth2
- Protect token signing keys/secrets
- Deployment agnostic (AKS or not)

K8s workflow identity



Basic process

Cluster

- Enable OIDC metadata discovery
- Get cluster OIDC .well-known url

Pod/deployment

- Create a service account
- Assign it to a pod
- Configure pod to have access to service account’s k8s API token

AAD

- Register client as application
- Configure federated identity for the app
- Assign required API permissions

Cluster

- AKS
  - Requires aks-preview and WorkloadIdentity enabled
  - Az aks create … --enable-oidc-issuer
  - Get issuer uri: az aks show –n… --query "oidcIssuerProfile.issuerUrl"

- Minikube? – is it possible?
- Other, common k8s implementations?

Deployment

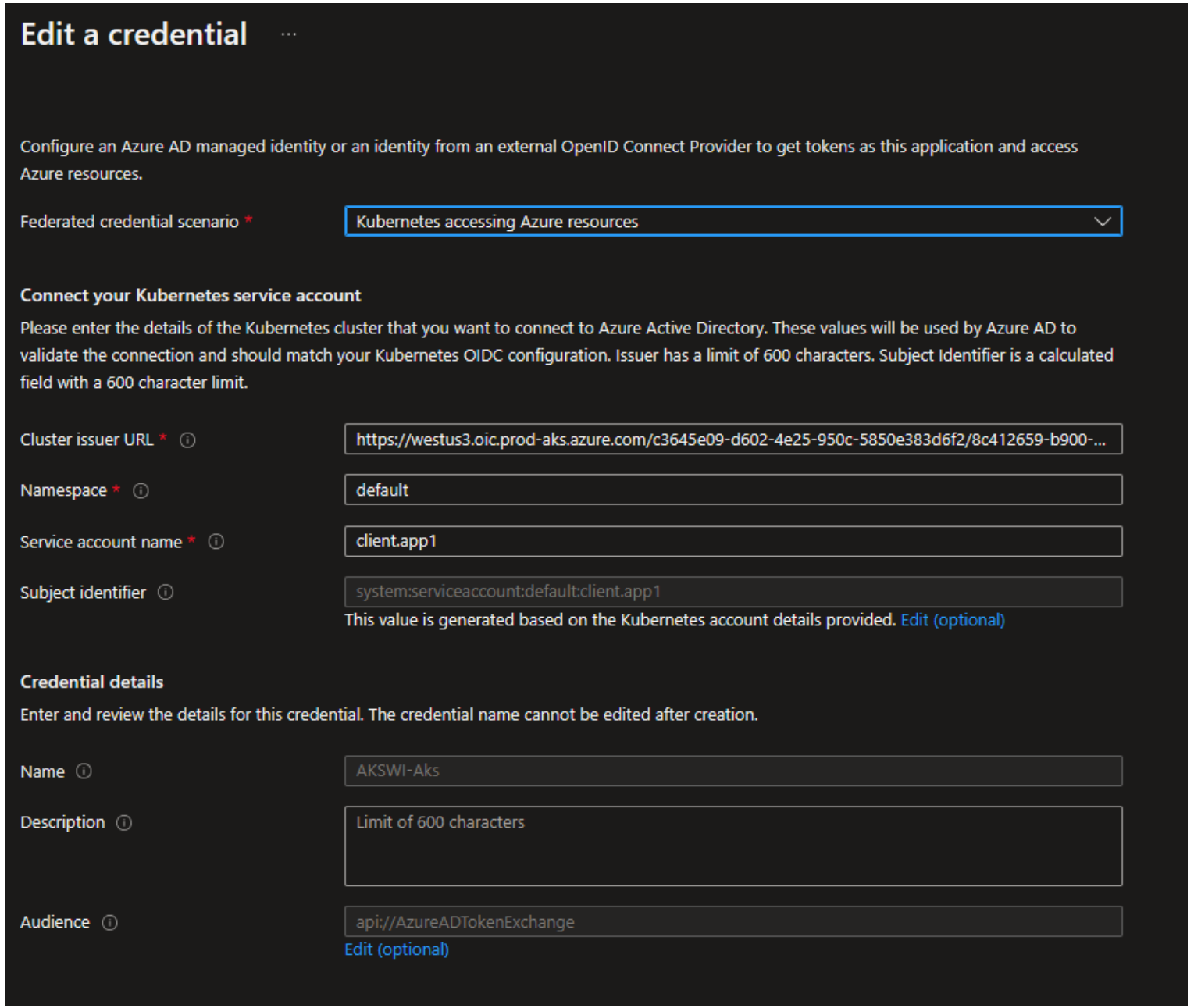
Associate a service account to pod

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: client.app1
---
apiVersion: apps/v1
kind: Deployment
metadata:
  …
spec:
    …
    spec:
      containers:
        - name: api-client
 …
 volumeMounts:
 - name: token-vol
   mountPath: "/service-account"
   readOnly: true
      serviceAccountName: client.app1
      volumes:
      - name: token-vol
        projected:
 sources:
 - serviceAccountToken:
     audience: "api://AzureADTokenExchange"
     expirationSeconds: 3600
     path: token
```

Provide token to app code

Configure token

AAD


Edit a credential
Configure an Azure AD managed identity or an identity from an external OpenID Connect Provider to get tokens as this application and access Azure resources.
Federated credential scenario *
Kubernetes accessing Azure resources
Connect your Kubernetes service account
Please enter the details of the Kubernetes cluster that you want to connect to Azure Active Directory. These values will be used by Azure AD to validate the connection and should match your Kubernetes OIDC configuration. Issuer has a limit of 600 characters. Subject Identifier is a calculated field with a 600 character limit.
Cluster issuer URL *
https://westus3.oic.prod-aks.azure.com/c3645e09-d602-4e25-950c-5850e383d6f2/8c412659-b900-...
Namespace *
default
Service account name *
client.app1
Subject identifier
system:serviceaccount:default:client.app1
This value is generated based on the Kubernetes account details provided. Edit (optional)
Credential details
Enter and review the details for this credential. The credential name cannot be edited after creation.
Name
AKSWI-Aks
Description
Limit of 600 characters
Audience
api://AzureADTokenExchange
Edit (optional)

Application code

```
string k8sToken = String.Empty;
try
{
   using (var sr = new StreamReader("/service-account/token"))
   {
      k8sToken = await sr.ReadToEndAsync();
   }
} catch(Exception ex)
{
   _logger.LogError(ex.Message);
}
if(!string.IsNullOrEmpty(k8sToken))
{
   var msal = ConfidentialClientApplicationBuilder
     .Create(_clientId)
     .WithAuthority($"https://login.microsoftonline.com/{_tenantId}/")
     .WithClientAssertion(k8sToken)
     .Build();
   try
   {
      var tokens = await msal.AcquireTokenForClient(new string[] { api://xyz/.default" }).ExecuteAsync();
      // Call your API with tokens.AccessToken in Authorization header
   }
   catch (MsalServiceException ex)
   {
      _logger.LogError(ex.Message);
   }
}
```

Demo

- AKS cluster create script
- Deployment yaml
- AAD configuration
- Code
- Executing web app

Benefits

- Native K8s APIs, any cloud, both Windows and Linux
- Access control remains in AAD
- [Protection of secrets - secrets never leave their source]
- OAuth2 standard

Resources

- Kubernetes and AAD Workload Identity
- AAD Workload Identity (Preview)
- Setup
- Sample
- My sample
- Deployment with self-managed clusters