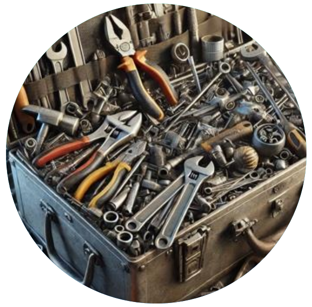
Passwordless App Infrastructures

Utilizing Azure Managed Identities and Identity Federation



Pasi Huuhka
DevOps Architect @ Zure

ZURE

- We solve business problems with technology
- Designing, building and managing on Azure since 2011
- 108 employees, with an average of 16 years of experience
- Offices in Finland, Belgium, Denmark and UK
- Fully independent, owned by employees

Microsoft Solutions Partner
Digital & App Innovation
Azure
Specialist
AI and Machine Learning
Migrate Enterprise Applications to Microsoft Azure

Microsoft Solutions Partner
Data & AI
Azure
Specialist
AI and Machine Learning
Migrate Enterprise Applications to Microsoft Azure

Microsoft Solutions Partner
Infrastructure
Azure
Specialist
Azure Virtual Desktop

Microsoft Solutions Partner
Security

MVP Microsoft® Most Valuable Professional

Microsoft Partner
Microsoft
2023 Partner of the Year Winner
Application Innovation and Modernization
Finland

FAUG - "Low code vs. Pro code" panel (LIVE) at Microsoft house

Hosted By
Sakari N. and 4 others

Finland Azure User Group
Public group ?

Tuesday, March 4, 2025
12:30 PM to 4:00 PM EET
Add to calendar

Microsoft Oy
Keilalahdentie 2-4 · Espoo

- Low code vs. pro code - where's the limit?
- Technology stack, Power Platform, Azure Logic Apps, AI Studio..
- Is Citizen Developer a lie?
- How about the governance?
- How does DevOps sit in to the picture? Or does it?
- Fusion development!

ZURE

In this presentation…

- We'll go through multiple examples of implementing aspects of an application without passwords from deployment to runtime
- We'll dive into the code level both for infra and application to see how it's done… and what caveats I've run into
- We'll see how to implement passwordless auth to Azure services running anywhere, even on your local machine
- I'm happy if everyone takes one of these to implement in their own projects. It's worth it!

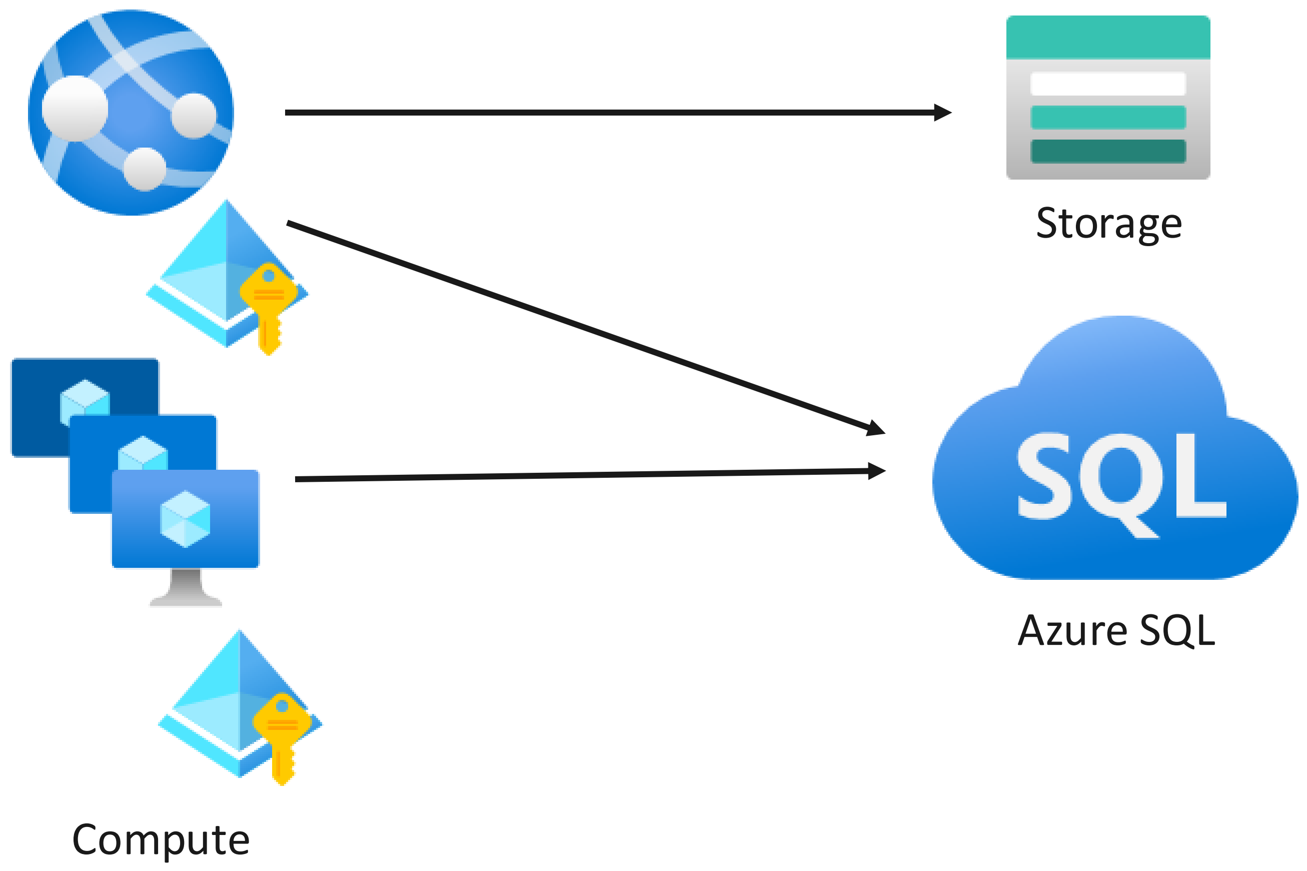
ZURE

Why am I talking about this?

- I’m lazy. Sometimes that equals security!
- Renewing secrets sucks
  - It’s risky
  - No-one has documented the whole process
  - Automation is expensive to create, sometimes impossible
  - Someone needs to have huge privileges to handle renewals
- The solutions of this talk are not magic, even if they feel like it. I wanted to understand the mechanics behind them better.

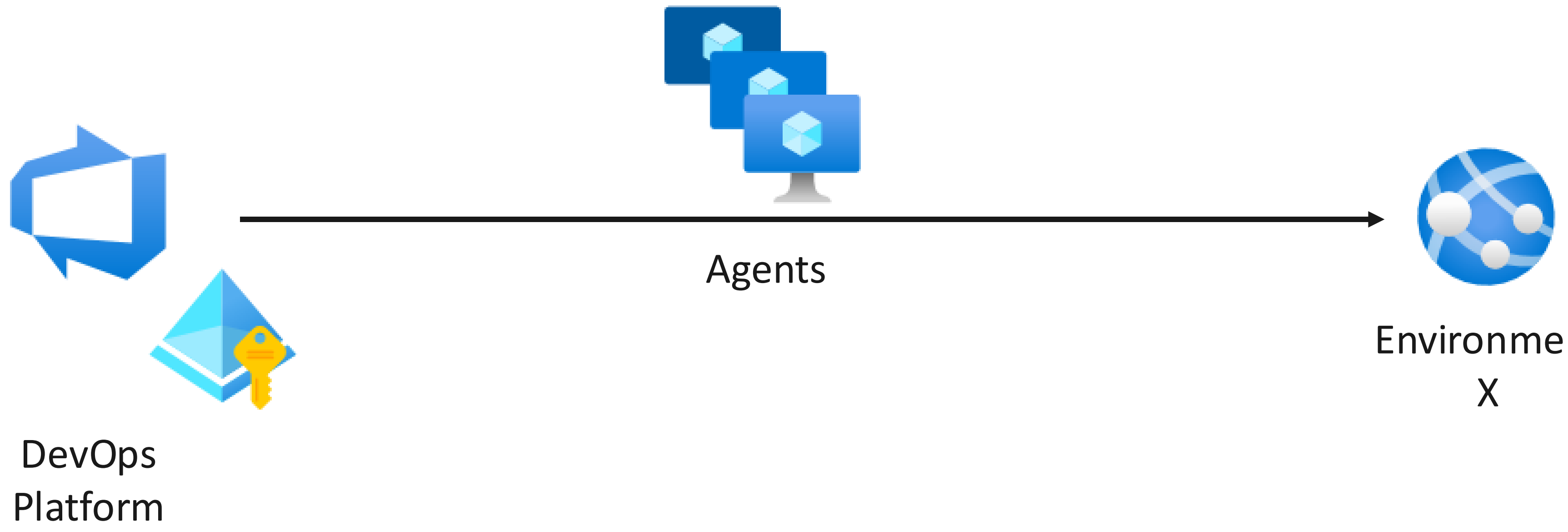
Setting the scope…

Application internal communication (with your code)

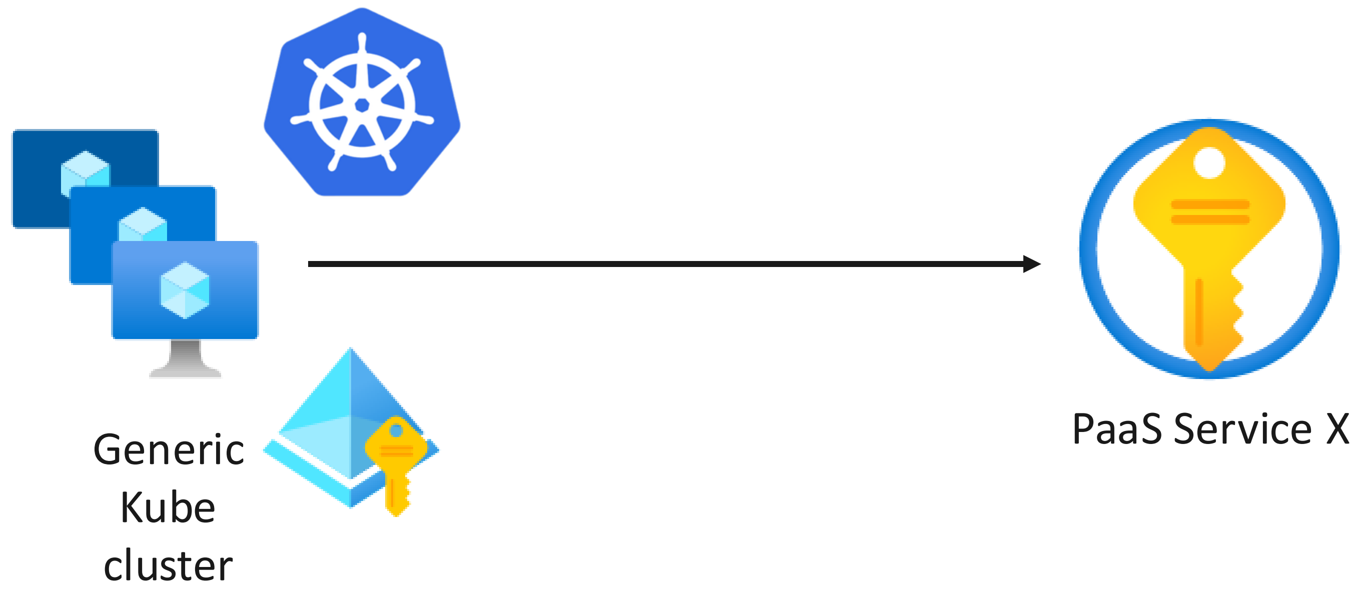


ZURE

Deployments



Kubernetes from anywhere



But first…

What is a managed identity?

- Managed resource in Azure, no password even exists
- Permissions given through Azure RBAC (or Entra ID permissions)
- System Assigned / User Assigned
- Supported by most Azure Services
  - Use Key Vault as an intermediary if not
- In a nutshell:
  - Create Identity / Resource
  - Assign identity to resource
  - Resource can fetch a 24h token for that identity

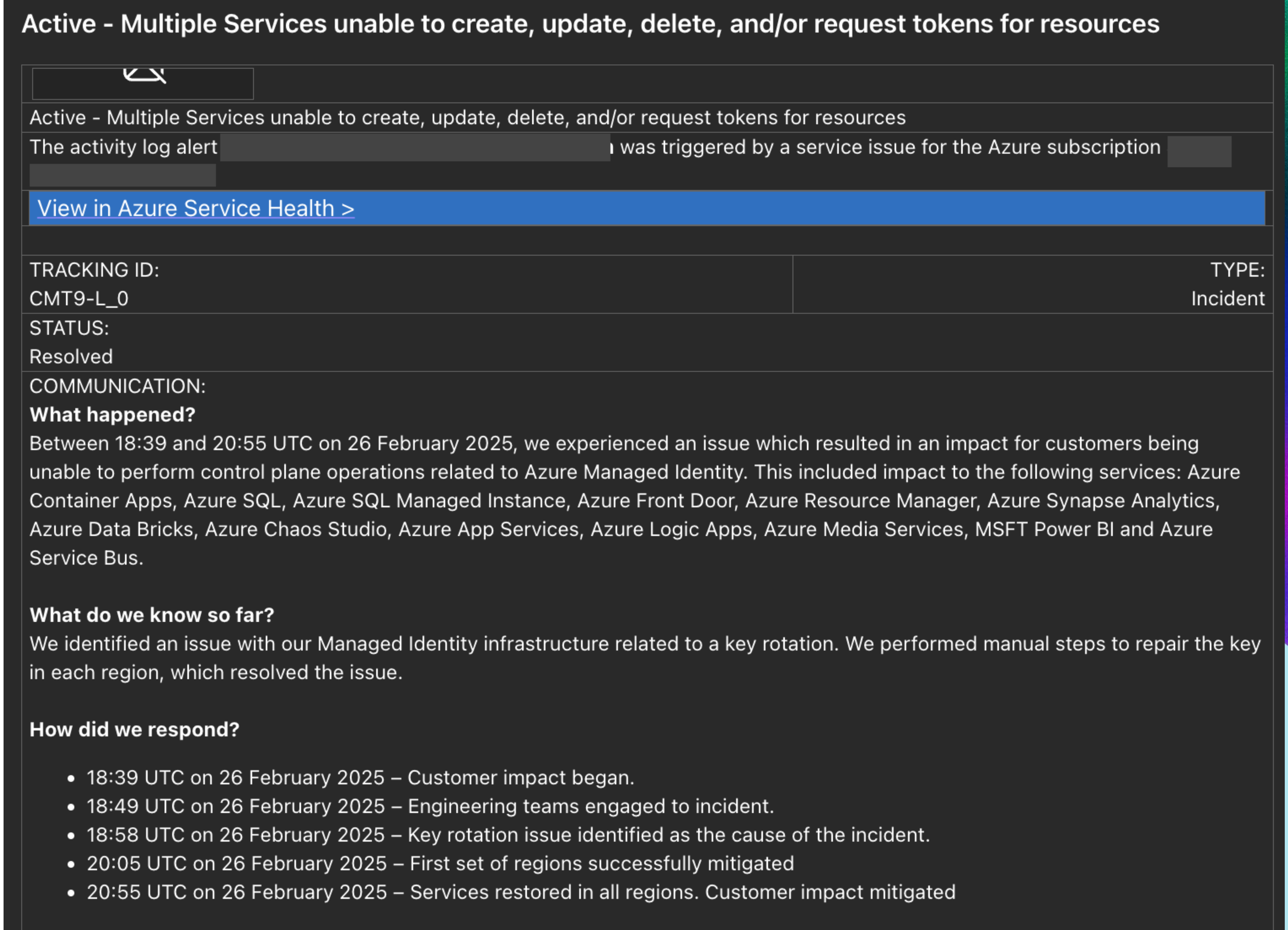
Fetching Tokens

- Resources have a local token endpoint that abstracts the call to Entra ID
  - Addresses differ by resource
- You might need to provide an additional header found in the env variables of your runtime
  - Underlying Azure platform also verifies assignment
- Usually, however, all this is abstracted away in your code

DEMO!

Best Practices - Caching

- Microsoft developed SDKs provide in-memory caching for tokens
- Turned on by default
- Persistent caching for ManagedIdentity tokens is not supported
- Longer 24h expiration time can be a bane or a boon
- Should let MSAL handle token renewal



Active - Multiple Services unable to create, update, delete, and/or request tokens for resources

Active - Multiple Services unable to create, update, delete, and/or request tokens for resources

The activity log alert [redacted] was triggered by a service issue for the Azure subscription [redacted]

View in Azure Service Health >

TRACKING ID:	TYPE:
CMT9-L_0	Incident

STATUS:
Resolved

COMMUNICATION:

What happened?

Between 18:39 and 20:55 UTC on 26 February 2025, we experienced an issue which resulted in an impact for customers being unable to perform control plane operations related to Azure Managed Identity. This included impact to the following services: Azure Container Apps, Azure SQL, Azure SQL Managed Instance, Azure Front Door, Azure Resource Manager, Azure Synapse Analytics, Azure Data Bricks, Azure Chaos Studio, Azure App Services, Azure Logic Apps, Azure Media Services, MSFT Power BI and Azure Service Bus.

What do we know so far?

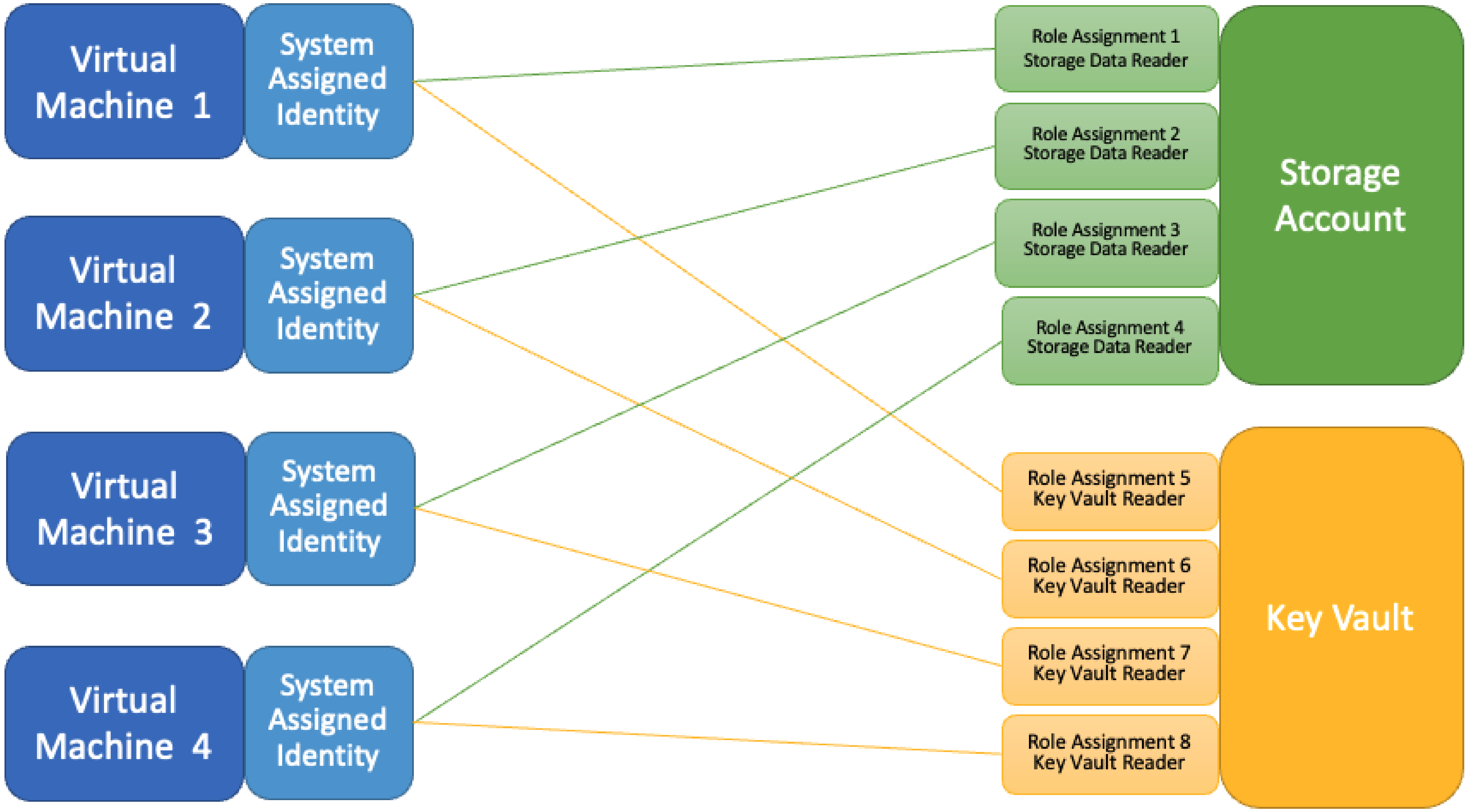
We identified an issue with our Managed Identity infrastructure related to a key rotation. We performed manual steps to repair the key in each region, which resolved the issue.

How did we respond?

- 18:39 UTC on 26 February 2025 – Customer impact began.
- 18:49 UTC on 26 February 2025 – Engineering teams engaged to incident.
- 18:58 UTC on 26 February 2025 – Key rotation issue identified as the cause of the incident.
- 20:05 UTC on 26 February 2025 – First set of regions successfully mitigated
- 20:55 UTC on 26 February 2025 – Services restored in all regions. Customer impact mitigated

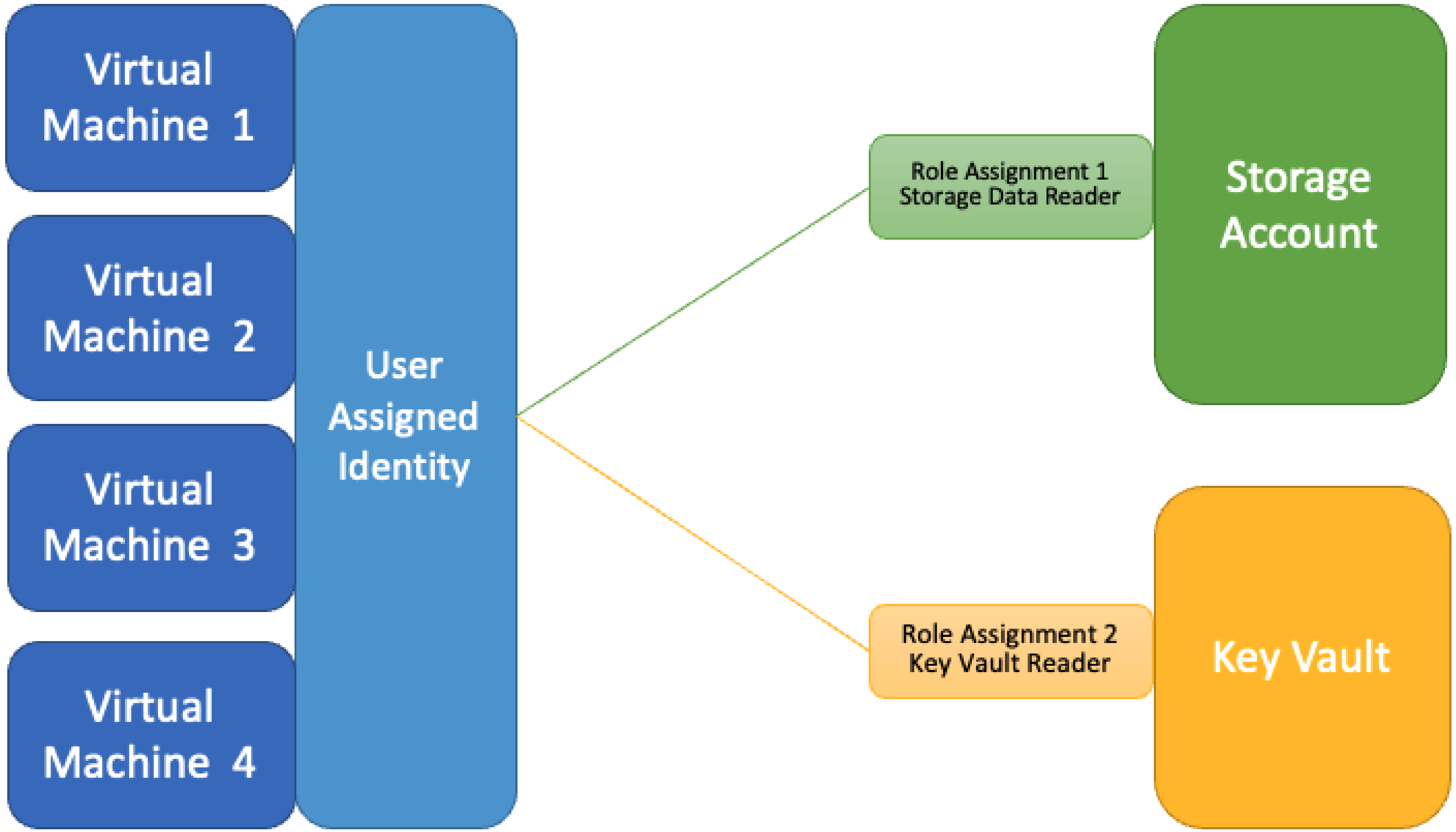
Best Practices - Patterns

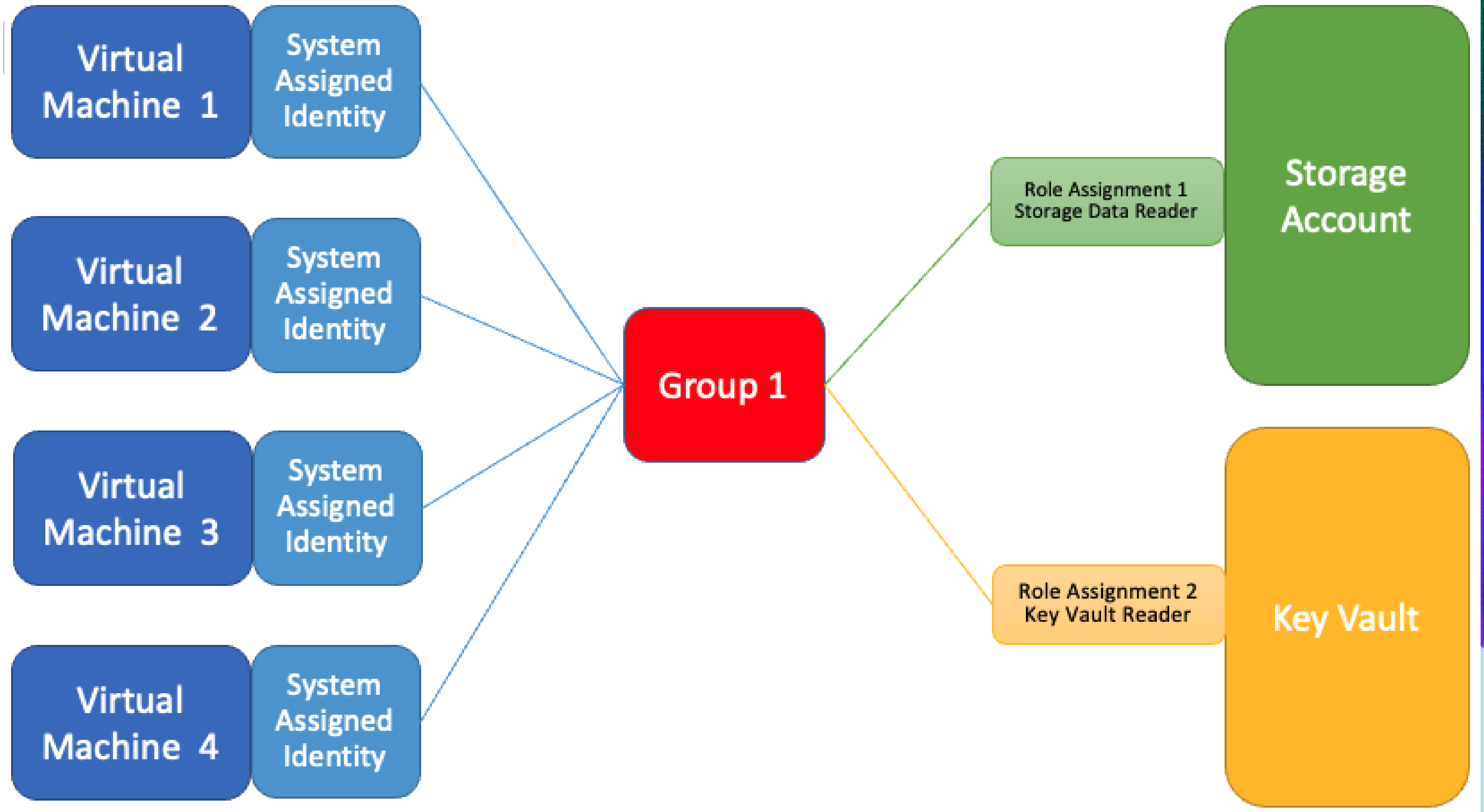
- Lifecycle – User Assigned does not die along with the compute resource
- More shared identities = less management, harder to audit
- Find the golden middle path

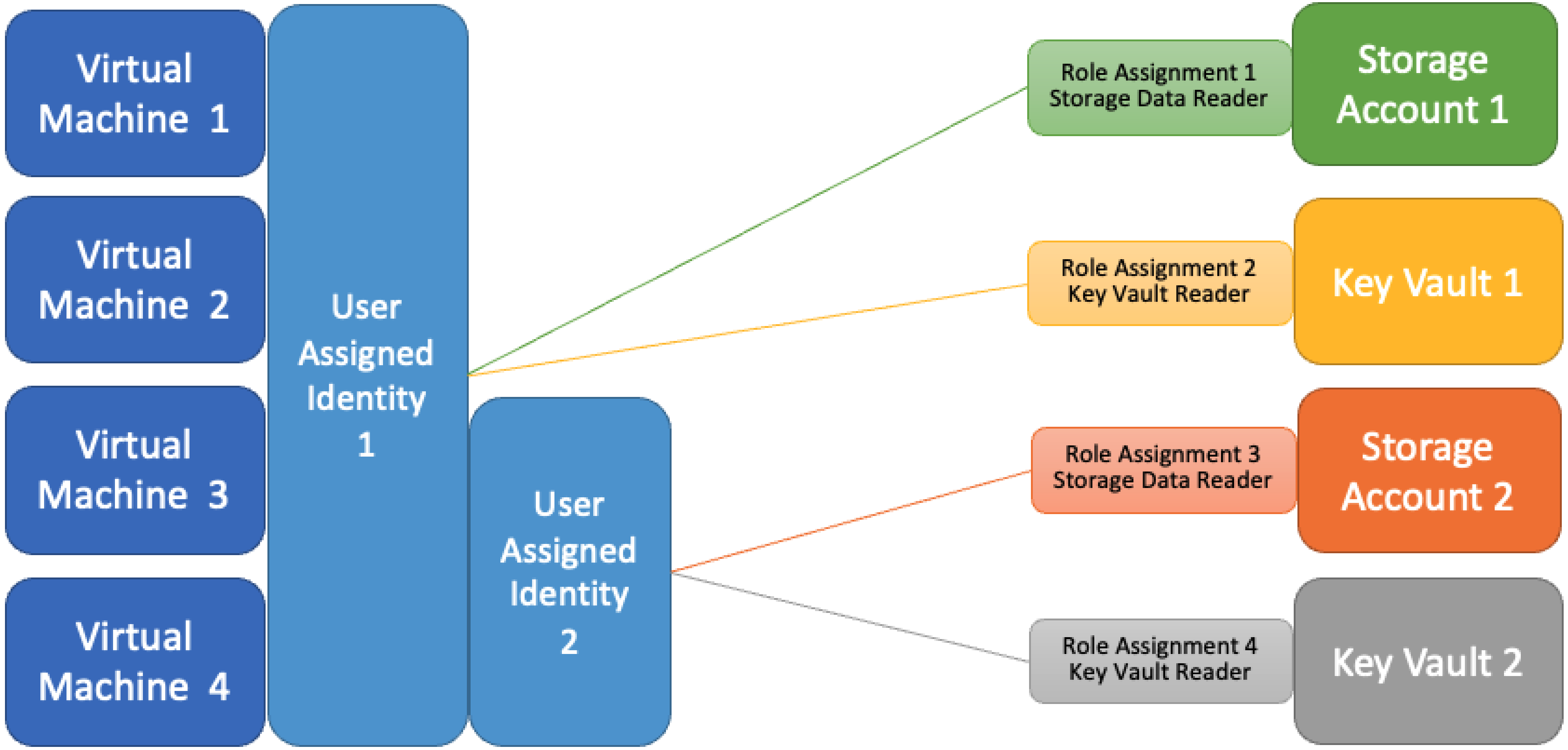


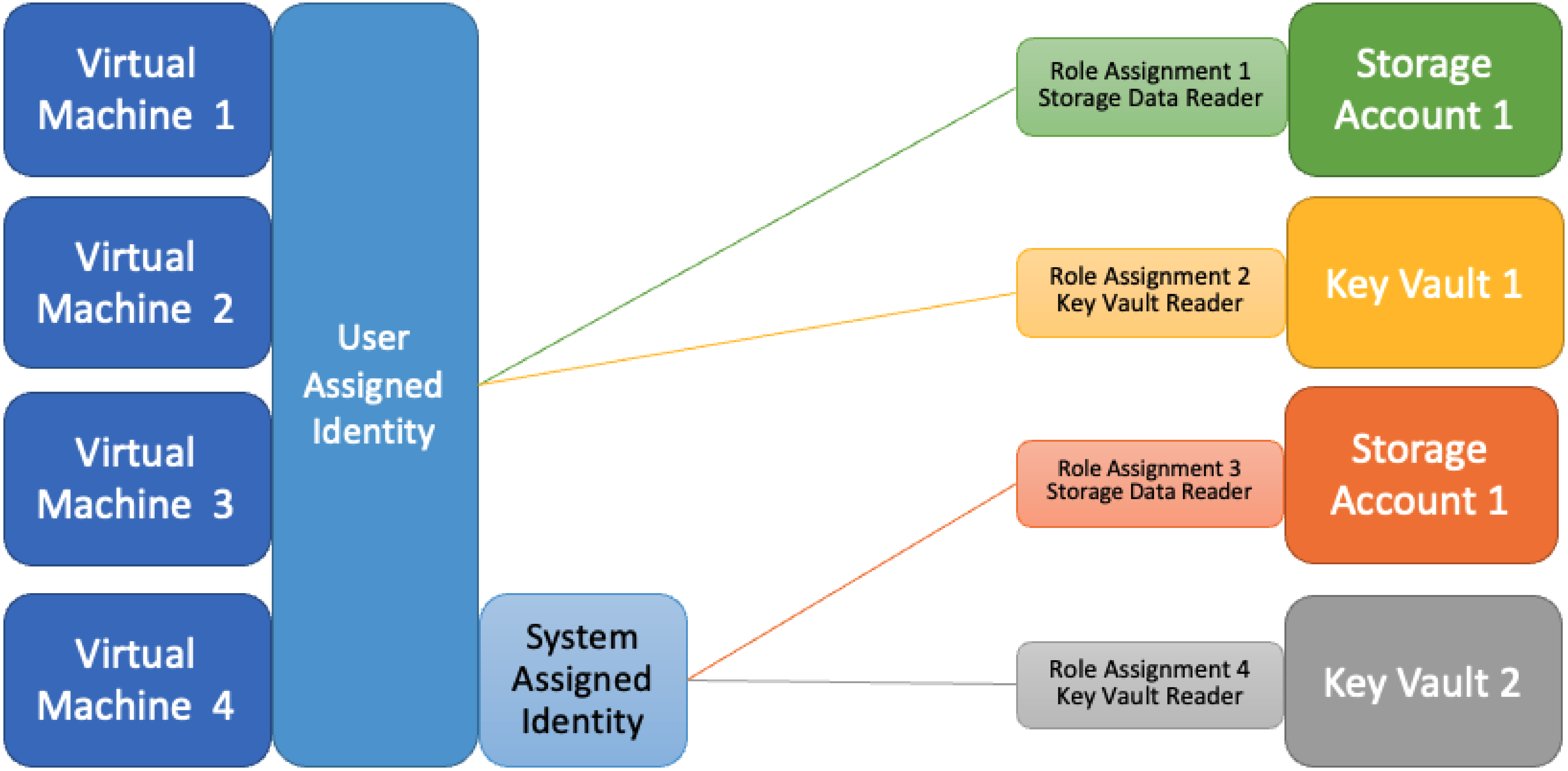
Virtual Machine 1
System Assigned Identity
Virtual Machine 2
System Assigned Identity
Virtual Machine 3
System Assigned Identity
Virtual Machine 4
System Assigned Identity
Role Assignment 1 Storage Data Reader
Role Assignment 2 Storage Data Reader
Role Assignment 3 Storage Data Reader
Role Assignment 4 Storage Data Reader
Storage Account
Role Assignment 5 Key Vault Reader
Role Assignment 6 Key Vault Reader
Role Assignment 7 Key Vault Reader
Role Assignment 8 Key Vault Reader
Key Vault

ZURE


Virtual Machine 1
Virtual Machine 2
Virtual Machine 3
Virtual Machine 4
User Assigned Identity
Role Assignment 1 Storage Data Reader
Storage Account
Role Assignment 2 Key Vault Reader
Key Vault


Virtual Machine 1
System Assigned Identity
Virtual Machine 2
System Assigned Identity
Virtual Machine 3
System Assigned Identity
Virtual Machine 4
System Assigned Identity
Group 1
Role Assignment 1 Storage Data Reader
Storage Account
Role Assignment 2 Key Vault Reader
Key Vault


Virtual Machine 1
Virtual Machine 2
Virtual Machine 3
Virtual Machine 4
User Assigned Identity 1
User Assigned Identity 2
Role Assignment 1 Storage Data Reader
Role Assignment 2 Key Vault Reader
Role Assignment 3 Storage Data Reader
Role Assignment 4 Key Vault Reader
Storage Account 1
Key Vault 1
Storage Account 2
Key Vault 2


Virtual Machine 1
Virtual Machine 2
Virtual Machine 3
Virtual Machine 4
User Assigned Identity
System Assigned Identity
Role Assignment 1 Storage Data Reader
Role Assignment 2 Key Vault Reader
Role Assignment 3 Storage Data Reader
Role Assignment 4 Key Vault Reader
Storage Account 1
Key Vault 1
Storage Account 1
Key Vault 2

Best Practices - Roles

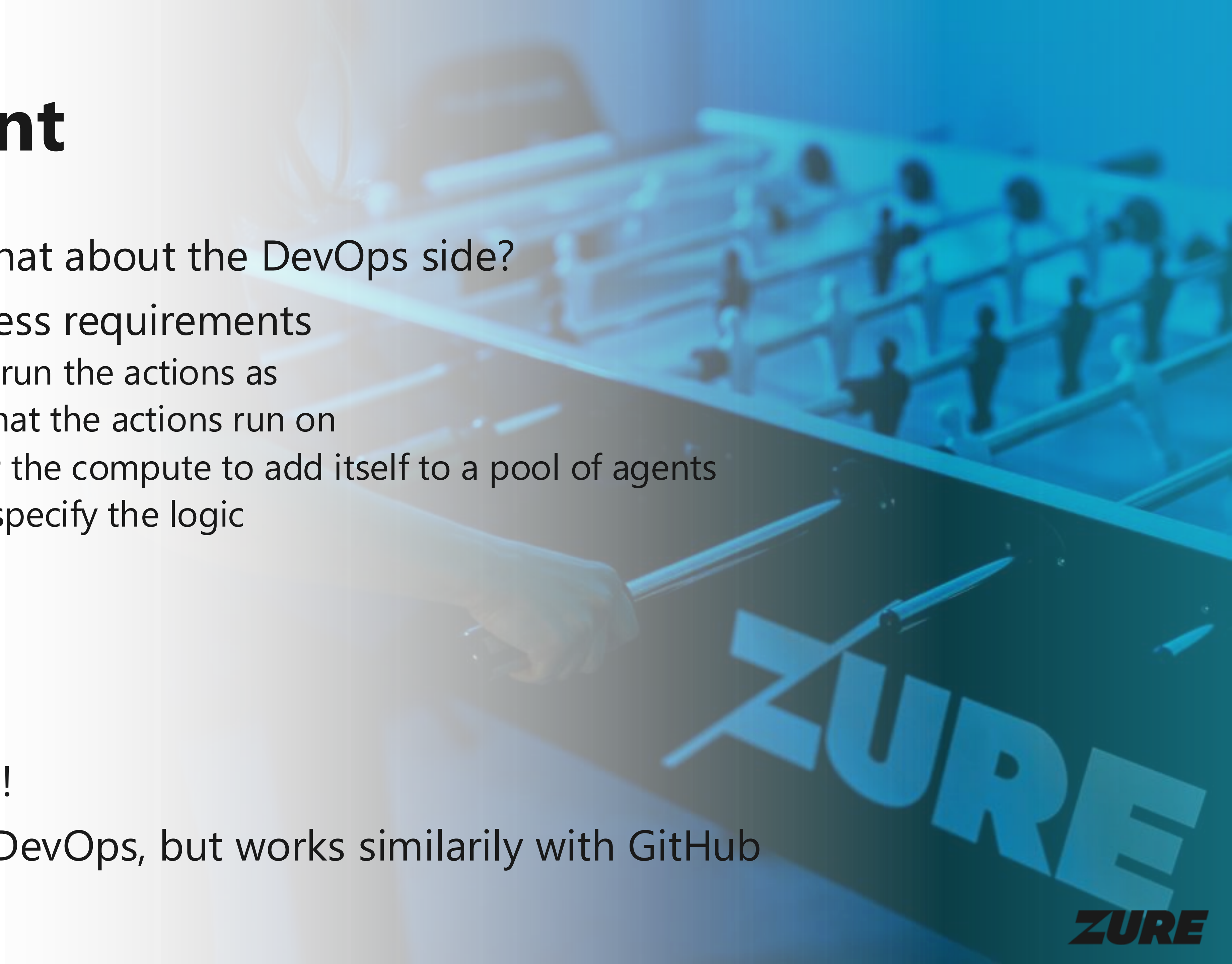
- **Managed Identity Contributor** role can create and delete identities
- **Managed Identity Operator** role can assign an identity to a resource
- Permission to **assign ==** permission to **assume**
- To give role assignments, you need **Owner** or **User Access Administrator**

DEMO!

Deployments?

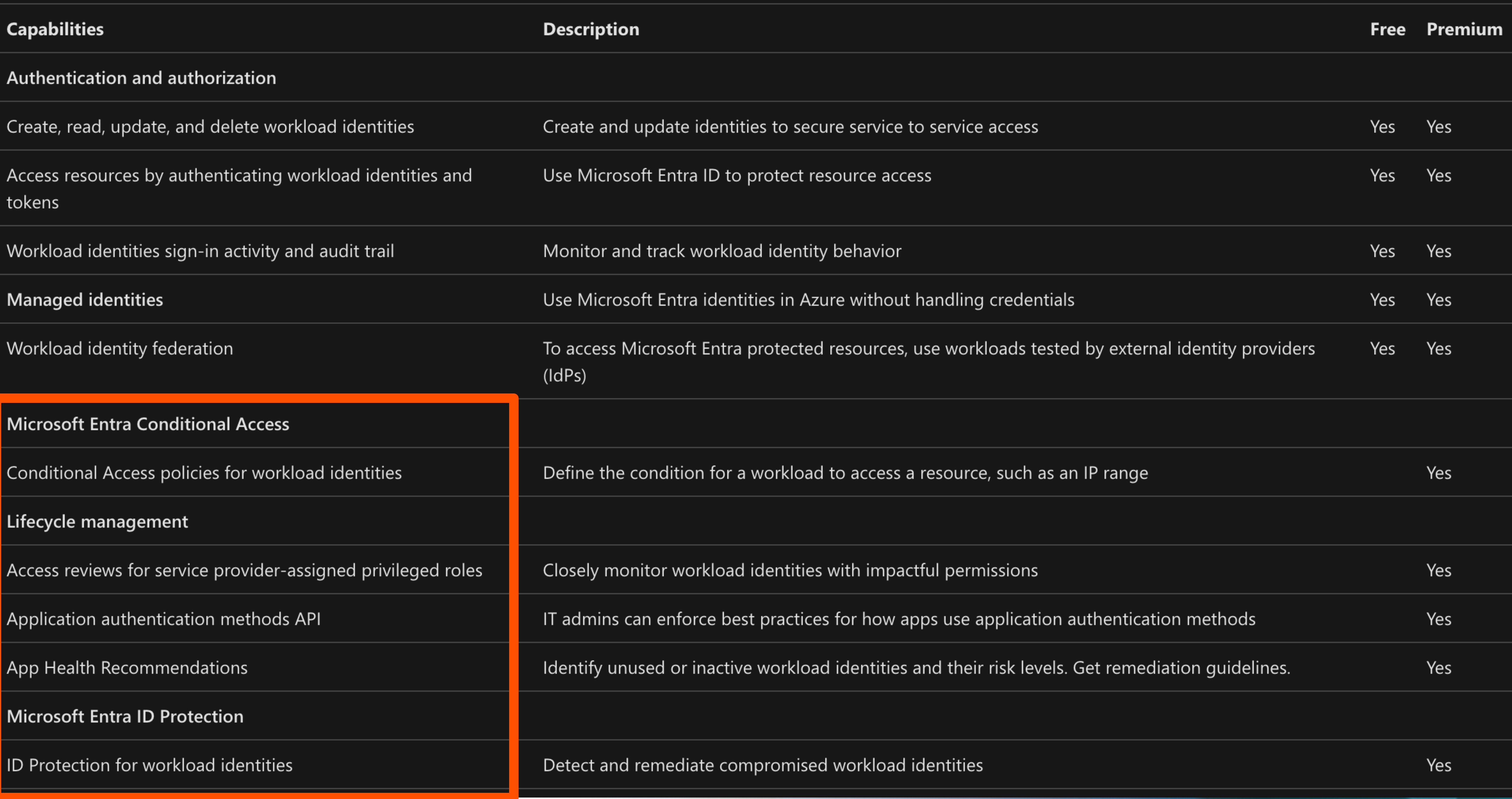
Deployment

- App works, but what about the DevOps side?
- Deployment process requirements
  - Some Identity to run the actions as
  - Some compute that the actions run on
  - Some identity for the compute to add itself to a pool of agents
  - Some service to specify the logic
- Options:
  - FTP?
  - Basic Auth?
  - Service Principal?
- Workload identity!
- Demos on Azure DevOps, but works similarily with GitHub



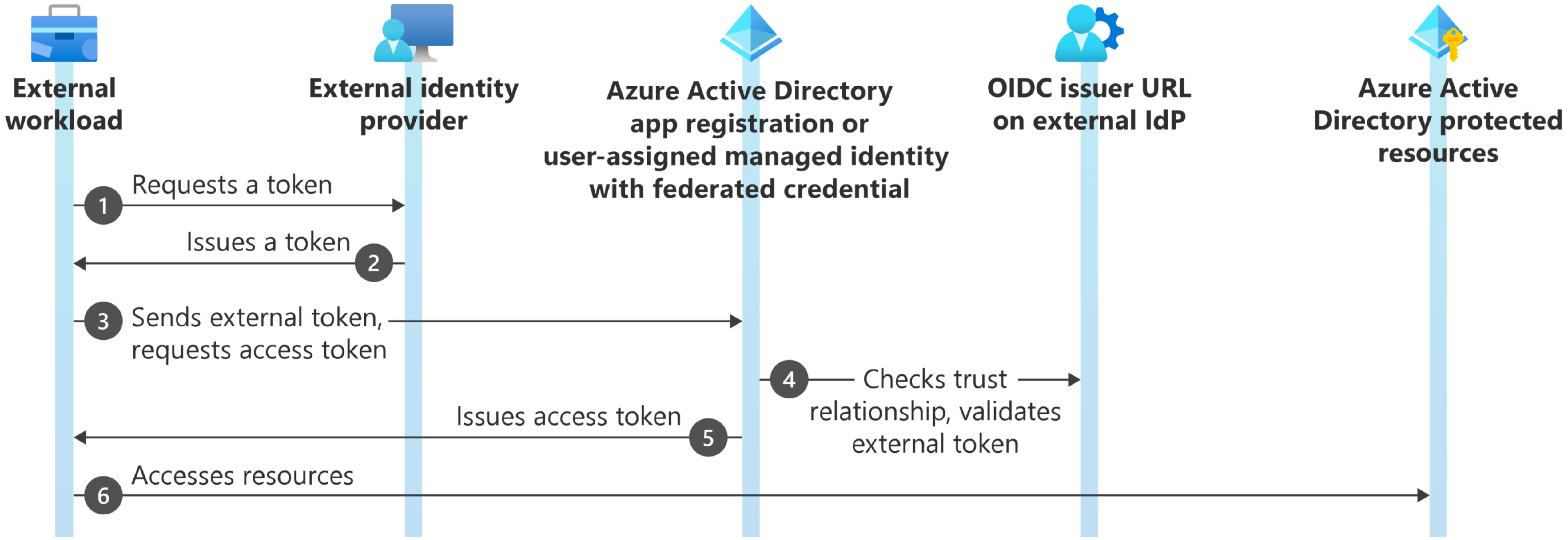
Workload Identity Federation?

- Mechanism to delegate authentication to an external identity provider
- Supported in all the major cloud providers
  - and platforms like Kubernetes, Github, Azure DevOps…
- Available in AKS for… maybe 1,5 years now?
- 2 SKUs, Free and Premium (3$/month per identity)
  - Free 90-day trial also available
  - Bought through Entra Admin console, somewhere…



Capabilities	Description	Free	Premium
Authentication and authorization			
Create, read, update, and delete workload identities	Create and update identities to secure service to service access	Yes	Yes
Access resources by authenticating workload identities and tokens	Use Microsoft Entra ID to protect resource access	Yes	Yes
Workload identities sign-in activity and audit trail	Monitor and track workload identity behavior	Yes	Yes
Managed identities	Use Microsoft Entra identities in Azure without handling credentials	Yes	Yes
Workload identity federation	To access Microsoft Entra protected resources, use workloads tested by external identity providers (IdPs)	Yes	Yes
Microsoft Entra Conditional Access			
Conditional Access policies for workload identities	Define the condition for a workload to access a resource, such as an IP range		Yes
Lifecycle management			
Access reviews for service provider-assigned privileged roles	Closely monitor workload identities with impactful permissions		Yes
Application authentication methods API	IT admins can enforce best practices for how apps use application authentication methods		Yes
App Health Recommendations	Identify unused or inactive workload identities and their risk levels. Get remediation guidelines.		Yes
Microsoft Entra ID Protection			
ID Protection for workload identities	Detect and remediate compromised workload identities		Yes

Workload Identity Federation?


External workload
External identity provider
Azure Active Directory app registration or user-assigned managed identity with federated credential
OIDC issuer URL on external IdP
Azure Active Directory protected resources
1 Requests a token
2 Issues a token
3 Sends external token, requests access token
4 Checks trust relationship, validates external token
5 Issues access token
6 Accesses resources

Critical configuration parameters

- Issuer
  - URL of the external identity provider, e.g., Github
  - Must match the **issuer** claim of the external token
  - Leading or trailing spaces cause issues
- Subject
  - Identifier of the external software workload
  - Must match the **sub** claim of the external token
  - No fixed format, depends on the external provider
  - Kube example: "system:serviceaccount:NAMESPACE:SERVICEACCOUNTNAME"
- Audiences
  - Appear in the external token. Must add at least one
  - Recommended value "**api://AzureADTokenExchange**", but can be anything
  - Determines what MS Identity platform must accept In the **aud** claim in the incoming token

These are not secrets!

Deployment – Identity

- Can be created manually or fully automated
- Can be backed by app registration or a managed identity
  - I like using managed identities – No access to Entra ID portal needed
- Some Azure DevOps tasks might not yet support this?
  - But I can’t think of any that don’t

DEMO!

Deployment - Agents

- MSFT-hosted Agents
- Self-hosted Agents
- Managed DevOps pools

Managed DevOps Pools

- Fully Managed
- Configurable with built-in or custom images
- Configurable VM sizes, scaling
- Joinable to your private networks
- Easy to create, no more noisy neighbor
- Resources live in MSFT subscriptions
- **GA since Ignite!**

ZURE

DEMO!

Self hosted…

- Azure DevOps APIs now support bearer tokens wherever Personal Access Tokens (PAT) are accepted
- We can utilize this with managed identity
- Identity needs permissions in DevOps
    - Basic License
    - Reader permissions to project
    - Admin permissions to pool

ZURE

DEMO!

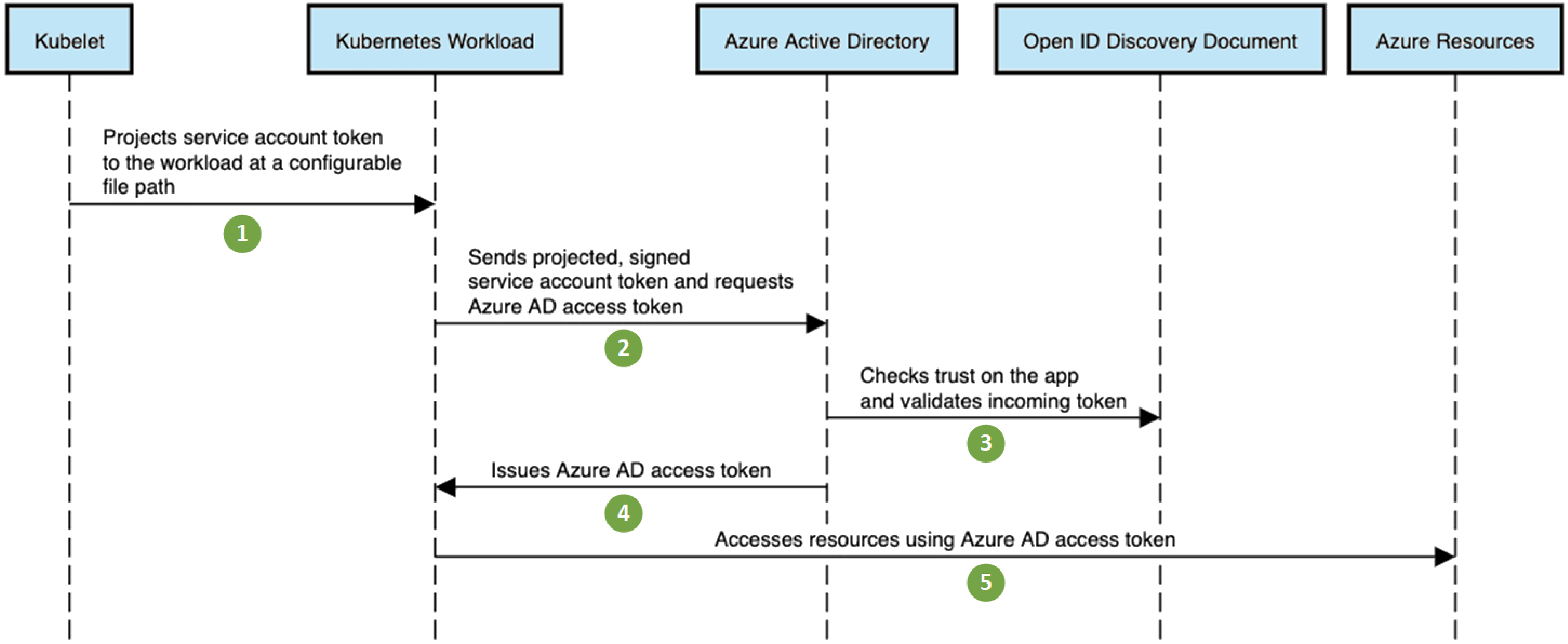
Workload identity from anywhere?

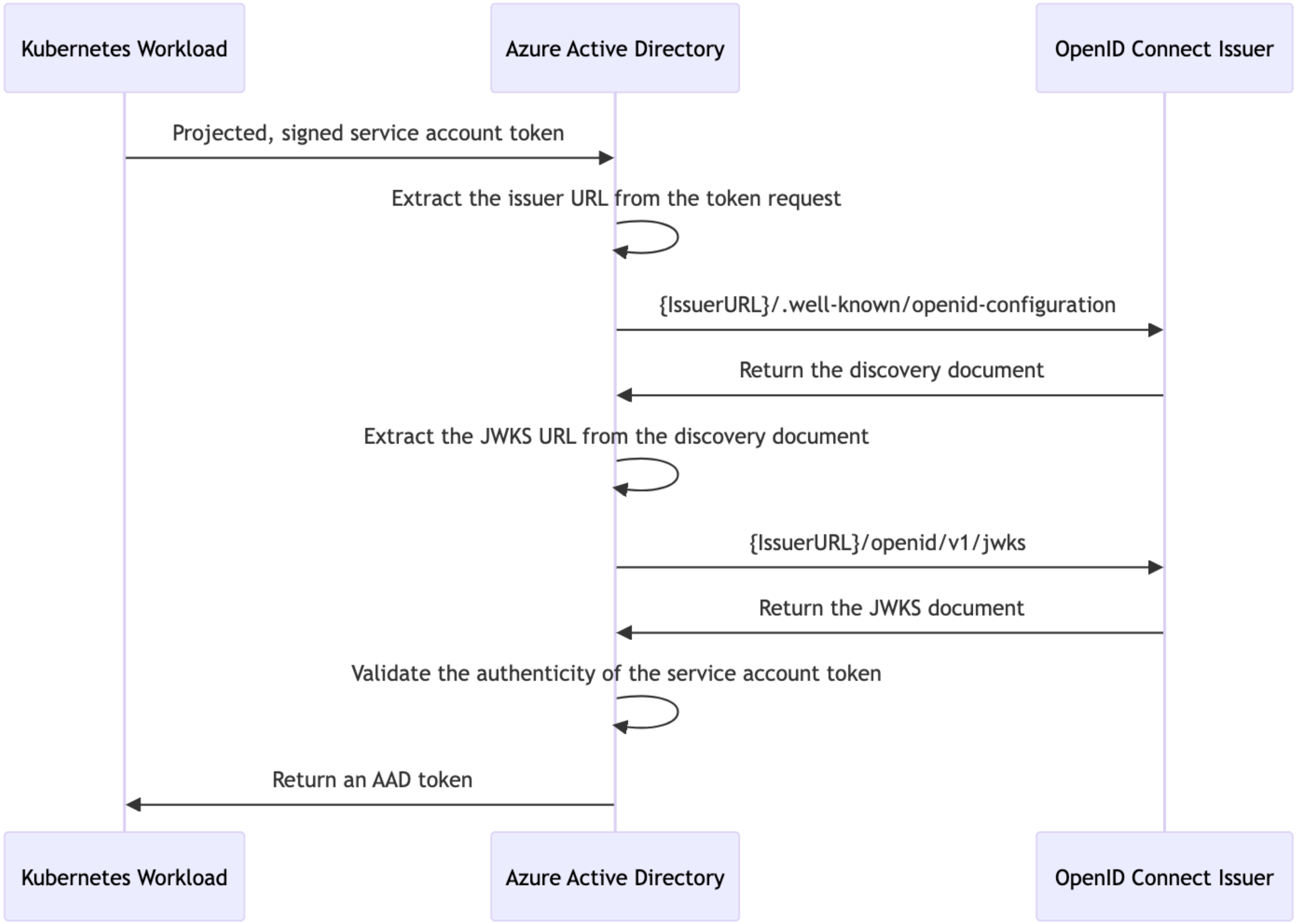
ZURE

Enter Kubernetes

- Allows us to set up our own external identity provider
  - Our workload can run anywhere
- Requires public hosting of…
  - OpenID Discovery Document
  - JWKS document (JSON Web Key Sets)
- Tools:
  - azwi cli / bicep graph extension
  - azwi mutating admission webhook

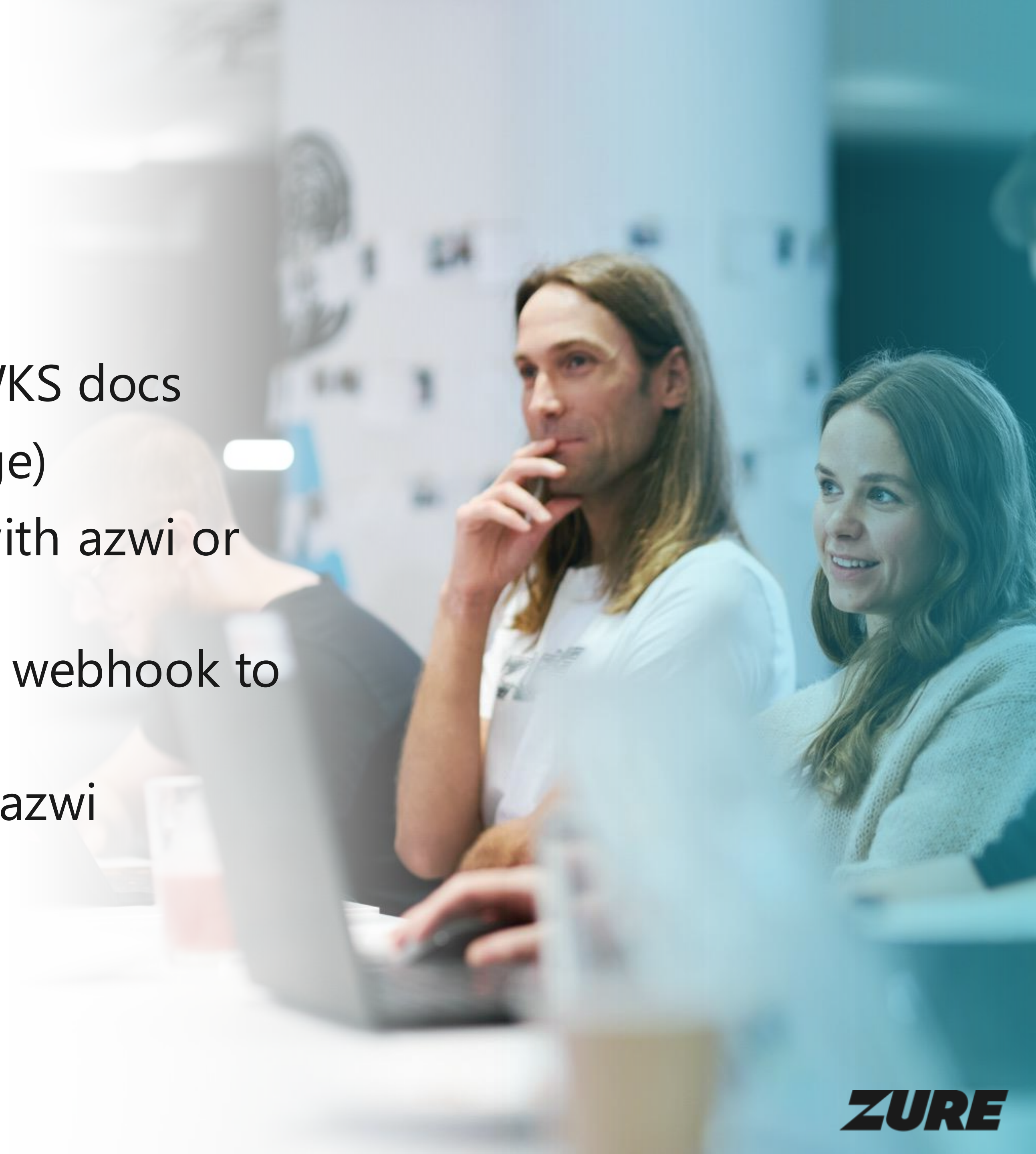
Enter Kubernetes


Kubelet
Kubernetes Workload
Azure Active Directory
Open ID Discovery Document
Azure Resources
Projects service account token to the workload at a configurable file path
1
Sends projected, signed service account token and requests Azure AD access token
2
Checks trust on the app and validates incoming token
3
Issues Azure AD access token
4
Accesses resources using Azure AD access token
5


Kubernetes Workload
Azure Active Directory
OpenID Connect Issuer
Projected, signed service account token
Extract the issuer URL from the token request
{IssuerURL}/.well-known/openid-configuration
Return the discovery document
Extract the JWKS URL from the discovery document
{IssuerURL}/openid/v1/jwks
Return the JWKS document
Validate the authenticity of the service account token
Return an AAD token
Kubernetes Workload
Azure Active Directory
OpenID Connect Issuer

Steps

1. Generate RSA keys for kube cluster
2. Generate OpenID Discovery and JWKS docs
3. Host docs publicly (on Azure Storage)
4. Create a Entra ID app registration with azwi or bicep
5. Configure azwi mutating admission webhook to inject required values to pods
6. Create a kube service account with azwi
7. Deploy workload



Generate RSA Keys

```
echo "Generating RSA keys..."
openssl genrsa -out /home/$USER/sa.key 2048
openssl rsa -in /home/$USER/sa.key -pubout -out /home/$USER/sa.pub
```

Generate Docs

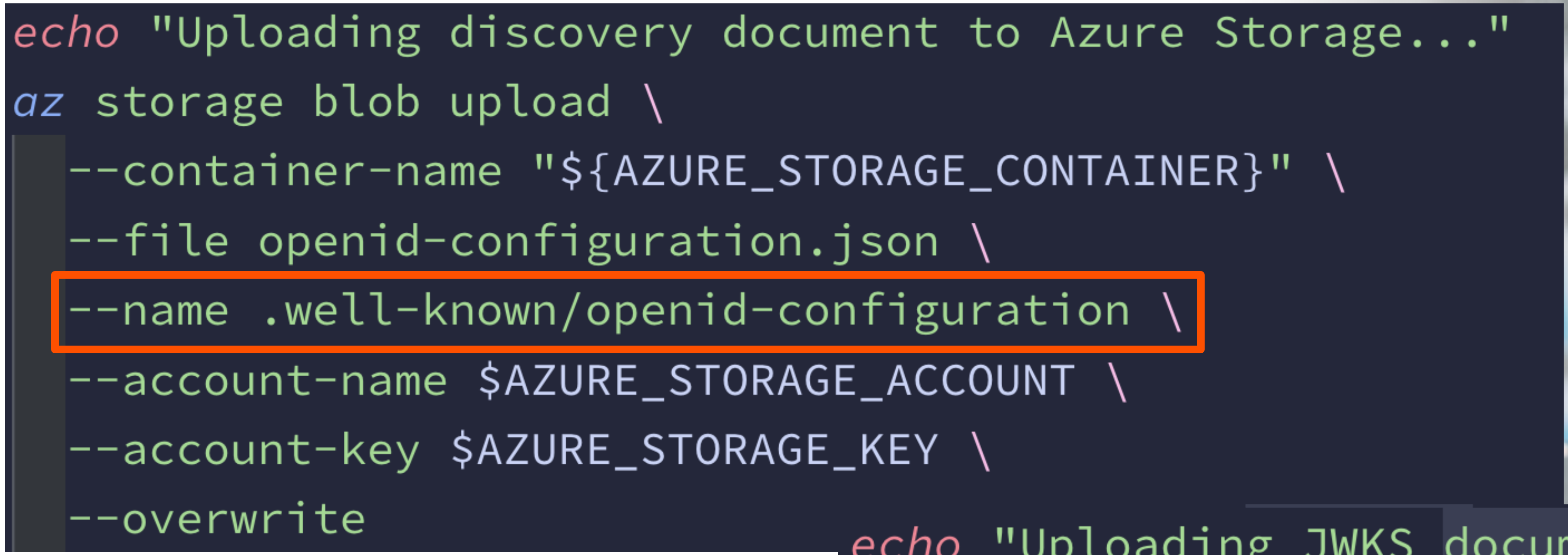
```
echo "Generating OpenID Connect discovery document..."
cat <<EOF > openid-configuration.json
{
  "issuer": "https://${AZURE_STORAGE_ACCOUNT}.blob.core.windows.net/${AZURE_STORAGE_CONTAINER}/",
  "jwks_uri": "https://${AZURE_STORAGE_ACCOUNT}.blob.core.windows.net/${AZURE_STORAGE_CONTAINER}/openid/v1/jwks",
  "response_types_supported": [
    "id_token"
  ],
  "subject_types_supported": [
    "public"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ]
}
EOF
```

Generate Docs

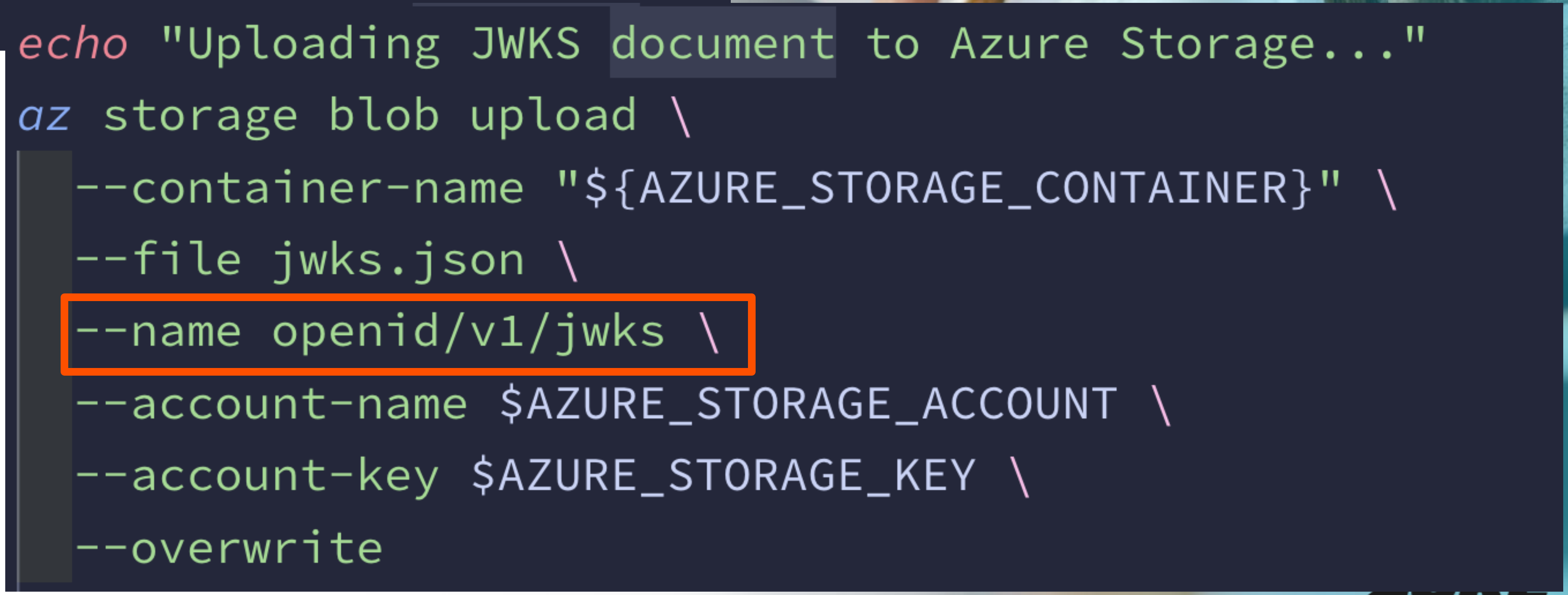
```
echo "Generating JWKS document..."
azwi jwks --public-keys /home/$USER/sa.pub --output-file jwks.json
```

```
{
  "keys": [
    {
      "use": "sig",
      "kty": "RSA",
      "kid": "0iMZTwqb7wk6NjkgLo8P-2iTufRpH725fjdMCXgsvAY",
      "alg": "RS256",
      "n": "udACDLOExJXuJbGqkB0Ye7-NjeKfcnjr...",
      "e": "AQAB"
    }
  ]
}
```

Host Docs



```
echo "Uploading discovery document to Azure Storage..."
az storage blob upload \
  --container-name "${AZURE_STORAGE_CONTAINER}" \
  --file openid-configuration.json \
  --name .well-known/openid-configuration \
  --account-name $AZURE_STORAGE_ACCOUNT \
  --account-key $AZURE_STORAGE_KEY \
  --overwrite
```



```
echo "Uploading JWKS document to Azure Storage..."
az storage blob upload \
  --container-name "${AZURE_STORAGE_CONTAINER}" \
  --file jwks.json \
  --name openid/v1/jwks \
  --account-name $AZURE_STORAGE_ACCOUNT \
  --account-key $AZURE_STORAGE_KEY \
  --overwrite
```

Configure Kube Cluster

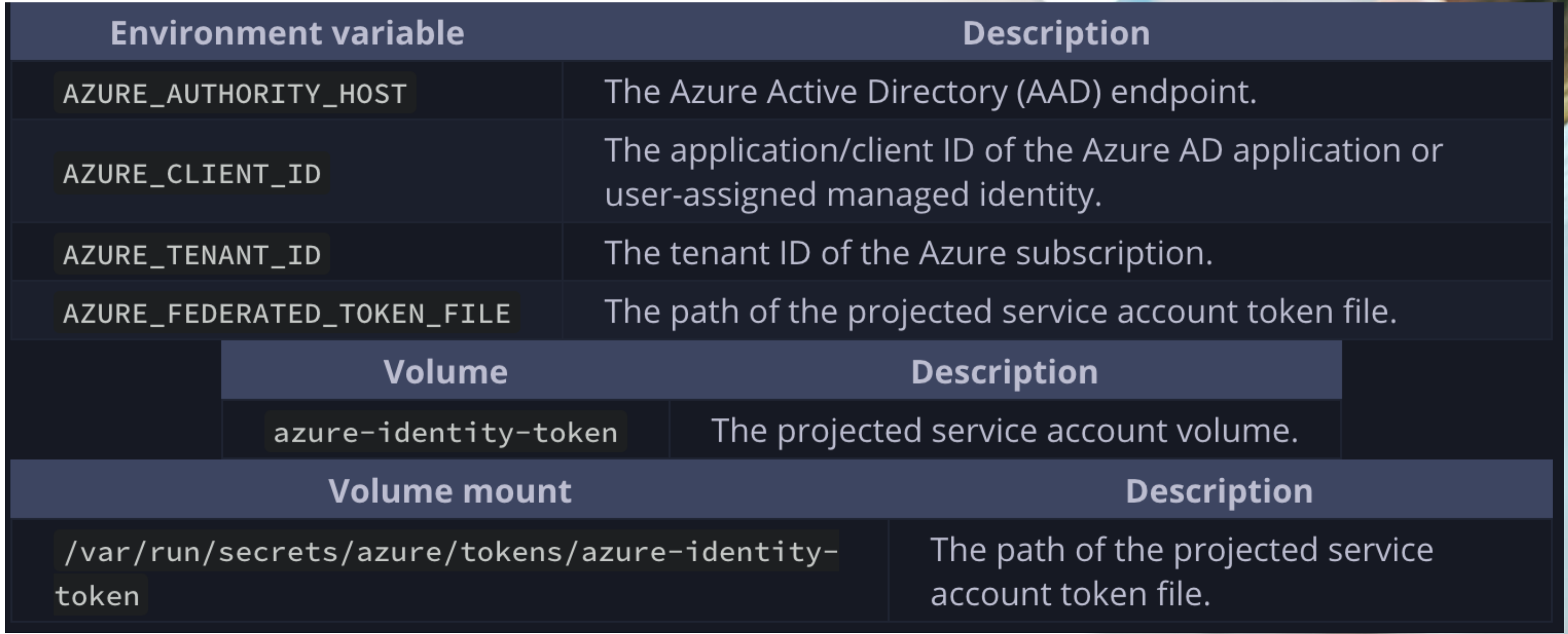
```
minikube start
minikube cp /home/$USER/sa.key /var/lib/minikube/certs/sa.key
minikube cp /home/$USER/sa.pub /var/lib/minikube/certs/sa.pub
minikube stop

echo "Starting Minikube with new config..."
minikube start \
    --extra-config=apiserver.service-account-issuer="https://${AZURE_STORAGE_ACCOUNT}.blob.core.windows.net/${AZURE_STORAGE_CONTAINER}/" \
    --extra-config=apiserver.service-account-signing-key-file="/var/lib/minikube/certs/sa.key" \
    --extra-config=apiserver.service-account-key-file="/var/lib/minikube/certs/sa.pub" \
    --extra-config=controller-manager.service-account-private-key-file="/var/lib/minikube/certs/sa.key"
```

Generate App Reg (See Bicep)

Install mutating webhook

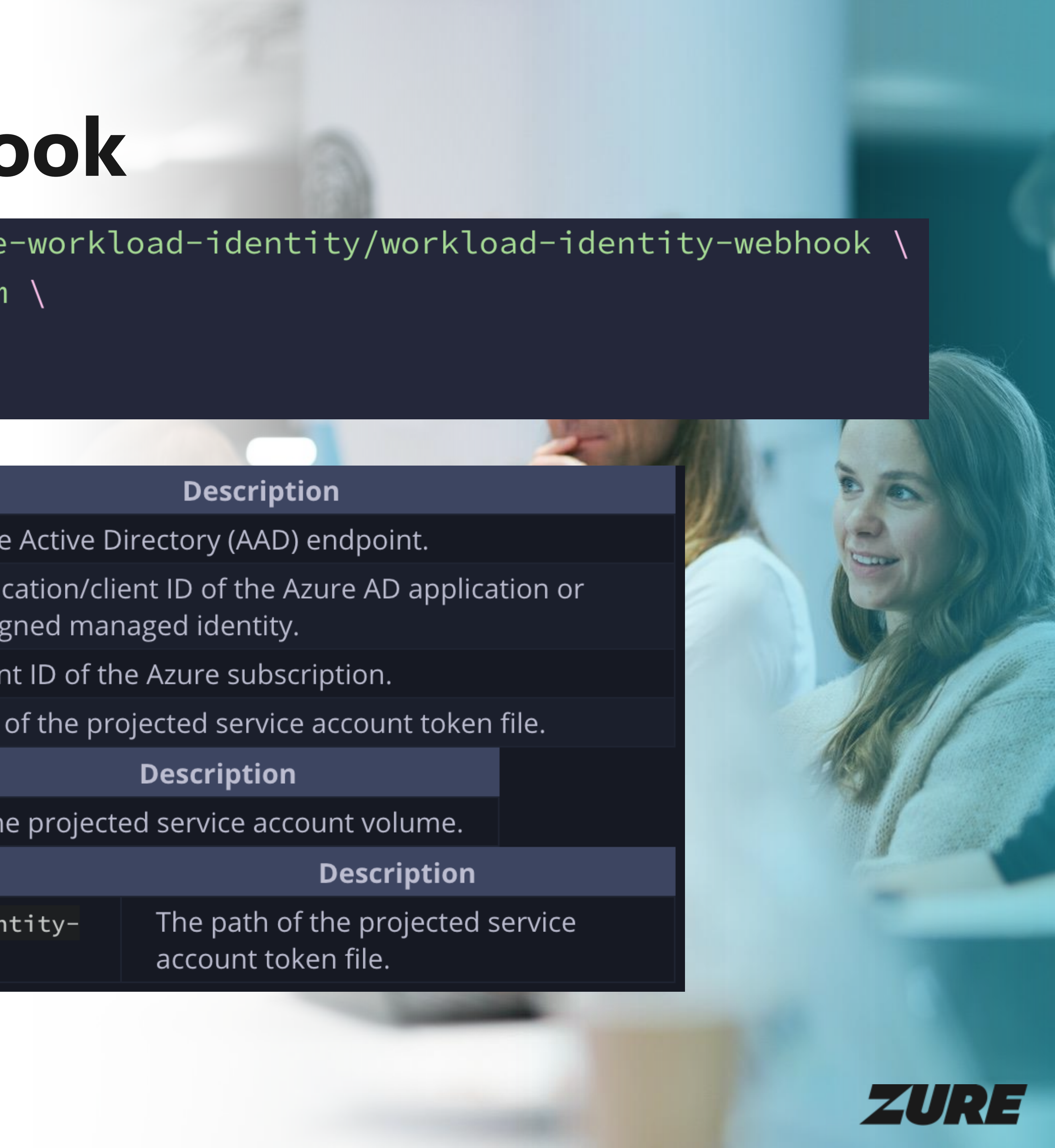
```
helm install workload-identity-webhook azure-workload-identity/workload-identity-webhook \
  --namespace azure-workload-identity-system \
  --create-namespace \
  --set azureTenantID="${AZURE_TENANT_ID}"
```



Environment variable	Description
`AZURE_AUTHORITY_HOST`	The Azure Active Directory (AAD) endpoint.
`AZURE_CLIENT_ID`	The application/client ID of the Azure AD application or user-assigned managed identity.
`AZURE_TENANT_ID`	The tenant ID of the Azure subscription.
`AZURE_FEDERATED_TOKEN_FILE`	The path of the projected service account token file.

Volume	Description
`azure-identity-token`	The projected service account volume.

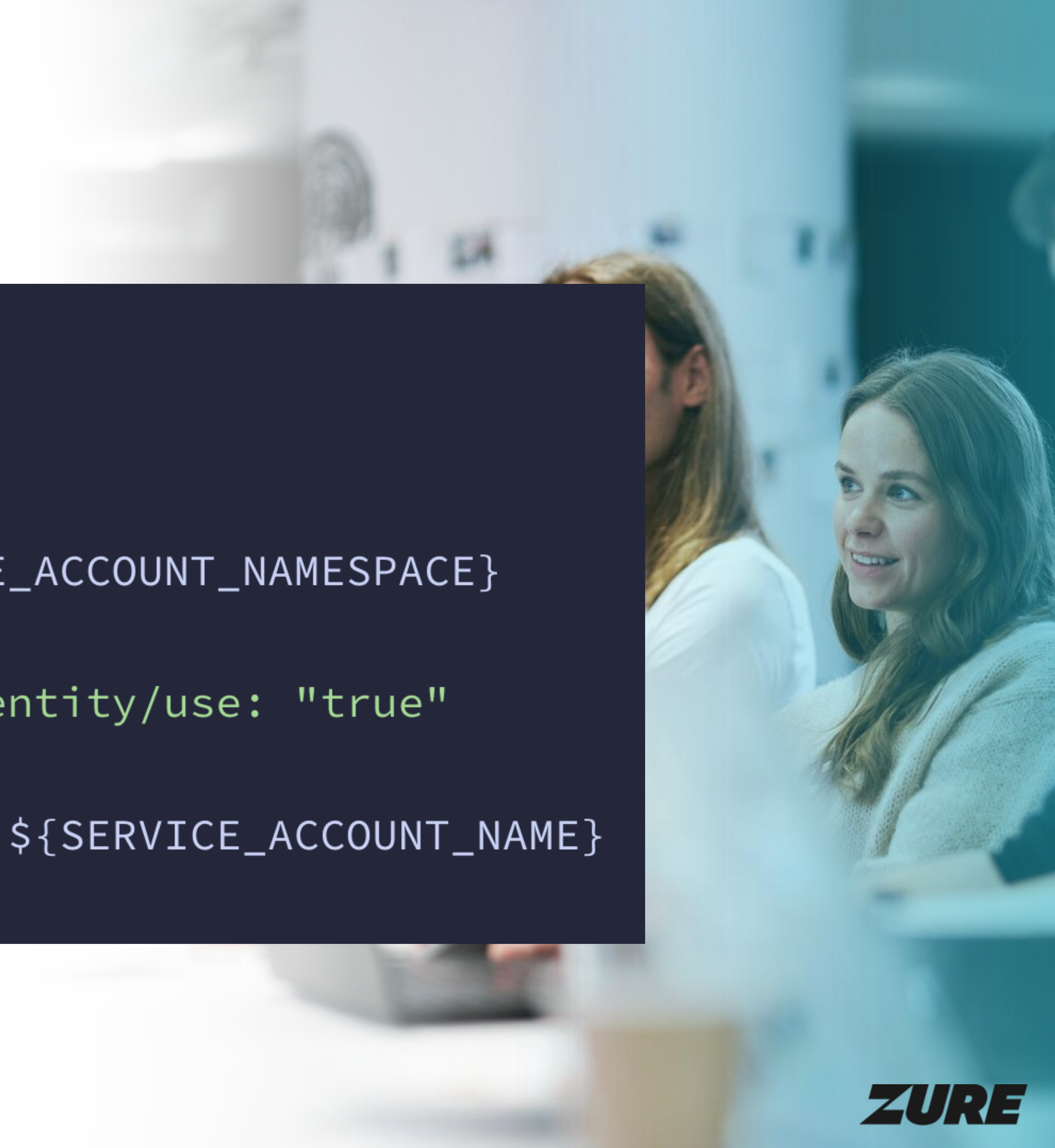
Volume mount	Description
`/var/run/secrets/azure/tokens/azure-identity-token`	The path of the projected service account token file.

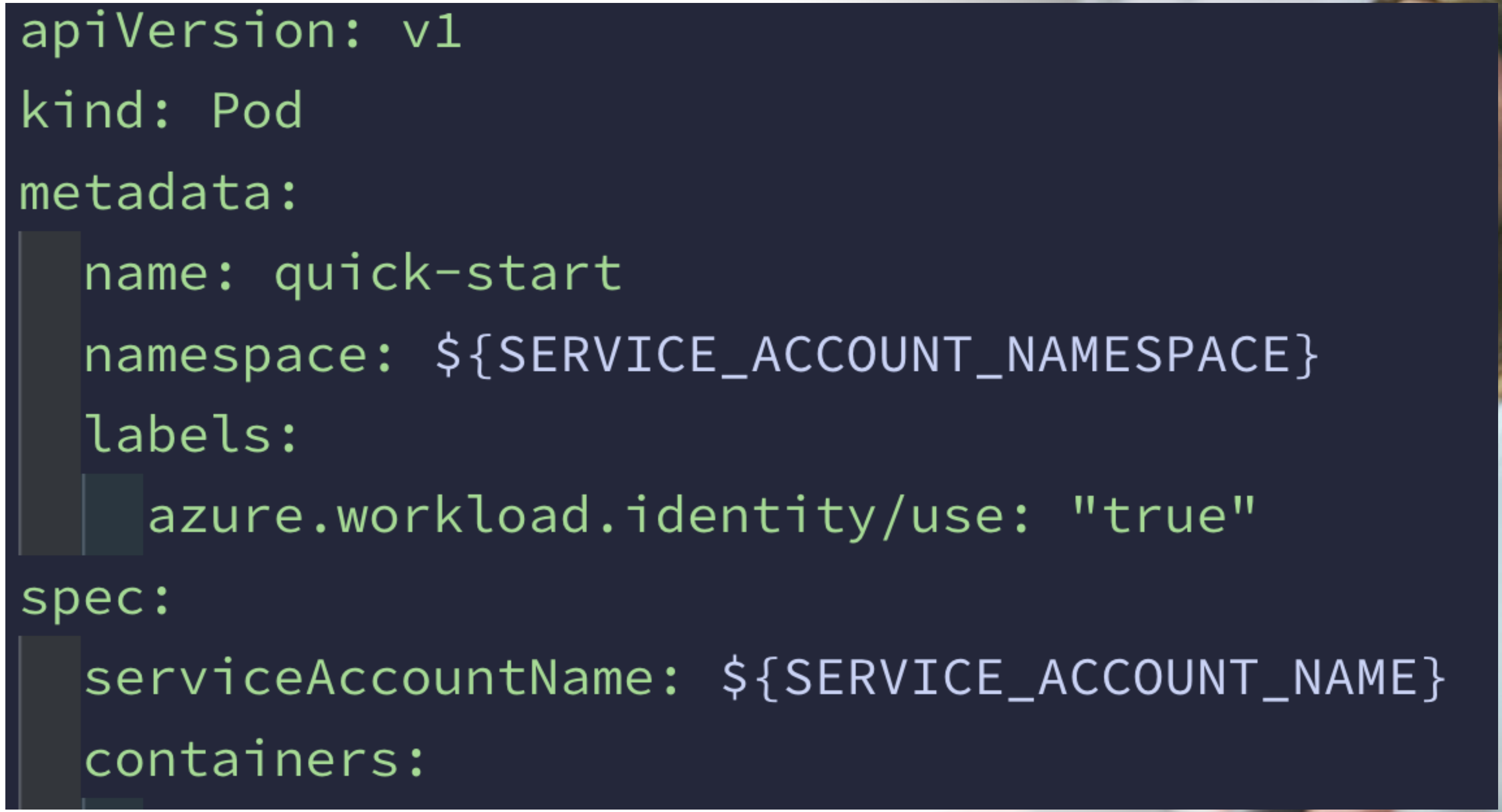


Create Service Account

```
azwi sa create phase service-account\
  --service-account-namespace $SERVICE_ACCOUNT_NAMESPACE \
  --service-account-name $SERVICE_ACCOUNT_NAME \
  --aad-application-client-id $AAD_APPLICATION_ID
```

Deploy Workload





```
apiVersion: v1
kind: Pod
metadata:
  name: quick-start
  namespace: ${SERVICE_ACCOUNT_NAMESPACE}
  labels:
    azure.workload.identity/use: "true"
spec:
  serviceAccountName: ${SERVICE_ACCOUNT_NAME}
  containers:
```

Questions?

All code: zure.ly/pasi/cb
Slides: zure.ly/pasi/cbslides

Blogs: huuhka.net
DevSecOps workshop: zure.ly/pasi/devsecops

Janne Mattila’s post on cross tenant access:
zure.ly/jmid

ZURE