

[Open in app](#)**Jonathan**[Follow](#)

29 Followers

[About](#)

Azure Kubernetes Service (AKS) with Azure Key Vault (AKV) Part 1 — Secrets Store CSI



Jonathan Apr 7 · 5 min read

The title of this article is awfully long, but the purpose for having both secrets store CSI and AKV provider in AKS environment is really simple, letting AKS to get AKV's resource, including secrets, certificates and keys, as native resource. In order to achieve that, we would have to implement pod identity or Azure active directory service principal (AAD SP) as the object that has sufficient permissions. We would use AAD SP for the following content.

Step-By-Step Guidance

1. Create an AKS resource: Create a resource group then an AKS cluster
2. Create an AKV resource

```
az keyvault create -g <resource group name> -n <key vault name> --  
location <Ex: westus2, eastus...>
```

3. Create a namespace to store every resource for this demonstration

```
kubectl create ns <namespace name>
```

[Open in app](#)

bookbuyer	Active	2d7h
bookstore	Active	2d7h
bookthief	Active	2d7h
bookwarehouse	Active	2d7h
curl	Active	2d9h
default	Active	2d23h
httpbin	Active	2d
ingress-test	Active	6m1s
kube-node-lease	Active	2d23h
kube-public	Active	2d23h
kube-system	Active	2d23h
osm-system	Active	2d23h

4. Create an AAD SP and set appropriate permissions to it to manage AKV

```
export SERVICE_PRINCIPAL_CLIENT_SECRET="$(az ad sp create-for-rbac -
-skip-assignment --name http://secrets-store-test --query 'password'
-otsv) "
```

```
export SERVICE_PRINCIPAL_CLIENT_ID="$(az ad sp show --id
http://secrets-store-test --query 'appId' -otsv) "
```

```
az keyvault set-policy -n <AKV name> --secret-permissions get --spn
${SERVICE_PRINCIPAL_CLIENT_ID}
```

```
jonw@JONWSL3:~/aks-keyvault$ az ad sp list --display-name "http://secrets-store-test"
[
  {
    "accountEnabled": "True",
    "addIns": [],
    "alternativeNames": [],
    "appDisplayName": "http://secrets-store-test",
    "appId": "a6c8e8e8-8e8e-8e8e-8e8e-8e8e8e8e8e8e",
    "appOwnerTenantId": "72f988bf-86f1-41af-b1b7-b2a575a367e9",
    "appRoleAssignmentRequired": false,
    "appRoles": [],
    "applicationTemplateId": null,
    "deletionTimestamp": null,
    "displayName": "http://secrets-store-test",
    "errorUrl": null,
    "homepage": null,
    "informationalUrls": {
      "marketing": null,
      "privacy": null,
      "support": null,
      "termsOfService": null
    },
    "keyCredentials": [],
    "logoutUrl": null,
    "notificationEmailAddresses": [],
    "oauth2Permissions": [],
```

[Open in app](#)

5. Create a secret within the AKS cluster as the identity managing AKV in the future steps. Label the secret.

```
# Create a secret with AAD SP client ID and secret
kubectl create secret generic secrets-store-creds --from-literal
clientid=${SERVICE_PRINCIPAL_CLIENT_ID} --from-literal
clientsecret=${SERVICE_PRINCIPAL_CLIENT_SECRET} -n <namespace name>

# Label the just-created secret
kubectl label secret secrets-store-creds secrets-
store.csi.k8s.io/used=true

# Check whether the secret has been created in the environment
kubectl get secrets secrets-store-cred -n <namespace name>
```

```
jonw@JONWSL3:~/aks-keyvault$ kubectl get secrets secrets-store-creds -n ingress-test
NAME                TYPE      DATA   AGE
secrets-store-creds  Opaque    2       56m
```

6.. Generate a TLS certificate in the Linux environment

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -out ingress-tls.crt \
  -keyout ingress-tls.key \
  -subj "/CN=demo.test.com/O=ingress-tls"
```

```
jonw@JONWSL3:~/aks-keyvault$ ls
akv-secret.yaml  ingress-tls.key  pod-get-aks-secrets.yaml  secrets-store-class.yaml
ingress-tls.crt  ingresscert.pfx  secrets-store-class-certificate.yaml  test-nginx.yaml
```

7. Import the TLS certificate in AKV

```
# export the a .pfx file with both .crt and .key
# skip Password prompt
openssl pkcs12 -export -in ingress-tls.crt -inkey ingress-tls.key -
out <certificate name>.pfx

az keyvault certificate import --vault-name <AKV name> -n
<certificate name> -f <certificate name>.pfx
```

[Open in app](#)

```

jonw@JONWSL3:~/aks-keyvault$ az keyvault certificate list --vault-name osmakskeyvault
[
  {
    "attributes": {
      "created": "2021-04-06T20:16:12+00:00",
      "enabled": true,
      "expires": "2022-04-06T20:16:12+00:00",
      "notBefore": "2021-04-06T20:06:12+00:00",
      "recoveryLevel": null,
      "updated": "2021-04-06T20:16:12+00:00"
    },
    "id": "https://osmakskeyvault.vault.azure.net/certificates/helloworldcert",
    "name": "helloworldcert",
    "subject": "",
    "tags": null,
    "x509Thumbprint": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "x509ThumbprintHex": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
  },
  {
    "attributes": {
      "created": "2021-04-06T21:17:07+00:00",
      "enabled": true,
      "expires": "2022-04-06T21:14:05+00:00",
      "notBefore": "2021-04-06T21:14:05+00:00",
      "recoveryLevel": null,
      "updated": "2021-04-06T21:17:07+00:00"
    },
    "id": "https://osmakskeyvault.vault.azure.net/certificates/ingresscert",
    "name": "ingresscert",
    "subject": "",
    "tags": null,
    "x509Thumbprint": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "x509ThumbprintHex": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
  }
]

```

8. Deploy SecretProviderClass. Create a new file named “secretproviderclass.yaml” and create it via “kubectl apply -f secretproviderclass.yaml”.

```

# the content of secretproviderclass.yaml
apiVersion: secrets-store.csi.x-k8s.io/v1alpha1
kind: SecretProviderClass
metadata:
  name: azure-tls
spec:
  provider: azure
  secretObjects:
  - secretName: ingress-tls-csi
    type: kubernetes.io/tls
    data:
    - objectName: <certificate name>
      key: tls.key
    - objectName: <certificate name>

```

[Open in app](#)

```

keyvaultName: <AKV name> # the name of the KeyVault
objects: |
  array:
    - |
      objectName: <certificate name>
      objectType: secret
  tenantId: <AAD tenant ID> # the tenant ID of the KeyVault

# create the secret provider class
kubectl apply -f secretproviderclass.yaml -n <namespace name>

# Check whether secretproviderclass has been created successfully
kubectl get secretproviderclasses -n <namespace name>

```

```

jonw@JONWSL3:~/aks-keyvault$ kubectl get secretproviderclass -n ingress-test
NAME          AGE
azure-tls     59m

```

9. Create NGINX Ingress Controller

```

# add and HELM repo
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

# update HELM repo
helm repo update

# create the NGINX Ingress Controller with HELM
helm install ingress-nginx/ingress-nginx --generate-name \
  --namespace <namespace name> \
  --set controller.replicaCount=2 \
  --set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \
  --set
defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux \
  -f - <<EOF
controller:
  extraVolumes:
    - name: secrets-store-inline
      csi:
        driver: secrets-store.csi.k8s.io
        readOnly: true
        volumeAttributes:
          secretProviderClass: "azure-tls"
        nodePublishSecretRef:
          name: secrets-store-creds
  extraVolumeMounts:
    - name: secrets-store-inline
      mountPath: "/mnt/secrets-store"

```


[Open in app](#)

```
# Check whether all related resources are up and running, especially
the secret "ingress-tls-csi"
kubectl get deploy,pod,svc,ing,secret -n <namespace name>
```

```
jon@JONWSL3:~/aks-keyvault$ kubectl get deploy,pod,svc,ing,secret -n ingress-test
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/ingress-nginx-controller	1/1	1	1	12h
deployment.apps/nginx-one	1/1	1	1	12h
deployment.apps/nginx-two	1/1	1	1	12h

NAME	READY	STATUS	RESTARTS	AGE
pod/ingress-nginx-admission-create-vggm4	0/1	Completed	0	12h
pod/ingress-nginx-admission-patch-g5h88	0/1	Completed	0	12h
pod/ingress-nginx-controller-6f5454cbfb-lj8zm	1/1	Running	0	12h
pod/nginx-one-5667bf6dbd-5zt7	1/1	Running	0	12h
pod/nginx-two-549b64f978-zmj67	1/1	Running	0	12h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/ingress-nginx-controller	LoadBalancer	10.0.196.217	20.72.220.43	80:38536/TCP,443:32572/TCP	12h
service/ingress-nginx-controller-admission	ClusterIP	10.0.62.197	<none>	443/TCP	12h
service/nginx-one	ClusterIP	10.0.91.78	<none>	80/TCP	12h
service/nginx-two	ClusterIP	10.0.97.148	<none>	80/TCP	12h

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress.extensions/ingress-tls	<none>	demo.test.com	20.72.220.43	80, 443	12h

NAME	TYPE	DATA	AGE
secret/default-token-wcdc4	kubernetes.io/service-account-token	3	12h
secret/ingress-nginx-admission	Opaque	3	12h
secret/ingress-nginx-admission-token-tx2sk	kubernetes.io/service-account-token	3	12h
secret/ingress-nginx-token-kq64j	kubernetes.io/service-account-token	3	12h
secret/ingress-tls-csi	kubernetes.io/tls	2	12h
secret/secrets-store-creds	Opaque	2	12h

10. Create test applications and Ingress

```
# create app 1
kubectl apply -f https://raw.githubusercontent.com/Azure/secrets-store-csi-driver-provider-azure/5eff51f5a04b5e91db5c18080c30316a5dee772a/docs/sample/ingress-controller-tls/deployment-app-one.yaml -n <namespace name>
```

```
# create app 2
kubectl apply -f https://raw.githubusercontent.com/Azure/secrets-store-csi-driver-provider-azure/5eff51f5a04b5e91db5c18080c30316a5dee772a/docs/sample/ingress-controller-tls/deployment-app-two.yaml -n <namespace name>
```

```
# create Ingress
kubectl apply -f https://raw.githubusercontent.com/Azure/secrets-store-csi-driver-provider-azure/5eff51f5a04b5e91db5c18080c30316a5dee772a/docs/sample/ingress-controller-tls/ingress.yaml -n <namespace name>
```

```
# Check whether all related resources are up and running
kubectl get deploy,pod,svc,ing -n <namespace name>
```

[Open in app](#)

NAME	READY	STATUS	RESTARTS	AGE
pod/ingress-nginx-admission-create-vggm4	0/1	Completed	0	52m
pod/ingress-nginx-admission-patch-g5h88	0/1	Completed	0	52m
pod/ingress-nginx-controller-6f5454cbfb-lj8zm	1/1	Running	0	52m
pod/nginx-one-5667bf6dbd-5ztx7	1/1	Running	0	50m
pod/nginx-two-549b64f978-zmj67	1/1	Running	0	49m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/ingress-nginx-controller	LoadBalancer	10.0.196.217	20.72.220.43	80:30536/TCP, 443:32572/TCP	52m
service/ingress-nginx-controller-admission	ClusterIP	10.0.62.197	<none>	443/TCP	52m
service/nginx-one	ClusterIP	10.0.91.78	<none>	80/TCP	50m
service/nginx-two	ClusterIP	10.0.97.140	<none>	80/TCP	49m

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress.extensions/ingress-tls	<none>	demo.test.com	20.72.220.43	80, 443	45m

11. Get the Ingress public IP address and try to curl the service

```
# Get Ingress public IP address
kubectl get ing -n <namespace name>
```

```
# curl the service
curl -v -k --resolve demo.test.com:443:<public IP address>
https://demo.test.com
```

```
jonw@JONWSL3:~/aks-keyvault$ kubectl get ing -n ingress-test
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
NAME      CLASS      HOSTS      ADDRESS      PORTS      AGE
ingress-tls  <none>    demo.test.com  20.72.220.43  80, 443    48m
jonw@JONWSL3:~/aks-keyvault$ curl -vk --resolve demo.test.com:443:20.72.220.43 https://demo.test.com
* Added demo.test.com:443:20.72.220.43 to DNS cache
* Hostname demo.test.com was found in DNS cache
* Trying 20.72.220.43:443 ...
* TCP_NODELAY set
* Connected to demo.test.com (20.72.220.43) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*  CAfile: /etc/ssl/certs/ca-certificates.crt
*  CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server accepted to use h2
* Server certificate:
*  subject: CN=demo.test.com; O=ingress-tls
*  start date: Apr  6 21:14:05 2021 GMT
*  expire date: Apr  6 21:14:05 2022 GMT
*  issuer: CN=demo.test.com; O=ingress-tls
*  SSL certificate verify result: self signed certificate (18), continuing anyway.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x564baa4087c0)
> GET / HTTP/2
> Host: demo.test.com
> user-agent: curl/7.68.0
> accept: */*
```

[Open in app](#)

```
Error from server (InternalError): error when creating
"https://raw.githubusercontent.com/Azure/secrets-store-csi-driver-
provider-
azure/5eff51f5a04b5e91db5c18080c30316a5dee772a/docs/sample/ingress-
controller-tls/ingress.yaml": Internal error occurred: failed
calling webhook "validate.nginx.ingress.kubernetes.io": Post
"https://ingress-nginx-1617752626-controller-admission.ingress-
test.svc:443/networking/v1beta1/ingresses?timeout=10s": service
"ingress-nginx-1617752626-controller-admission" not found
```

The workaround is to remove the unnecessary ValidatingWebhookConfiguration.

```
# Get Ingress service
kubectl get svc -n <namespace name>

# Get all ValidatingWebhookConfigurations
kubectl get ValidatingWebhookConfiguration -n <namespace name>

# Delete every ValidatingWebhookConfigurations that does not have
the same naming convention you are seeing in Ingress service
kubectl delete ValidatingWebhookConfiguration -n <namespace name>
```

I was having too many ValidatingWebhookConfiguration when first queried. After deleting, I have left only the one that has the same naming convention seeing in Ingress service.

```
jonw@JONWSL3:~/aks-keyvault$ kubectl get deploy,pod,svc,ing,secret,secretproviderclass -n ingress-test
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/ingress-nginx-controller  1/1      1              1            2m35s
deployment.apps/nginx-one              1/1      1              1            24s

NAME                                READY    STATUS      RESTARTS    AGE
pod/ingress-nginx-admission-create-vggm4  0/1      Completed   0            2m34s
pod/ingress-nginx-admission-patch-g5h88    0/1      Completed   0            2m34s
pod/ingress-nginx-controller-6f5454cbfb-lj8zm  1/1      Running     0            2m35s
pod/nginx-one-5667bf6dbd-5rtx7             1/1      Running     0            24s

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)                                AGE
service/ingress-nginx-controller      LoadBalancer  10.0.196.217  20.72.220.43   80:30536/TCP,443:32572/TCP            2m35s
service/ingress-nginx-controller-admission  ClusterIP     10.0.62.197   <none>          443/TCP                               2m35s
service/nginx-one                     ClusterIP     10.0.91.78    <none>          80/TCP                                24s

NAME                                TYPE          DATA          AGE
secret/default-token-wcdc4           kubernetes.io/service-account-token  3            11m
secret/ingress-nginx-admission        Opaque        3            2m33s
secret/ingress-nginx-admission-token-tx2sk  kubernetes.io/service-account-token  3            2m35s
secret/ingress-nginx-token-kq64j       kubernetes.io/service-account-token  3            2m35s
secret/ingress-tls-csi                kubernetes.io/tls                        2            23s
secret/secrets-store-creds            Opaque        2            11m
```


[Open in app](#)

```

ingress-nginx-admission      1      32h
osm-webhook-osm              1      2d8h
jonw@JONWSL3:~/aks-keyvault$ kubectl delete ValidatingWebhookConfiguration ingress-nginx-1617751997-admission
validatingwebhookconfiguration.admissionregistration.k8s.io "ingress-nginx-1617751997-admission" deleted
jonw@JONWSL3:~/aks-keyvault$ kubectl delete ValidatingWebhookConfiguration ingress-nginx-1617752626-admission
validatingwebhookconfiguration.admissionregistration.k8s.io "ingress-nginx-1617752626-admission" deleted
jonw@JONWSL3:~/aks-keyvault$ kubectl delete ValidatingWebhookConfiguration ingress-nginx-1617754834-admission
validatingwebhookconfiguration.admissionregistration.k8s.io "ingress-nginx-1617754834-admission" deleted
jonw@JONWSL3:~/aks-keyvault$ kubectl get ValidatingWebhookConfiguration
NAME                                WEBHOOKS  AGE
ingress-nginx-admission            1         32h
osm-webhook-osm                    1         2d8h

```

Here are all the links I have taken for reference for composing this article.

- [Secrets Store CSI and AKV Provider GitHub for NGINX Ingress Controller TLS Connection](#)
- [Secrets Store CSI and AKV Provider GitHub for Pod](#)
- [Azure CLI for AAD SP](#)
- [NGINX Installation on AKS or Azure-Bare-Metal K8s](#)
- [How to Replace String in Linux](#)
- [How to Resolve NGINX Error Message about Ingress Validation](#)

Hope this would save the time of people trying to learn how to leverage secrets store CSI and AKV provider as native AKS resources. Happy learning!

[Azure Key Vault](#)[Azure Kubernetes Service](#)[Nginx Ingress](#)[Secret Store Csi](#)[Demo](#)[About](#) [Help](#) [Legal](#)

Open in app

