how to test this azure file share with pod

**ChatGPT said:**

Great! Let's walk through how to **test your Azure File Share** with a pod after it's dynamically provisioned via a PVC in AKS.

**✅ Steps to Test Azure File Share with Pod**

**🔹 Step 1: PVC (already created, assuming)**

If you already have a PersistentVolumeClaim like this:

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: azurefile-pvc

spec:

accessModes:

- ReadWriteMany

storageClassName: azurefile-dynamic

resources:

requests:

storage: 5Gi

You're good to go!

**🔹 Step 2: Create a Test Pod that Mounts the PVC**

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: azurefile-test-pod

spec:

containers:

- name: busybox

image: busybox

command: [ "sleep", "3600" ] # keeps pod running

volumeMounts:

- name: azurefile-vol

mountPath: /mnt/azure

volumes:

- name: azurefile-vol

persistentVolumeClaim:

claimName: azurefile-pvc

Apply it:

bash

CopyEdit

kubectl apply -f azurefile-test-pod.yaml

**🔹 Step 3: Connect to the Pod**

bash

CopyEdit

kubectl exec -it azurefile-test-pod -- sh

Inside the pod:

bash

CopyEdit

cd /mnt/azure

echo "Hello from Azure File Share!" > testfile.txt

cat testfile.txt

If you can write and read the file — ✅ it means your Azure File Share is mounted and working properly.

**🔹 Step 4: Verify in Azure**

Go to the **Azure Portal** > Storage Account > **File shares**.  
Open the auto-created share (name like pvc-xxxxx) > **Browse**.  
You’ll see testfile.txt there.

Let me know if you’d like a version with nginx instead of busybox, or a YAML bundle with everything!

Awesome! Here's a full YAML bundle for testing **Azure File Share** in AKS using:

✅ A StorageClass for dynamic provisioning  
✅ A PersistentVolumeClaim  
✅ A test pod (busybox) that mounts the Azure File Share and lets you test it

**📁 azurefile-test-bundle.yaml**

yaml

CopyEdit

---

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: azurefile-dynamic

provisioner: file.csi.azure.com

parameters:

skuName: Standard\_LRS

secretName: azure-secret

secretNamespace: default

shareName: "" # creates a new share per PVC

reclaimPolicy: Retain

volumeBindingMode: Immediate

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: azurefile-pvc

spec:

accessModes:

- ReadWriteMany

storageClassName: azurefile-dynamic

resources:

requests:

storage: 5Gi

---

apiVersion: v1

kind: Pod

metadata:

name: azurefile-test-pod

spec:

containers:

- name: busybox

image: busybox

command: ["sleep", "3600"]

volumeMounts:

- name: azurefile-vol

mountPath: /mnt/azure

volumes:

- name: azurefile-vol

persistentVolumeClaim:

claimName: azurefile-pvc

**🚀 Apply it**

bash

CopyEdit

kubectl apply -f azurefile-test-bundle.yaml

Then test:

bash

CopyEdit

kubectl exec -it azurefile-test-pod -- sh

cd /mnt/azure

echo "Hello Azure File!" > test.txt

cat test.txt

Let me know if you want to mount this into an **nginx** pod instead (e.g., for static site testing or shared content)!

Perfect! Here's how to mount the **Azure File Share** (created via your PVC) into an **NGINX pod** so it can serve static content or files from Azure File.

**📁 nginx-azurefile-pod.yaml**

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: nginx-azurefile-pod

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

volumeMounts:

- name: azurefile-vol

mountPath: /usr/share/nginx/html # nginx serves files from here

volumes:

- name: azurefile-vol

persistentVolumeClaim:

claimName: azurefile-pvc

**🚀 Apply it**

bash

CopyEdit

kubectl apply -f nginx-azurefile-pod.yaml

**✅ Test It**

1. **Access the pod shell (optional)**

bash

CopyEdit

kubectl exec -it nginx-azurefile-pod -- sh

echo "Welcome from Azure File!" > /usr/share/nginx/html/index.html

1. **Port-forward to test in browser**

bash

CopyEdit

kubectl port-forward pod/nginx-azurefile-pod 8080:80

Then open: <http://localhost:8080>

**🔄 Bonus**

Since it's using Azure File Share:

* You can **update the contents** from any other pod that mounts the same PVC.
* The data will **persist** even if the pod is deleted.

Let me know if you’d like to turn this into a deployment + service setup next!

4o

Awesome! Here's how to turn that setup into a **full Kubernetes deployment with a service**, so you can scale and expose your NGINX pod that serves files from an **Azure File Share**.

**📁 nginx-azurefile-deployment.yaml**

yaml

CopyEdit

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-azurefile-deployment

spec:

replicas: 1

selector:

matchLabels:

app: nginx-azurefile

template:

metadata:

labels:

app: nginx-azurefile

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

volumeMounts:

- name: azurefile-vol

mountPath: /usr/share/nginx/html

volumes:

- name: azurefile-vol

persistentVolumeClaim:

claimName: azurefile-pvc

---

apiVersion: v1

kind: Service

metadata:

name: nginx-azurefile-service

spec:

type: LoadBalancer # You can change this to ClusterIP or NodePort

selector:

app: nginx-azurefile

ports:

- port: 80

targetPort: 80

**🚀 Apply It**

bash

CopyEdit

kubectl apply -f nginx-azurefile-deployment.yaml

**✅ Access It**

**Option 1: 🌐 External (LoadBalancer)**

Wait for an external IP:

bash

CopyEdit

kubectl get svc nginx-azurefile-service

Then open http://<EXTERNAL-IP> in your browser.

**Option 2: 🔁 Local Testing (Port-forward)**

bash

CopyEdit

kubectl port-forward svc/nginx-azurefile-service 8080:80

Open: <http://localhost:8080>

**🔧 Update Content**

To test if the Azure File Share is working:

bash

CopyEdit

kubectl exec deploy/nginx-azurefile-deployment -- sh -c "echo 'Hello from shared Azure File!' > /usr/share/nginx/html/index.html"

Then refresh the page — you should see the message!

Let me know if you'd like to add an **Ingress** to expose it under a domain, or auto-create the Azure secret as part of the setup!