# Installing 3Scale on Azure OpenShift

The document is based on the following guide with some notes. It should be read while performing the corresponding steps.

<https://access.redhat.com/documentation/en-us/red_hat_3scale_api_management/2.7/html/installing_3scale/install-threescale-on-openshift-guide#system-requirements-for-installing-threescale-on-openshift>

Tested with 3Scale 2.7 and OpenShift 4.3.

## 2.1 System requirements:

As noted in the guide, 3Scale needed 4 PVs: 3 ReadWriteOnce and 1 ReadWriteMany. When creating OpenShift on Azure, a default Storage Class by the name “Manage Premium” is created. This storage class is based on azure-disk, which provides only ReadWriteOnce ability. In addition, the class has the following property:

volumeBindingMode: WaitForFirstConsumer.

From Kubernetes documentation:

*The volumeBindingMode field controls when*[*volume binding and dynamic provisioning*](https://kubernetes.io/docs/concepts/storage/persistent-volumes/#provisioning)*should occur.*

*By default, the Immediate mode indicates that volume binding and dynamic provisioning occurs once the PersistentVolumeClaim is created. For storage backends that are topology-constrained and not globally accessible from all Nodes in the cluster, PersistentVolumes will be bound or provisioned without knowledge of the Pod’s scheduling requirements. This may result in unschedulable Pods.*

*A cluster administrator can address this issue by specifying the WaitForFirstConsumer mode which will delay the binding and provisioning of a PersistentVolume until a Pod using the PersistentVolumeClaim is created. PersistentVolumes will be selected or provisioned conforming to the topology that is specified by the Pod’s scheduling constraints. These include, but are not limited to,*[*resource requirements*](https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container)*,*[*node selectors*](https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#nodeselector)*,*[*pod affinity and anti-affinity*](https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity)*, and*[*taints and tolerations*](https://kubernetes.io/docs/concepts/configuration/taint-and-toleration)*.*

All of the PVCs created during 3Scale’s deployment are using the default storage class. In my case, I encountered a problem when deploying 3Scale. All the of PVC were stuck at pending state.

The PVC for RWX (ReadWriteMany) was understandably stuck, because the storage class couldn’t provide that capability. The 3 RWO (ReadWriteOnce) PVC were what puzzled me.

As far as I understand, this happened because of a kind of dead-lock – the PVC was waiting for the pods to be created (because of WaitForFirstConsumer binding mode) and the pod was waiting for the PVC.

To change that, I create a new StorageClass, named RWOclass, and changed the following parameter:

**volumeBindingMode: Immediate**

In addition, I created another class called RWXclass, to address the issues of the RWX PVC (called system-storage). To do that, follow those guides:

Create a file share in azure: <https://docs.microsoft.com/en-us/azure/storage/files/storage-how-to-create-file-share?tabs=azure-portal>

Create a PV that references the share: <https://docs.openshift.com/container-platform/4.3/storage/persistent_storage/persistent-storage-azure-file.html>

Don’t create the PVC yet, stop after create the PV. You can use 100Mi as your size (that what the System-Storage PVC needs).

## 2.2 Configuring nodes:

The information in the guide (as of March 20) is not relevant for Openshift 4, where the installation is different. To get the amp.yml file, just download it from here: <https://github.com/3scale/3scale-amp-openshift-templates/blob/master/amp/amp.yml>.

Place is someplace with oc available.

## 2.3 Deploying 3Scale

Notice that step 2.3.2 should be performed before 2.3.1.

After creating the needed service account and 3Scale project, you are ready to deploy 3Scale:

*oc new-app --file amp.yml --param WILDCARD\_DOMAIN=apps.OPENSHIFT-DOMAIN.cloudlet-dev.com.*

Take notice that the wildcard domain is not your OpenShift domain, but starts with “apps.”. Otherwise, the routes the 3Scale deployment will create will not be exposed through the DNS rules the OpenShift installation created.

The first thing I encountered was that the pods were waiting for the images to be pulled. I couldn’t figure out why that didn’t happen automatically, so I pushed them manually:

*oc import-image istag/amp-apicast:latest*

*oc import-image istag/amp-backend:latest*

*oc import-image istag/amp-zync:latest*

*oc import-image istag/zync-database-postgresql*

*oc import-image istag/amp-system:latest*

*oc import-image istag/system-redis:latest*

*oc import-image istag/system-mysql:latest*

*oc import-image istag/system-memcached:latest*

*oc import-image istag/backend-redis:latest*

Then the Pods started to run successfully, and half of the 14 needed ones deployed.

### PVC

Now we are back to the PVC issues. If the 3 RWO PVCs are in state “Bound” and not “Pending”, than everything’s alright. If not, you need to edit the PVCs to use the new storage class (RWOclass) that we created previously.

To do that, use:

oc edit pvc PVC-NAME

Copy the yaml file and create a new file with the same content (lets call it PVC-NAME.yml). Then, edit the file and change the StorageClass to RWOclass.

Delete the old pvc using:

oc delete pvc PVC-NAME

(If the pvc is stuck at “terminating”, I used the following command to solve it:

*kubectl patch pvc system-storage -p '{"metadata":{"finalizers": []}}' --type=merge*)

than recreate it:

oc apply -f PVC-NAME.yml

An example of one the PVC I edited (mysql-storage.yml)

*apiVersion: v1*

*kind: PersistentVolumeClaim*

*metadata:*

*annotations:*

*openshift.io/generated-by: OpenShiftNewApp*

*creationTimestamp: "2020-03-16T06:55:55Z"*

*finalizers:*

*- kubernetes.io/pvc-protection*

*labels:*

*app: 3scale-api-management*

*threescale\_component: system*

*threescale\_component\_element: mysql*

*name: mysql-storage*

*namespace: 3scale*

*resourceVersion: "151349"*

*selfLink: /api/v1/namespaces/3scale/persistentvolumeclaims/mysql-storage*

*uid: cdd25149-b9b6-45fb-bd44-ea453f2e62ce*

*spec:*

*accessModes:*

*- ReadWriteOnce*

*resources:*

*requests:*

*storage: 1Gi*

***storageClassName: RWOclass***

*volumeMode: Filesystem*

If you followed the steps successfully, 3 of the 4 PVCs should be now in “Bound” state.

To fix the 4th PVC, follow the same steps:

*oc edit pvc system-storage*

*vi system-storage.yml*

edit the content to use the PV and the RWXclass we created earlier. It should look like this:

*apiVersion: v1*

*kind: PersistentVolumeClaim*

*metadata:*

*….*

*spec:*

*accessModes:*

*- ReadWriteMany*

*resources:*

*requests:*

*storage: 1Gi*

***storageClassName: RWXclass***

***volumeName: "pv-rwx"***

Then run:

*oc delete pvc system-storage*

*oc apply -f system-storage.yml*

Wait around 10 minutes, and 3Scale should be running!