# Portworx Enterprise on Microsoft Azure Kubernetes Service (AKS) Reference Architecture

Quickstart reference deployment

portworx by Pure Storage

Azure

# 1. OVERVIEW

This reference architecture and deployment guide provides step-by-step instructions for deploying Portworx Enterprise on the Microsoft Azure Kubernetes Service (AKS). This reference architecture is suitable for any users who want to use services from Azure—such as Azure Cloud Servers, Cloud Disks, and AKS—to launch Kubernetes and Portworx for their container environment.

This reference architecture document provides guidance for deploying a highly-available Kubernetes conformant control plane for Portworx Enterprise. Of course, you can customize your environment to suit specific needs, so this document should be taken as foundational guidance.

## 1.1 Portworx Enterprise on Azure Kubernetes Service

Portworx Enterprise is the software-defined container storage platform built from the ground up for Kubernetes. By providing scale-out software-defined container storage, data availability, data security, backup, and disaster recovery for Kubernetes-based applications running on-prem or across clouds, Portworx has helped dozens of Global 2000 companies—such as Carrefour, Comcast, GE Digital, Lufthansa, T-Mobile, and SAIC—run containerized data services in production.

With Portworx Enterprise, you can:

- Run any database or data-rich service on Kubernetes, even those that require strict performance, backup & DR, security, and data mobility capabilities.
- Improve application performance and uptime by avoiding the limitations of storage platforms like Azure block storage built for VMs, not containers.
- Achieve Zero RPO and < 1 minute RPO Disaster Recovery for mission-critical data services.
- Cut cloud storage costs in half without sacrificing performance.

Portworx Enterprise was named the #1 Kubernetes Storage Platform by GigaOm Research for the breadth of the solution, scale of supported use cases, and list of reference customers.

The Portworx Enterprise platform is made up of the following components that provide everything an enterprise needs to successfully run stateful applications on Azure Kubernetes Services.


### 1.1.1 PX-Store — Scalable Persistent Storage for Kubernetes

Built from the ground up for containers, PX-Store provides cloud native storage for applications running in the cloud, on-prem, and in hybrid/multi-cloud environments.

PX-Store includes:

- Container-optimized volumes with elastic scaling for no application downtime.
- High Availability across nodes/racks/AZs so you can failover in seconds.
- Multi-writer shared volumes across multiple containers.
- Storage-aware class-of-service (COS) and application-aware I/O tuning.
- And much more...

### 1.1.2 PX-DR — Data Protection for Mission Critical Apps

PX-DR extends the data protection included in PX-Store with Zero RPO Disaster Recovery for data centers in a metropolitan area as well as continuous backups across the WAN for an even greater level of protection.

PX-DR includes:

- Multi-site synchronous replication for Zero RPO DR across a metro area.
- Multi-site Asynchronous Replication for DR across a wide area network (WAN).
- The ability to set all DR policies at the container-granular level.

### 1.1.3 PX-Backup — What Backup for Kubernetes Should Be

PX-Backup allows you to capture entire applications—including data, application configuration, and Kubernetes objects—and move them to any backup location at the click of a button. You can recover entire applications just as easily.

PX-Backup includes:

- Continuous backups across global data centers.
- Point-and-click recovery for any Kubernetes app.
- Backup and recovery of cloud volumes from Amazon, Microsoft, and Google.
- The capability to fulfill your compliance and governance responsibilities with a single pane of glass for all your containerized applications.

### 1.1.4 PX-Autopilot for Capacity Management — Stop Over-provisioning Cloud Storage

PX-Autopilot for Capacity Management allows you to stop over-provisioning storage capacity in the cloud so you can cut your cloud storage bill in half.

PX-Autopilot includes:

- The ability to automatically resize individual container volumes or your entire storage pools.
- Rules-based engine that is fully customizable so you can optimize your apps based on performance requirements.
- Integration with Amazon EBS, Google PD, and Azure Block Storage.

### 1.1.5 PX-Secure — Container Data Security without Compromise

With PX-Secure encryption and access controls, you can move securely at the speed of Kubernetes.

PX-Secure includes:

- Cluster-wide encryption.
- Container-granular or Storage-class based BYOK encryption.
- Role-based access control for
  - Authorization
  - Authentication
  - Ownership
- Integration with Active Directory and LDAP.

### 1.1.6 PX-Migrate — Multi-cloud Data Mobility at Your Fingertips

Complete control over your Kubernetes data no matter where it lives.

PX-Migrate includes:

- Multi-cloud/multi-cluster application migration.
  - Snapshot-based backup to any cloud
  - Application-consistent snapshots

## 1.2 Cost

You are responsible for the cost of the Azure services used while running this reference deployment.

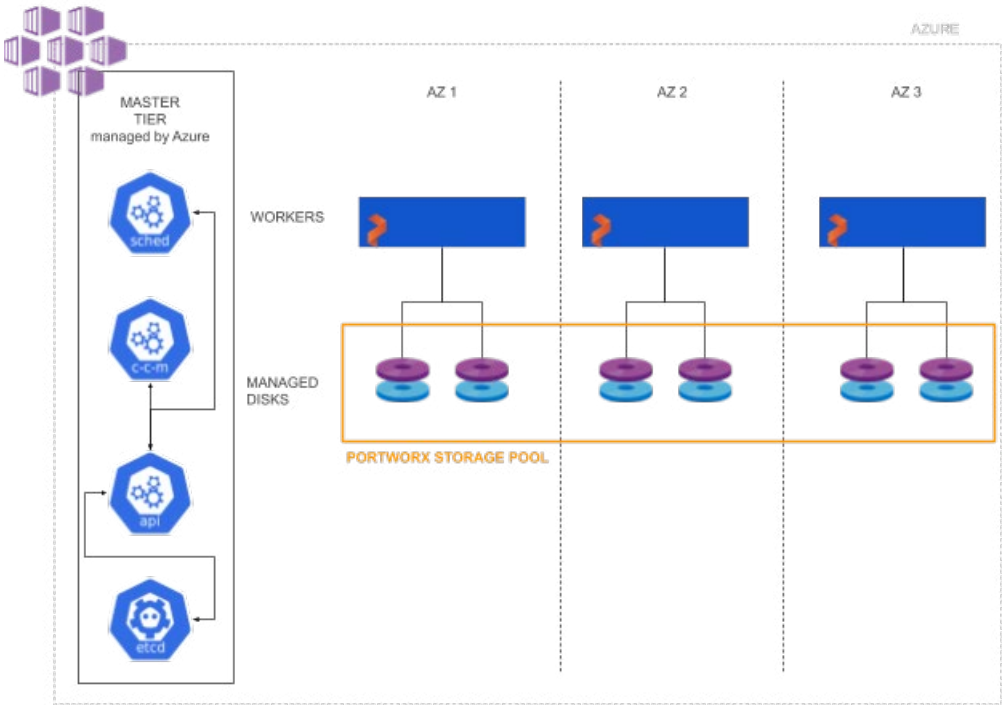
The guidance in this document includes configuration parameters that you can customize. Some of these settings—such as instance type—will affect the cost of deployment. For cost estimates, see the pricing pages (https://azure.microsoft.com/en-gb/pricing/) for each Azure service you will be using.

Portworx Enterprise is deployed as part of this guide and is available with a free 30-day license. After the trial is over, you can purchase a license to continue using the product. For more information on license types, what features are included with each type of license, and how to upgrade or transfer your license, visit the Portworx license documentation (https://docs.portworx.com/reference/knowledge-base/px-licensing/#Portworx Enterprise-license) or contact your sales representative.

Alternatively, you may wish to continue with PX-Essentials. This is a free version of the Portworx product with limited functionality. A comparison of the two options is available on the Portworx website: https://portworx.com/products/features/.

## 2. ARCHITECTURE

Deploying this reference architecture for an AKS cluster with attached cloud storage disks builds the following Portworx Enterprise environment in the Azure Cloud. The diagram shows AKS workers spread across three Availability Zones.




It's important to note that there are multiple availability zones. Using Azure Cloud Volumes without Portworx would normally limit nodes to a single AZ or stateful operations as described in the following taken from the Azure documentation: https://docs.microsoft.com/en-us/azure/aks/availability-zones#azure-disks-limitations.

**Azure disks limitations**

Volumes that use Azure managed disks are currently not zonal resources. Pods rescheduled in a different zone from their original zone can't reattach their previous disk(s). It's recommended to run stateless workloads that don't require persistent storage that may come across zonal issues.

If you must run stateful workloads, use taints and tolerations in your pod specs to tell the Kubernetes scheduler to create pods in the same zone as your disks. Alternatively, use network-based storage—such as Azure Files—that can attach to pods as they're scheduled between zones.

The above architecture with Portworx deployed mitigates the need for any zonal constraints or only running stateless workloads.

# 3. PLANNING THE DEPLOYMENT

## 3.1 Specialized knowledge

Before you deploy this reference architecture, we recommend that you become familiar with the following Azure services.

- Azure cloud servers
- Azure cloud volume (block storage devices)
- Azure resource groups
- Azure Kubernetes Service
- Portworx Enterprise

## 3.2 Azure account

If you don't already have an Azure account, create one at https://signup.azure.com by following the on-screen instructions. Part of the signup process involves selecting a support level— choose the one most suitable for your needs. Your Azure account is automatically signed up for all Azure services. You are charged only for the services you use unless you choose a paid support service.

## 3.3 Technical requirements

- Basic creation of an Azure account — This is a prerequisite for any actions on Azure. https://docs.microsoft.com/en-us/azure/guides/developer/azure-developer-guide
- Azure CLI installed https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest
- Knowledge of Kubernetes https://kubernetes.io/
- Knowledge of how to use kubectl https://kubernetes.io/docs/reference/kubectl/overview/

## 3.4 Infrastructure requirements

For Portworx Enterprise to be deployed, the AKS worker nodes must meet the following infrastructure requirements.

The documentation is available on the Portworx website: https://docs.portworx.com/start-here-installation/.

**Hardware**

| | |
|---|---|
| **CPU** | 4 cores |
| **RAM** | 4GB |
| **Disk (/var)** | 2GB free |
| **Backing drive** | 8GB (minimum required)<br>128 GB (minimum recommended) |
| **Storage drives** | Storage drives must be unmounted block storage: raw disks, drive partitions, LVM, or cloud block storage. |
| **Ethernet NIC card** | 10 GB (recommended) |

**Network**

| | |
|---|---|
| **Open needed ports** | TCP ports 9001-9022 and UDP port 9002 on all Portworx nodes. Also open the KVDB port. (As an example, etcd typically runs on port 2379.) |
| **Lighthouse** | If installing the Portworx Lighthouse management UI, please open ports 32678 and 32679. |

**Software**

| | |
|---|---|
| **Linux kernel** | Version 3.10 or greater |
| **Docker** | Version 1.13.1 or greater |
| **Key-value store** | Portworx needs a key-value store to perform its operations. As such, install a clustered key-value database (etcd or consul) with a three node cluster.<br><br>With Portworx 2.0, you can use Internal KVDB during installation. In this mode, Portworx will create and manage an internal key-value store (kvdb) cluster.<br><br>If you plan on using your own etcd, refer to Etcd for Portworx for details on recommendations for installing and tuning etcd. |
| **Disable swap** | Please disable swap on all nodes that will run the Portworx software. Ensure that the swap device is not automatically mounted on server reboot. |

## 4. DEPLOYMENT OPTIONS

This reference architecture includes two resources for deploying into AKS on Azure Cloud.

- **CLI-based manual deployment** — This option is a step-by-step guided deployment into a new resource group and includes Portworx Enterprise.
- **Template-based deployment** — This option uses a Quick Start ARM template to provision the infrastructure ready for you to then deploy Portworx Enterprise.

# 5. DEPLOYMENT STEPS

## 5.1 CLI-based manual deployment

This guide assumes you have the Azure CLI installed and configured. If not, please follow the documentation to do this: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest.

### Step 1 — Login to your Azure account

Start by using the Azure CLI to login to your account, and then set the subscription you will use for this deployment.

```
az login
```

Once you have logged in, set a specific subscription to use for this deployment if necessary. If you do not do this, the default or single subscription will be used.

```
az account set --subscription <Your-Azure-Subscription-UUID>
```

### Step 2 — Prepare the environment

Before deploying an AKS cluster and Portworx Enterprise, a resource group needs to be created.

```
az group create --name <Your-resource-group> --location <Your-location>
```

A list of available locations can be retrieved with the following command.

```
az account list-locations
```

To allow an AKS cluster to interact with other Azure resources, an Azure Active Directory service principal needs to be created. This will allow Portworx to automatically provision Azure Cloud Disks.

```
az ad sp create-for-rbac --role="Contributor" --scopes="/
subscriptions/<subscription-ID>"
```

Make a note of the output, as this will be required later.

### Step 3 — Deploy the AKS cluster

The following command will deploy an AKS cluster in the resource group previously created.

```
az aks create \
--resource-group joetest \--name myAKSCluster \
  --node-count 3 \
  --zones 1 2 3
  --node-vm-size Standard _ F16s _ v2 \
  --service-principal <ID from service principal> \
  --client-secret <password from service principal>
```

The full reference is available here:
https://docs.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest#az-aks-create.

The following items are important to note:

- **Availability Zones** — It is important to specify multiple availability zones to deploy a highly available, production-suitable cluster. Portworx provides replication across availability zones, avoiding the limitation of Azure Cloud Volumes.
- **VM type selection** — The Fsv2-series is based on the Intel® Xeon® Platinum 8168 processor. It features a sustained all core Turbo clock speed of 3.4 GHz and a maximum single-core turbo frequency of 3.7 GHz. Intel® AVX-512 instructions are new on Intel Scalable Processors. These instructions provide up to a 2X performance boost to vector processing workloads on both single and double precision floating point operations. In other words, they're really fast for any computational workload. Fsv2-series is chosen, as it offers premium storage options and an expected bandwidth of 7Gbps (https://docs.microsoft.com/en-us/azure/virtual-network/virtual-machine-network-throughput).
- **Node count** — Three nodes are selected, as this is the minimum number of workers required to run a production Portworx cluster. We recommend running more nodes than this, but the above command reflects the minimum.
- **SSH access** — A path to your public key can be included to allow worker node access or allow a keypair to be created.

## 5.2 Quick start ARM template deployment

A quickstart template is available in the Azure template directory: https://azure.microsoft.com/en-gb/resources/templates/101-aks/.

This template accepts parameters to deploy an AKS cluster in a resource group of your choice.

Please note that this template does not accept availability zone parameters, so it is not suitable for a production HA use case. We highly recommend using the CLI to define all availability zones for node placement.

Follow the quickstart template documentation to deploy into your Azure account.

## 5.3 Deploy Portworx

Before deploying Portworx, start by setting up kubectl CLI access to the deployed AKS cluster. First, use the Azure CLI to install the Kubernetes CLI.

```
az aks install-cli
```

Then configure kubectl to connect to the AKS cluster.

```
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
```

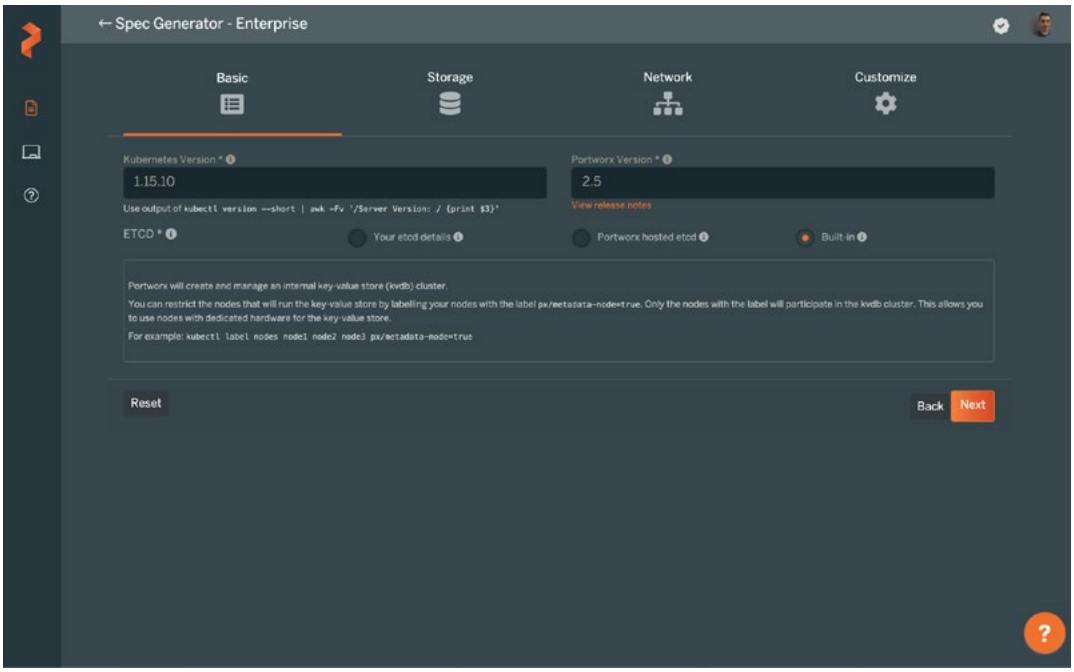
Verify the configuration.

```
kubectl get nodes
```

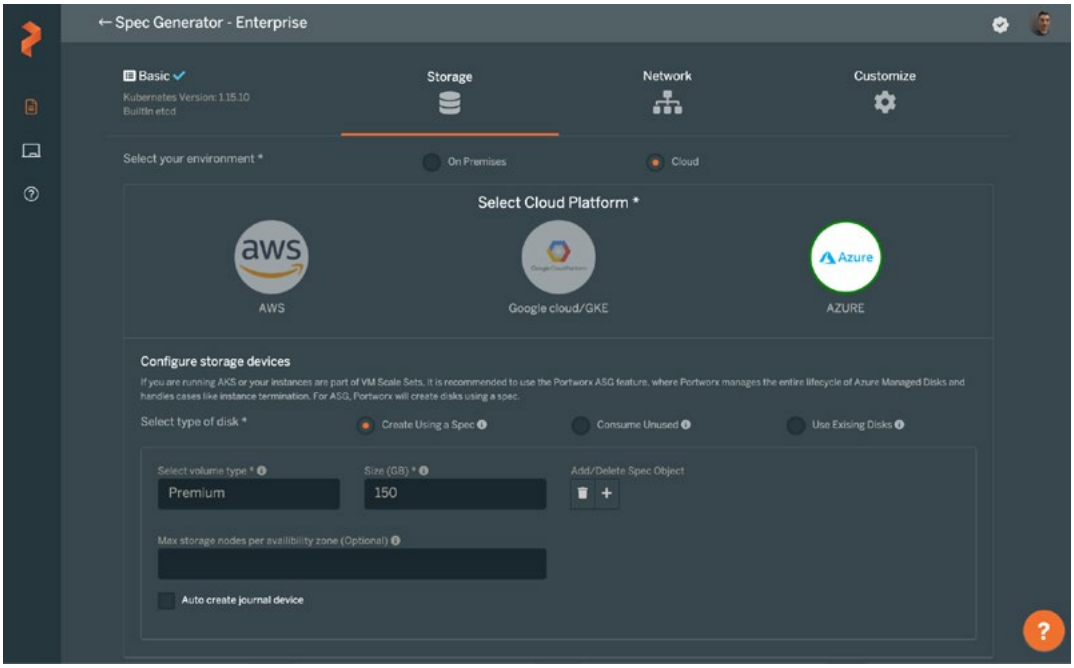
To allow Portworx to provision and attach Azure cloud volumes, a secret must be generated containing the information from the Service Principal created previously.

```
kubectl create secret generic -n kube-system px-azure
  --from-literal=AZURE_TENANT_ID=<tenant> \
  --from-literal=AZURE_CLIENT_ID=<appId> \
  --from-literal=AZURE_CLIENT_SECRET=<password>
```

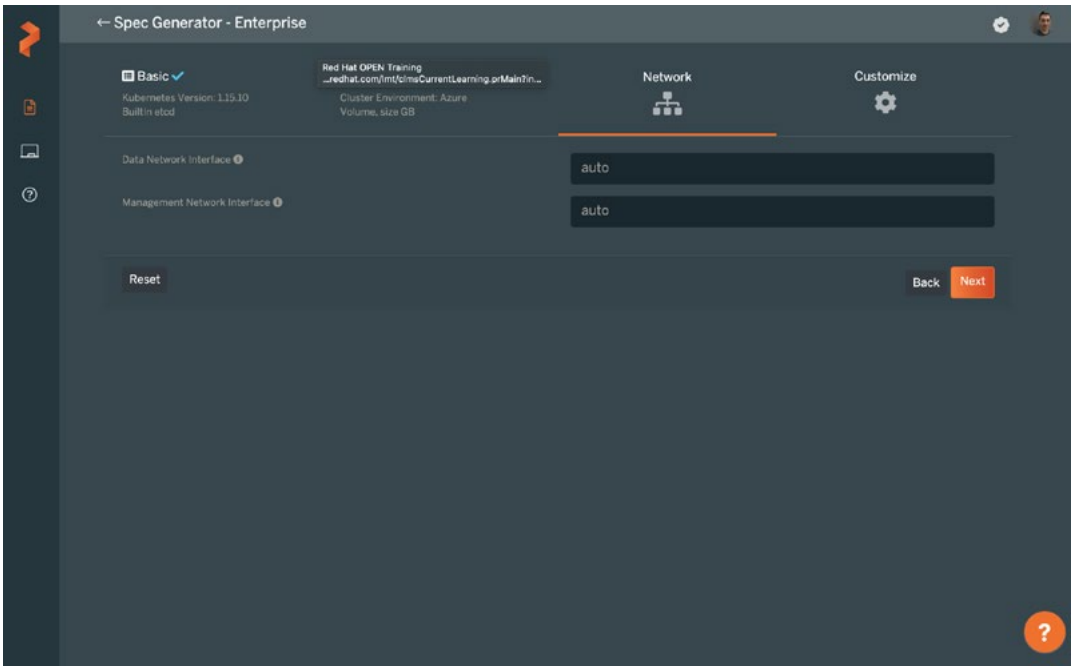
Now generate the Portworx install manifest. This can be done using the web-based tool available at https://central.portworx.com. After creating an account and accessing the tool, follow the steps below to generate an install spec.



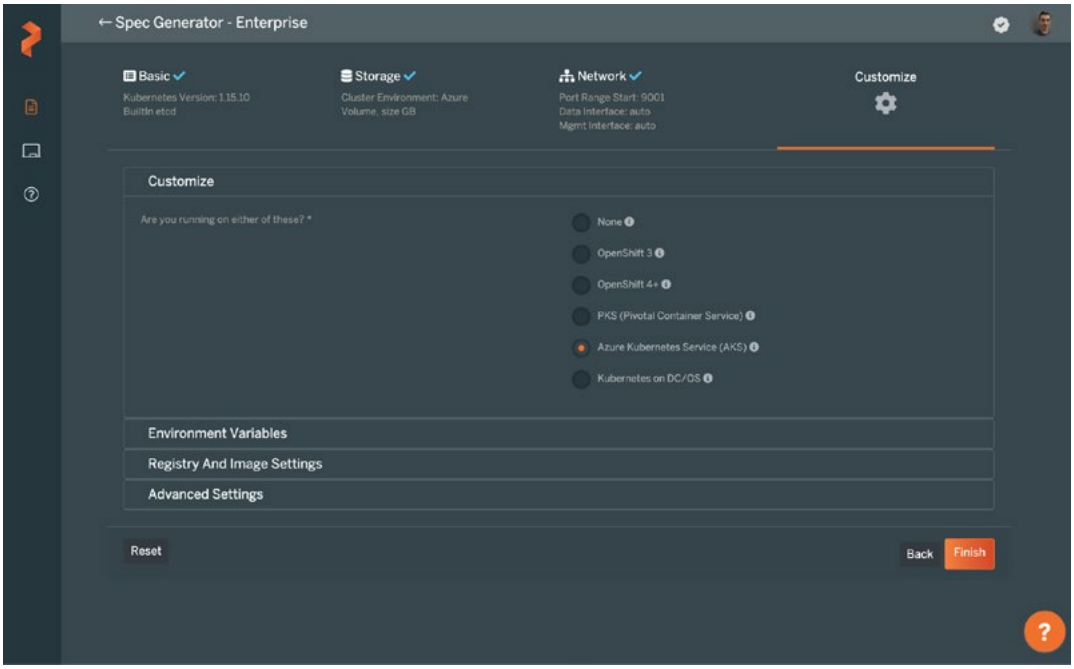

- **Kubernetes version** — Ensure this matches the AKS cluster.
- **Portworx version** — Choose the default unless you require a specific version.
- **ETCD** — For most deployments (up to 20 nodes), the in-built ETCD is sufficient. Portworx will deploy and manage an ETCD cluster across three workers—this can be controlled with labels. For larger deployments or IO intensive use cases, an external ETCD is recommended. More guidance is available on the Portworx documentation website: https://docs.portworx.com/reference/knowledge-base/etcd/.




- **Environment** — Ensure Cloud is selected with Azure as the cloud type.
- **Storage devices** — Select the storage required. This will be automatically provisioned by Portworx using the Service Principal previously created. Multiple disks can be added to create different tiers of storage. Performance information is available later in this document.
- **Auto create journal device** — Ensure this is selected, as it impacts performance.




- **Data and Management interface** — In most cases this can be left to auto, as Portworx will find and use the first routable interface. If a specific interface is required, it can be defined here.




- **Customize** — Ensure AKS is selected as an environment type.

Finish the spec generation to gain access to the installation manifest. It can then be applied on the AKS cluster as follows.

> Note: do not use this exact command, as it may not be suitable for your AKS environment.

```
kubectl apply -f
```

Wait for the Portworx pods to reach Running Status.

```
kubectl get pods -n kube-system -l name=portworx -o wide
```

You may notice devices created in the Azure Portal in a new resource group. Any device automatically provisioned by Portworx will be labeled appropriately.

| | |
|---|---|
| kubernetes | Load balancer |
| PX-DO-NOT-DELETE-0303536a-d7ee-48a6-9cea-5c307477d87b | Disk |
| PX-DO-NOT-DELETE-1c1d930e-1759-4fc7-b784-7b94283e6114 | Disk |
| PX-DO-NOT-DELETE-451e79ea-fbb8-4541-8f4f-6e22867f994f | Disk |
| PX-DO-NOT-DELETE-95f0c841-46e7-41ad-b699-5ee0c79449e6 | Disk |
| PX-DO-NOT-DELETE-d49b69b8-c6d4-42da-a431-5804a57a0827 | Disk |

# 6. ADDITIONAL CONFIGURATION

Some advanced configurations of Portworx require the use of the Portworx CLI - pxctl. This binary is automatically installed and added to the PATH on the worker nodes as part of a Portworx deployment. The CLI is also available inside a Portworx pod.

```
/opt/pwx/bin/pxctl
```

An alias can be set using kubectl exec.

If you are accessing remotely, we strongly suggest that you configure Portworx Security to enforce authorization when accessing the Portworx API. You can find additional information here: https://docs.portworx.com/concepts/authorization/install/.

## 6.1 Security

The Portworx Enterprise component PX-Security is a critical component of the Portworx platform that provides:

- Cluster-wide encryption.
- Namespace-granular or Storage-class BYOK encryption.
- Role-based access control (RBAC) for authorization, authentication, and ownership.
- Support for integration with AD and LDAP. Note that this integration is not available out of the box, but you can implement it through your OIDC solution.

Portworx provides namespace-granular, role-based authentication, authorization, and ownership in addition to encryption.

Portworx RBAC centers around the ubiquitous JWT-based authentication and authorization model. This technology is currently used by most major internet systems, providing a proven secure model for user and account identification.

A token is generated by a token authority (TA) and signed using either a private key or a shared secret. Then, the user should provide the token to Portworx for identification. No passwords are ever sent to Portworx.

This secure model enables Portworx to only have to verify the validity of the token to authenticate the user. Portworx then destroys the token, ensuring tokens are never saved on a Portworx system.

The token contains a section called claims, which not only identifies the user but also provides authorization information in the form of RBAC. Portworx uses the RBAC information to determine if the user is authorized to make the request.

To implement RBAC please refer to this documentation: https://docs.portworx.com/concepts/authorization/pre-install/.

## 6.2 Blob storage integration

Following the deployment of Portworx Enterprise into the AKS cluster, additional configuration can be carried out. Portworx supports integration with Azure Blob storage to be used as an external storage system for Portworx container volume snapshots.

Portworx volumes can be backed up to the cloud via pxctl cloudsnap. If you run this command with the '--help' flag, it shows the list of available operations for the full lifecycle management of your cloud backups.

Use the following command to set up credentials for accessing Azure Blob storage.

```
pxctl credentials create --provider azure --azure-account-name portworxtest
--azure-account-key zbJSSpOOWENBGHSY12ZLERJJV my-azure-cred
```

Further information about Cloudsnaps is available in the Portworx documentation: https://docs.portworx.com/reference/cli/cloud-snaps/.

## 6.2 Azure Key Vault integration

Portworx Enterprise provides the functionality to encrypt container volumes using a secret store in Azure Key Vault. Documentation for implementing this is available on the Portworx documentation website: https://docs.portworx.com/key-management/azure-kv/.

# 7. INFRASTRUCTURE CONSIDERATIONS

When deploying a Portworx Enterprise cluster into AKS, it is important to understand the performance requirements of your applications. When auto provisioning storage, using Portworx Premium SSD is recommended. It offers the following storage profile.

| Name | Scenario | Max Size | Max Throughput | Max IOPS |
|---|---|---|---|---|
| Premium SSD | Production and performance sensitive workloads | 32,767 GiB | 900 MiB/s | 20,000 |

This is a suitable storage device for most production workloads, although the device size must be taken into consideration when provisioning. Please refer to the following Azure documentation: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/disks-types#disk-size.

As an example, to get storage performance equivalent to a physical SSD device, a P50 or above Premium SSD device should be used.

For less performance-sensitive applications, a second class of Standard storage can be attached to workers. Portworx will create storage pools, allowing the user to specify which class of storage they wish to use as a backing device for the provisioned Portworx volume.

## 7.1 Storage pools

Storage pools are created by grouping together drives of the same size and same type. A single node with different drive sizes and/or types will have multiple pools. Within a pool, by default, the drives are written to in a RAID-0 configuration. You can configure RAID-10 for pools with at least four drives. There can be up to 32 pools within the same node.

At the time of pool construction, individual drives are benchmarked and are categorized as high, medium, or low based on random/sequential IOPS and latencies. These are applied as individual labels on the pools and can be used in provisioning rules. For example, a provision rule can be written to provision volumes on pools that have random I/O latencies less than 2 ms or io_priority high.

Storage pools are continuously monitored for health using smartctl utilities. Alerts are raised for discrepancies.

# 8. BENCHMARKING

The following benchmarks were carried out using FIO (https://fio.readthedocs.io/en/latest/fio_doc.html). Please use this as a reference point only for an indication of the performance you can expect in Azure when using Portworx.

## 8.1 FIO parameters

**ioengine** — Linux native asynchronous IO engine (libaio)

**blocksize** — This is the block size used for each IO operation and varies from application to application.

**readwrite** — IO pattern (read/write sequential/random)

**size** — This is the dataset size. Ideally, it should be greater than the size of host cache to avoid any caching effects.

**direct** — Set to true for non-buffered IO. This implies files are opened with the O_DIRECT flag, which results in IOs bypassing host cache.

**iodepth** — This is the number of outstanding/queued IO requests in the system. A higher number typically means greater concurrency and better resource utilization.

**end_fsync** — This causes flushing of dirty data to disk at the end of the test. The flush time is included in the measurement.

**time_based** — Run test for fixed amount of time

**runtime** — Number of seconds that the test runs

**ramp_time** — Number of seconds before performance measurement is logged

## 8.2 System configuration

| | | |
|---|---|---|
| Instance | DS3 v2 | 4 vcpu x 14G<br>12800 IOPS / 192 MB/s throughput<br>3000 Mbps Network Bandwidth |

## 8.3 Portworx configuration

Two storage pools are created—one with a single P50 disk and the other with 2xP30. Performance is measured on EXT4 on top of two Portworx volumes, one in each pool.

## 8.4 Monitoring FIO runs

`iostat -xm 1` is the ideal way to monitor disk IO.
`top` can provide a way to monitor CPU and memory consumption.

## 8.5 Sequential and random write performance

Measuring sustained sequential write performance ensures that the amount of data that is written is more than the available RAM to offset the effects of Linux and storage caches. The key parameter here is `--size=2xRAM` in addition to direct-io.

```
```fio  --blocksize=16k --name=/mnt/perf.data --ioengine=libaio --readwrite=randwrite
--size=16G --direct=1 --invalidate=1 --iodepth=128```
```


```
```fio  --blocksize=16k --name=/mnt/perf.data --ioengine=libaio --readwrite=randwrite
--size=16G --direct=1 --invalidate=1 --iodepth=128```
```


## 8.6 Sequential and random read performance

Linux page cache should be evicted before doing a read test. In addition, the backing filesystem or block device should be prepopulated. The '`--readonly`' flag ensures that new data is not written, making it safe to baseline root block devices as well.

It is sufficient to do a time-based test for 300 seconds; a ramp up time of 30 seconds helps measure performance in a steady state.

```
```
io  --blocksize=16k --name=/mnt/perf.data --ioengine=libaio --readwrite=read
--size=16G --direct=1 --invalidate=1 --iodepth=128 --time_based --runtime=300
--ramp_time=30 --readonly
```
```

```
fio  --blocksize=16k --name=/mnt/perf.data --ioengine=libaio --readwrite=randread
--size=16G --direct=1 --invalidate=1 --iodepth=128 --time _ based --runtime=300
--ramp _ time=30 --readonly
```

## 8.7 Block device performance

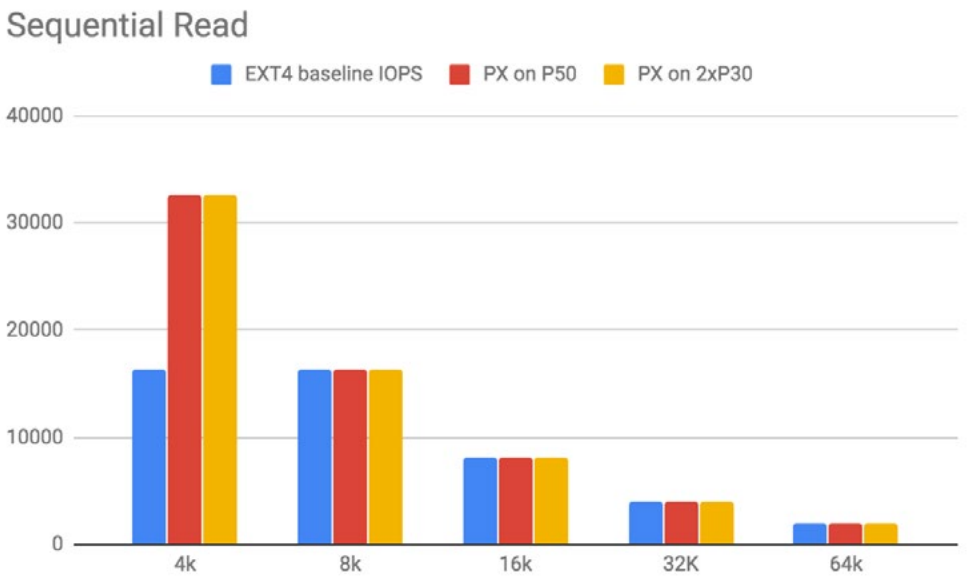
These benchmarks test the virtual Portworx block device.

### 8.7.1 Results

Baseline measurement is done by measuring local performance on EXT4 over a P50 device.

**Sequential read**

| Block Size | EXT4 baseline IOPS | PX on P50 | PX on 2xP30 |
|---|---|---|---|
| 4k | 16315 | 32622 | 32571 |
| 8k | 16322 | 16318 | 16293 |
| 16k | 8154 | 8159 | 8157 |
| 32K | 4079 | 4079 | 4077 |
| 64k | 2039 | 2039 | 2038 |



**Sequential write**

| Block Size | EXT4 baseline IOPS | PX on P50 | PX on 2xP30 |
|---|---|---|---|
| 4k | 16161 | 19335 | 31223 |
| 8k | 15405 | 15356 | 16164 |
| 16k | 7708 | 7725 | 8186 |
| 32K | 3851 | 3475 | 4056 |
| 64k | 1926 | 1814 | 2032 |

## 8.8 Shared volume: multi reader, multi writer

The following tests shared volume or Read Write Many Portworx volume performance.

In contrast to block volumes, shared volumes allow concurrent access to a Portworx volume from multiple nodes at the same time. The example above shows WordPress containers accessing a shared file system that is mounted on a storage node that hosts data for the volume.

The principles of a shared volume are the same as that of a block volume. A block device is attached on one of the nodes that hosts a replica for the volume, and this node is designated as the transaction coordinator. User containers can run on any host in the cluster; several containers accessing the same volume can run concurrently across the cluster. All the I/O for user containers is routed to the Portworx node where the volume is attached. In the event of a failure of a Portworx replica, another replica is selected as the transaction coordinator. The clients seamlessly switch over to another replica.

The performance of shared volumes is similar to that of NFS.

The disk type chosen is P70.

### 8.8.1 Baseline configuration

Baseline is NFS for shared volumes. EXT4 over base block devices are exported over NFS, and performance is measured from a remote client.

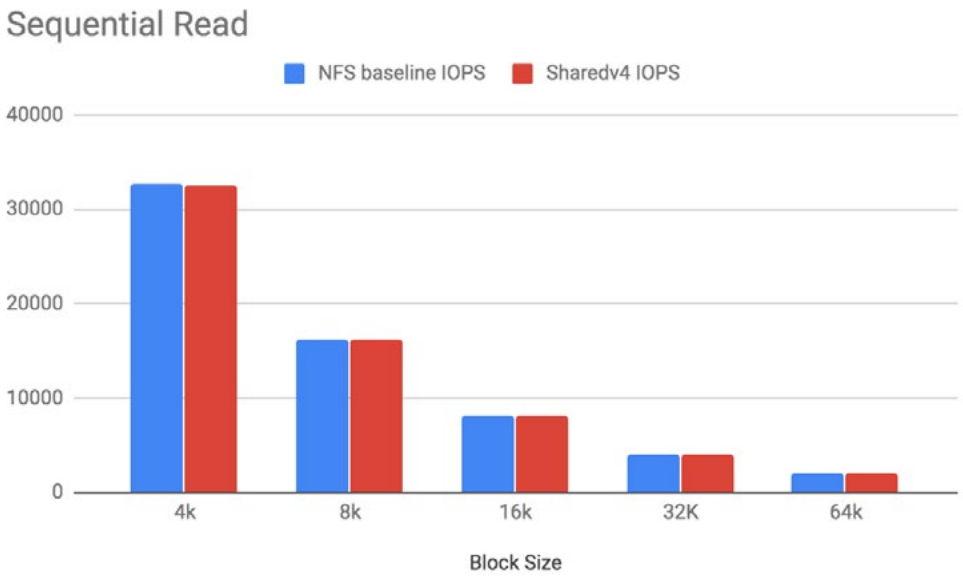
### 8.8.2 Portworx configuration

Sharedv4 Volume is created and attached over a remote network.

### 8.8.3 Results

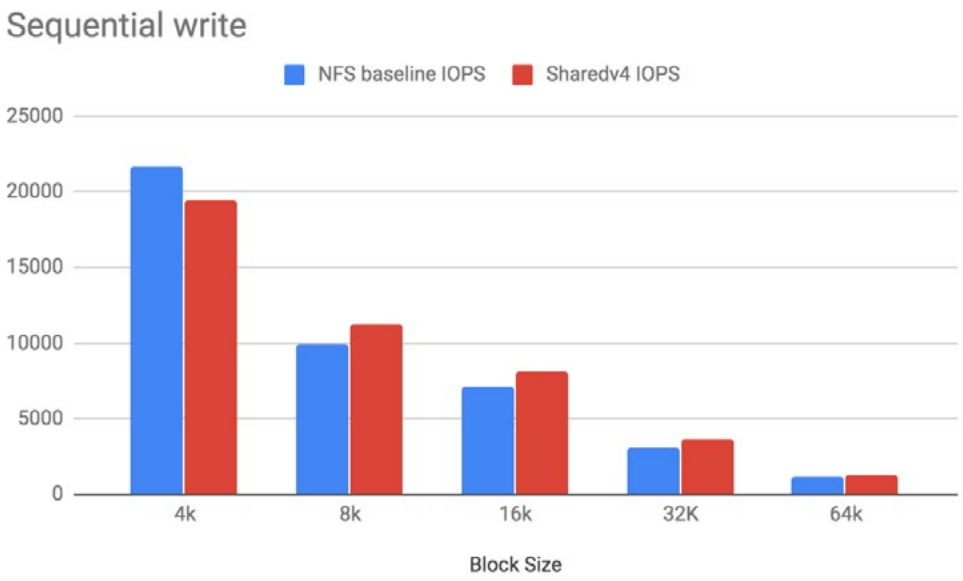
**Sequential read**

| Block Size | NFS baseline IOPS | Sharedv4 IOPS |
|---|---|---|
| 4k | 32700 | 32600 |
| 8k | 16300 | 16300 |
| 16k | 8161 | 8173 |
| 32K | 4079 | 4073 |
| 64k | 2040 | 2037 |



**Sequential write**

| Block Size | NFS baseline IOPS | Sharedv4 IOPS |
|---|---|---|
| 4k | 21700 | 19400 |
| 8k | 9858 | 11300 |
| 16k | 7056 | 8137 |
| 32K | 3073 | 3685 |
| 64k | 1142 | 1241 |




**Portworx, Inc.**
4940 El Camino Real, Suite 200, Los Altos, CA 94022
Tel: 650-241-3222 | info@portworx.com | www.portworx.com