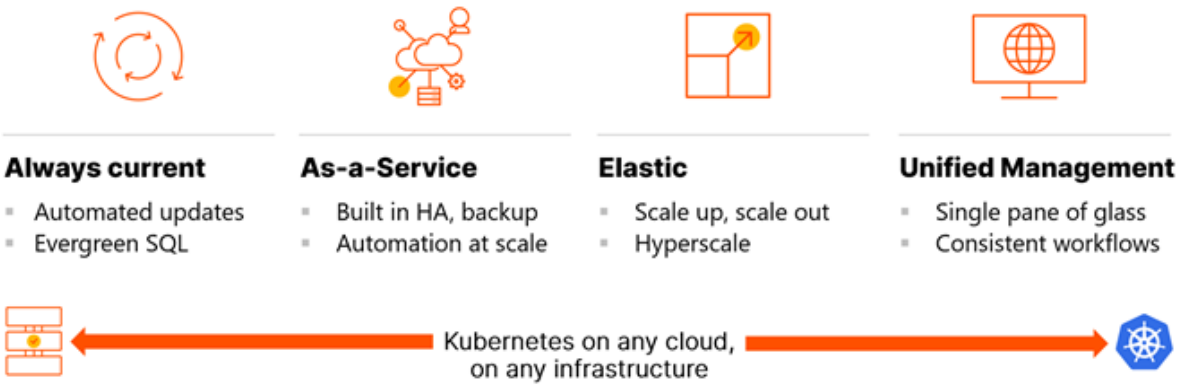
PURE VALIDATED DESIGN

Portworx with Azure Arc-enabled Data Services for Stateful Kubernetes Deployments

Enable HA, DR, backup, and data security for Azure data services on any Kubernetes-enabled cloud or infrastructure.

Contents



Executive Summary

As modern applications running on Kubernetes increasingly move from stateless to stateful, developers need to maintain access to Azure Arc-enabled data services like SQL Server, Postgres, and more. The traditional, legacy approach to managing these databases relies on scripting combined with a significant number of highly repetitive and manual activities to ensure databases remain online and highly available in support of mission-critical applications. These highly manual activities are prone to human error, excessive downtime, and can't be easily automated with existing infrastructure automation tools.

While Kubernetes provides some support for such automation, teams need an enterprise-grade Kubernetes data services platform to ensure the reliability and performance of these databases in production at scale.

Microsoft Azure Arc-enabled data services bring Azure data services to any cloud or infrastructure that can host Kubernetes clusters. This Pure Validated Design and Azure Arc validated solution outlines how Portworx® can provide a compelling Azure data service experience for mission-critical environments. To achieve this, the Kubernetes cluster storage solution needs to provide the following services and qualities:

- **Consistent Data Plane:** In the same way that Azure Arc-enabled data services can run on Kubernetes in any cloud and on any infrastructure, the Kubernetes storage solution data plane requires the ability to underpin any Kubernetes cluster in any cloud and on any infrastructure.
- **Elastic Scaling:** Because a key feature of Azure Arc-enabled data services is the ability to provide elastic scaling, the Kubernetes storage solution needs to support this by allowing thousands of persistent volumes to be associated with each worker node and for effortless management of persistent volumes at scale.
- **Performance:** The storage solution should provide mission-critical levels of performance consistently as the platform is scaled out.
- **Security:** Because cybercrime is rife, sensitive data needs to be protected against exfiltration and ransomware attacks.
- **High Availability:** Databases need to be highly available across worker nodes, different availability zones for cloud-based Kubernetes clusters, as well as clusters in different on-premises data centers.
- **Disaster Recovery:** Data needs to be protected at various levels of granularity in a Kubernetes cluster, ranging from namespace level down to object level.

The Portworx Kubernetes Data Services Platform provides enterprises with a Kubernetes-native storage solution that incorporates all these qualities and more. Portworx provides a fully integrated solution for persistent storage, data protection, disaster recovery, data security, cross-cloud and data migrations, and automated capacity management for applications running on Kubernetes. Administrators can leverage this consistent management framework, to manage the entire lifecycle of an Azure Arc-enabled data services, from automating day-2 operations like scaling and migration to delivering self-service database access to developers.

Microsoft Azure Arc-enabled Data Services

Azure Arc-enabled data services contain several layers:

- **Kubernetes:** The infrastructure that the solution runs on top of is Kubernetes-based. Because data platforms by their very nature are stateful, the Kubernetes cluster requires a storage solution irrespective of where it is hosted.
- **Control Plane Layer:** The Azure cloud is extended to the Kubernetes cluster via the Azure Arc data controller. More specifically, the controller extends the Azure Resource Manager (ARM) to the Kubernetes cluster. It also provides data services resource management and is how logging and telemetry data are delivered to Azure. Once a Kubernetes cluster and its associated infrastructure is in place, the very first step in deploying Azure Arc-enabled data services is to deploy a controller to the Kubernetes cluster.
- **Data Services Layer:** This layer includes Azure Arc-enabled SQL Server Managed instances and/or PostgreSQL Hyperscale instances, or more simply put the database services to be consumed.
- **Cloud Provisioning/Monitoring Layer:** Each controller has an associated Azure region. Metrics and logging are uploaded to the Azure region automatically for controllers deployed in 'Connected' mode and manually via the command line tool for controllers deployed in 'indirectly connected mode'. The cloud-based element of the solution stack also provides the capabilities to monitor Azure Arc-enabled data services and provision both data services and controllers.

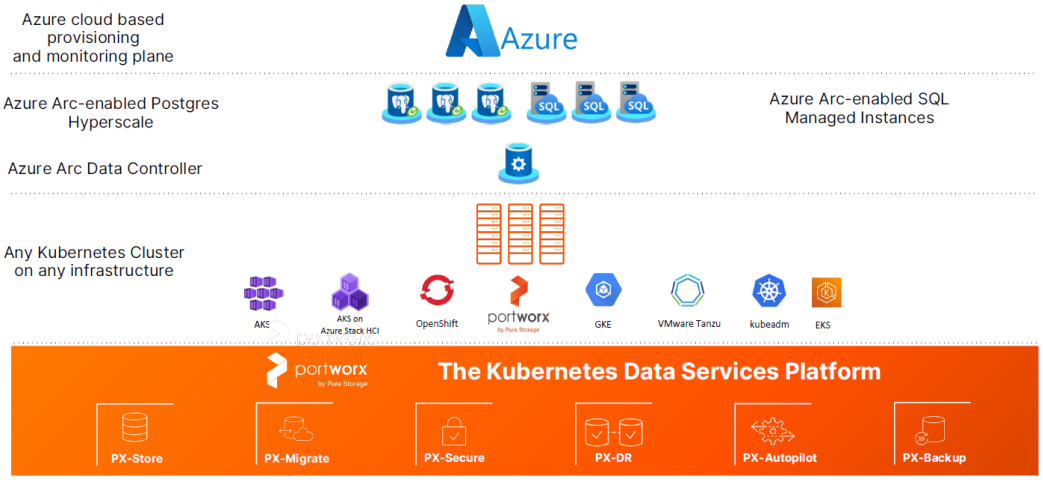


Figure 1: Azure Arc-enabled data service layers.

Tooling

You can deploy Azure Arc-enabled data services in a variety of ways:

- **Azure CLI Extensions:** Azure Arc-enabled data services extensions provided in python wheel format (a python packaging format) can be plugged into Azure CLI. At the time of writing an Azure CLI extension is provided for SQL Servers.
- **Azure Data Studio:** Azure Data Studio is a graphical user interface tool that comes with the built-in capability to run notebooks. It currently enables both controller and data service deployment.
- **Azure Portal:** Controllers and data services can both be deployed from the Azure marketplace. It can also be used to provide a single pane of glass across all Azure Arc managed resources.
- **Kubernetes Native Tools:** Both controllers and data services can be deployed via YAML manifests using kubectl.

Irrespective of the tool used to deploy Azure Arc-enabled data services, the touchpoint for storage is the Kubernetes storage class object.

Portworx

Portworx is a data management solution that serves applications and deployments in Kubernetes clusters. Portworx is deployed natively within Kubernetes and extends the automation capabilities down into the infrastructure to eliminate all the complexities of managing data. Portworx provides simple and easy to consume storage classes that are usable by stateful applications in a Kubernetes cluster.

At the core of Portworx is PX-Store, a software-defined storage platform that works on practically any infrastructure, regardless of whether it is in a public cloud or on-premises. PX-Store is complemented by:

- **PX-Migrate:** Allows applications to be easily migrated across clusters, racks, and clouds.
- **PX-Secure:** Provides access controls and enables data encryption at a cluster, namespace, or persistent volume level.

- **PX-DR:** A service that allows applications to have a zero RPO failover across data centers in a metro area as well as continuous backups across the WAN for even greater protection.
- **PX-Backup:** A solution that allows enterprises to back up and restore the entire Kubernetes application, including data, app configuration, and Kubernetes objects, to any backup location—including NFS, S3, or Azure Blob—with the click of a button.
- **PX-Autopilot:** A service that provides rules-based auto-scaling for persistent volumes and automatic expansion and re-balancing of Portworx storage pools.

This paper will review each element of Portworx in greater detail and highlight the value each specific service provides to Azure Arc-enabled data services.

PX-Store

PX-Store is a 100% software-defined storage solution that provides high levels of persistent volume density per block device per worker node. The key features of PX-Store include:

Storage Virtualization: The storage made available to each worker node is effectively virtualized such that each worker node can host pods that use up to hundreds of thousands of persistent volumes per Kubernetes cluster. This benefits Kubernetes clusters deployed to the cloud, in that larger volumes or disks are often conducive to better performance.

Storage-aware Scheduling: Stork, a storage-aware scheduler, collocates pods on worker nodes that host the persistent volume replicas associated with the same pods, resulting in reduced storage access latency.

Storage Pooling for Performance-based Quality-of-Service: PX-Store segregates storage into three distinct pools of storage based on performance: low, medium, and high. Applications, including Azure Arc-enabled data services, can select storage based on performance by specifying one of these pools at the storage class level.

- **Persistent Volume Replicas:** You can specify a persistent volume replication factor at the storage class level. This enables the state to be highly available across the cluster, cloud regions, and Kubernetes-as-a-service platforms such as AKS, EKS, and GKE.
- **Cloud Volumes:** Cloud volumes enable storage to be provisioned from the underlying platform without the need to present storage to worker nodes. PX-Store running on most public cloud providers and VMware have cloud volume capability, a feature introduced to FlashArray™ and FlashBlade® in the 2.8 release of Portworx.
- **Automatic I/O Path Tuning:** Portworx provides different I/O profiles for storage optimization based on the I/O traffic pattern. By default, Portworx automatically applies the most appropriate I/O profile for the data patterns it sees. It does this by continuously analyzing the I/O pattern of traffic in the background.
- **Metadata Caching:** High-performance devices can be assigned the role of Journal devices to lower I/O latency when accessing metadata.
- **Read and Write-through Caching:** PX-Cache-enabled high-performance devices can be used for read and write-through caching to enhance performance.

PX-Backup

Backup is essential for enterprise applications, serving as a core requirement for mission-critical production workloads. The risk to the enterprise is magnified for applications on Kubernetes where traditional, virtual machine (VM)-optimized data protection solutions simply don't work. Protecting stateful applications like databases in highly dynamic environments calls for a purpose-built, Kubernetes-native backup solution.

Portworx PX-Backup solves these shortfalls and protects your applications' data, application configuration, and Kubernetes objects with a single click at the Kubernetes pod, namespace, or cluster level. Enabling application-aware zero data loss backup and fast recovery for even complex distributed applications, PX-Backup delivers true multicloud availability with key features including:

- **App-Consistent Backup and Restore:** Easily protect and recover applications regardless of how they are initially deployed on, or rescheduled by, Kubernetes.
- **Seamless Migration:** Move a single Kubernetes application or an entire namespace between clusters in Azure.
- **Compliance Management:** Manage and enforce compliance and governance responsibilities with a single pane of glass for all your containerized applications.
- **Streamlined Storage Integration:** Back up and recover cloud volumes with storage providers including Amazon EBS, Google Persistent Disk, Azure Managed Disks, and CSI-enabled storage.

PX-DR

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 provides both synchronous and asynchronous replication, delivering key benefits including:

- **Zero Data Loss Disaster Recovery:** PX-DR delivers zero RPO failover across data centers in metropolitan areas in addition to HA within a single data center. You can deploy applications between clouds in the same region and ensure application survivability.
- **Continuous Global Backup:** For applications that span a country, or across the entire world, PX-DR also offers constant incremental backups to protect your mission-critical applications.

PX-Autopilot

PX-Autopilot allows enterprises to automate storage management to Intelligently provision cloud storage only when needed and eliminate the problem of paying for storage when over-provisioned:

- **Storage Capacity Growth On-demand:** Automate your applications' growing storage demands while also minimizing disruptions. Set growth policies to automate cloud drive and Kubernetes integration to ensure your application's storage needs are met without performance or availability degradations.
- **Slash Storage Costs by Half:** Intelligently provision cloud storage only when needed and eliminate the problem of paying for storage when over-provisioned instead of consumed. Scale at the individual volume or entire cluster level to save money and avoid application outages.

- **Integrate with All Major Clouds and VMware:** PX-Autopilot natively integrates with AWS, Azure, Google, as well as VMware enabling you to achieve savings, and increase automated agility across all your clouds.

Architecture

All Portworx products run on the same Kubernetes cluster as the applications that they provide service to. The core storage services of the Portworx platform are provided by PX-Store, which has two main components:

- **Control Plane:** The control plane exposes a REST API that can either be used by the Portworx native CLI, pxctl, or the OpenStorage SDK that can be leveraged either by Go or Python. It also monitors the health of the cluster via the use of a Gossip protocol. In addition to this, the control plane exposes a Prometheus exporter endpoint—which benchmarks available storage—to determine what performance-based pool devices should be allocated to. It also gathers metrics from the data plane.
- **Data Plane:** The data plane is responsible for all I/O, I/O caching, making data highly available via persistent volume replicas, and the encryption of data at rest.

PX-Store is deployed through a Portworx specification or spec for short. Depending on the Kubernetes distribution or platform, the specification can either be deployed via a Kubernetes Operator or DaemonSet YAML manifest. Worker nodes in a Kubernetes cluster that are part of the data plane are referred to as Portworx storage nodes.

Portworx requires a Kubernetes cluster with at least three worker nodes, each with the ability to mount block storage or NFS file shares, and this document will focus on block storage. Metadata is stored in etcd, commonly referred to as a 'KVDB' (key-value database) in Portworx documentation. Control plane nodes are usually separate from the worker nodes, except for Kubernetes clusters built on bare metal, such as a Red Hat OpenShift three-node edge configuration, whereby the physical nodes act as both control plane and worker nodes.

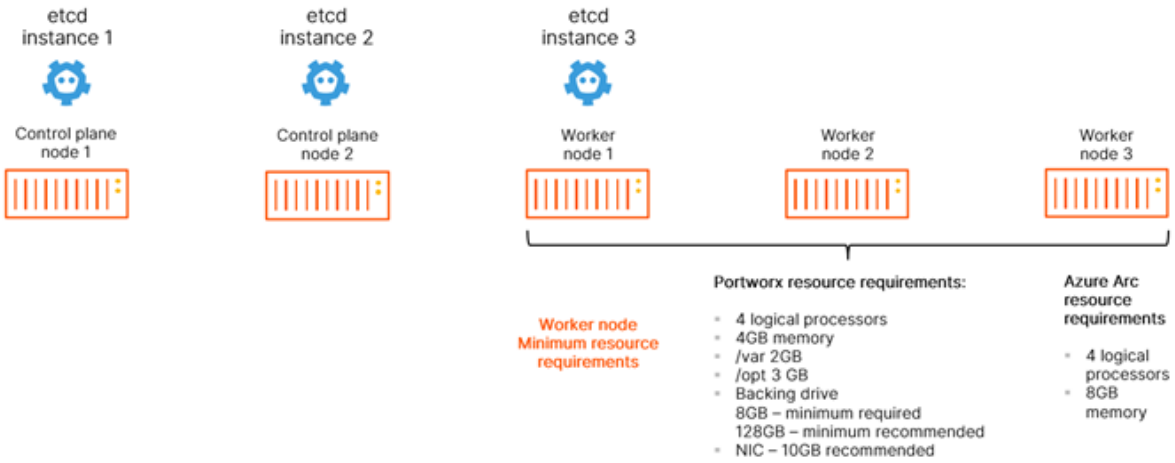


Figure 2: Etcd, control and worker nodes.

Each worker node requires a minimum of eight logical processors and 12GB of memory (Figure 2). Portworx can be deployed to virtually all the popular on-premises Kubernetes distributions and most of the popular Kubernetes-as-a-service platforms in the public cloud. This document will not go into the specifics of any specific distribution; however, it will make a distinction between Kubernetes on-premises and Kubernetes-as-a-service in the public cloud. An important distinction is that with

Portworx deployed to the public cloud, there must be at least two worker nodes per availability zone for a worker node to be considered for use as a Portworx storage node in any given availability zone.

Deployment Options

When creating a specification to deploy Portworx with, you have several options to consider:

- **Use an Existing KVDB:** For most deployments, you can create a deployment specification with the option of storing Portworx metadata in a separate etcd cluster. There are two exceptions to this:
  - The first scenario: When the PX-DR is used for Kubernetes clusters that are not within the same metro area, meaning the network round-trip latency between the primary and disaster recovery sites is greater than 10ms.
  - The second scenario in which a dedicated etcd cluster should be used is for large-scale deployment, with 10 or more worker nodes, in which a heavy dynamic provisioning activity takes place.
- **Dedicated Journal Device:** A dedicated Journal device can be specified to buffer metadata writes.
- **Dedicated Cache Device:** A dedicated cache device can be specified to improve performance by acting as a read/write-through cache.
- **Container Storage Interface (CSI) API Compatibility:** You can choose the option to deploy Portworx with CSI enabled if PX-Security is to be used.
- **Stork:** Stork is a storage-aware scheduler that attempts to collocate application pods onto the same nodes as the persistent volumes and persistent volume replicas that it uses. Use Stork if your underlying infrastructure uses either servers with dedicated internal storage or servers with dedicated network-attached storage appliances.
- **Dedicated Network:** Consider using a dedicated network for storage cluster traffic if the existing network infrastructure does not support quality-of-service.

Leveraging VMware

Portworx is infrastructure-agnostic in that it can be deployed to virtually any infrastructure either on-premises or in the public cloud. Despite this, VMware is worth highlighting because it is ubiquitous with most on-premises data centers that run Microsoft software. When creating a Portworx spec, selecting the cloud deployment option provides the ability to deploy Portworx so that persistent volumes are created inside VMware virtual disks inside of VMFS datastores.

Planning, Design, and Prework

This section of the document will cover detailed requirements and preparation before deployment. Prework of the vSphere Hosts and Environment must be carefully reviewed with preplanning the network requirements for a successful deployment.

vSphere Hosts

The solution is based on four ESXi 7.0.2 hosts with the following individual specifications:

- Dual socket, 12 cores per socket
- 512GB of physical memory
- 2 Fibre Channel HBA

- 2 10 GbE Dual port network controllers
  - Vmnic0: Assigned to vSwitch for ESXi host mgmt and VMKernel ports
  - Vmnic1: Assigned to vSwitch for ESXi host mgmt and VMKernel ports
  - Vmnic2: Available for distributed vSwitch Uplink - VLAN trunking configured
  - Vmnic3: Available for distributed vSwitch Uplink - VLAN trunking configured

The required physical server type is commodity x86-64 architecture with minimum specs based on VMware hardware requirements. For vSphere 7.0 ESXi requirements, see the ESXi Hardware Requirements or VMware Compatibility Guide.

vSphere Environment

The vSphere environment for this solution must meet all the following parameters before deploying the HAProxy Load Balancer and enabling Workload Management. This document will showcase the vCenter Server User interface to accomplish required tasks to manage and operate vSphere with Tanzu.

- vSphere cluster with three ESXi hosts (minimum)
  - Minimum: version 7.0.2
  - Minimum of one FlashArray//X VMFS datastore
  - Best practice: Hostnames using all lowercase to eliminate deployment issues
- vCenter Server Appliance
  - Minimum: version 7.0.2
  - Running on FlashArray VMFS datastore
- HA and DRS must be enabled for *tanzu-cluster*
  - DRS should be set to Fully Automated
- vSphere distributed switch
  - Minimum: Version 7
  - All ESXi hosts connected
  - Two port groups configured as “management” and “workload”
- vMotion network
  - Best practice: Dedicated VLAN for production workloads
- NTP configuration must be the same across ESXi hosts, vCenter Server, and HAProxy.
  - Synced to an external time source.

NOTE: vSphere 7.0+ includes a 60-Day trial of vSphere with Tanzu Basic.

Network Requirements

This deployment is based on two routable VLAN networks (Figure 3) and a single vSphere Distributed Switch with two port groups:

- The Management network supports ESXi hosts, vCenter Server, FlashArray, and operational management.
- The Workload network supports the Supervisor Cluster and Kubernetes workloads which includes access to HAProxy services.

NOTE: In addition to this network topology, you can also use NSX-T based or NSX ALB based network topologies.

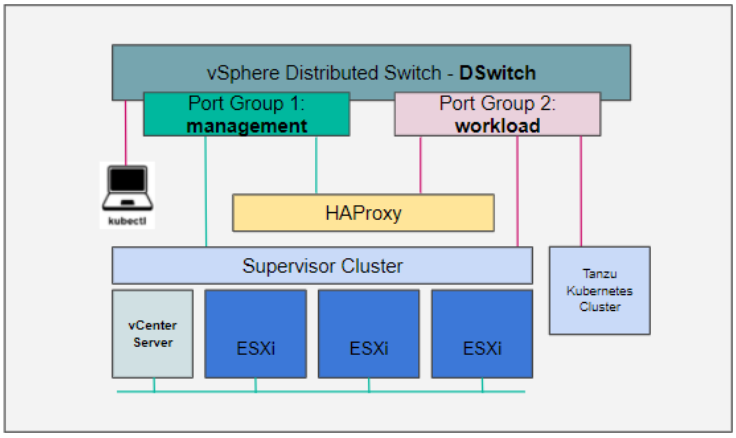


Figure 3: Virtual network

- Management network - vmnic0 [Example: VLAN111 w/ 192.168.111.0/24]
  - Must be routable to ESXi hosts, vCenter Server, the Supervisor Cluster, and HAProxy
  - DNS query enabled
  - Must have internet connectivity
- Workload Network - vmnic1 (Example: VLAN114 w/ 192.168.114.0/24)
  - DNS query enabled
- vSphere distributed switch
  - *Management* port group configured to appropriate VLAN ID
  - Workload port group configured to appropriate VLAN ID

NOTE: HAProxy Load Balancer requires assignment of static IP Addresses. This deployment cannot be based on DHCP.

- Management network
  - HAProxy Management - IP Address (Example: 192.168.111.159/24)
    - Default Gateway must be on this network
    - Dataplane management port 5556 (default), which is used for Step 5 of enabling Workload Management. (Example: 192.168.111.159:5556)
  - Supervisor Control Plane - Five IP Addresses (Example: 192.168.111.160–192.168.111.164)
    - The first IP address will be used to enable Workload Management, Step 6: Starting IP Address.

- Workload network
  - Workload IP address (Example: 192.168.114.7/24, must be outside of LB IP Range)
    - This IP address will be used and entered in CIDR format during the HAProxy installation, Step 9: Customize Template section 2.5.
  - Load balancer IP ranges (CIDR) (Example: 192.168.114.32/27; these are used for the TKC)
    - 10.21.114.32/27 CIDR Range = 192.168.114.32–192.168.114.63
    - HAProxy Installation Wizard Step 9, Section 3.1
  - Workload IP Range - Virtual Machines (Example: 192.168.114.96-10.21.114.127; must be outside of LB IP Range)
    - This will be used to enable Workload Management, see adding workload network, Step 7.

For more information, access extensive networking overview and official documentation.

Network Worksheet

HAProxy	Management	Workload	Notes
Management IP	_____._____._____._____		root
SControlPlane VM-1	_____._____._____._____		Start
SControlPlane VM-2	_____._____._____._____		
SControlPlane VM-3	_____._____._____._____		
SControlPlane VM-4	_____._____._____._____		
SControlPlane VM-5	_____._____._____._____		End
Workload IP		_____._____._____._____	
Load Balancer IP Range of 30 IPs (min)		_____._____._____._____/	CIDR
Virtual Machines Range of 30 IPs (min)		_____._____._____._____ _____._____._____._____	Out of LB range
NTP	_____._____._____._____		
DNS	_____._____._____._____		

See the CIDR conversion tool.

FlashArray

Deployment tasks in this documentation are based on the vSphere Client and the Pure Storage® Plugin for the vSphere Client.

The FlashArray is authenticated with the Pure Storage Plugin for the vSphere Client (referred to in the rest of this document as the vSphere Plugin) for ease of use. There are multiple methods to install the vSphere Plugin, such as PowerShell, vRealize Orchestrator, and the FlashArray Web UI as documented in our support guide.

- Required: ESXi hosts' Personality must be set to "ESXi"
- Purity Version 5.3.10 or later

FlashArray vVols

The following list is a set of best practices and tasks that must be completed before the creation and mounting of the vVols Datastore via vCenter:

- Required: Network port 8084 must be open and accessible from vCenter Server and ESXi hosts to the FlashArray.
- The vCenter Server should never run on vVols. The two are interdependent. The best practice is to run vCenter on VMFS.
- Best practice: Create a dedicated local FlashArray User for VASA registration. This prevents authentication issues in the event Active Directory/LDAP service is unreachable or the case of a *user account* of the storage provider being deleted/removed.

Access Best Practices Quick Guidance Points from Pure Support.

VMware HAProxy

The VMware HAProxy Appliance must be downloaded and uploaded to the vCenter Content Library.

- Download and upload to vCenter Content Library
- Requires an FQDN and static IP address for direct management on the *management* network and a static IP address on the *workload* network
- HAProxy v0.2.0

Managing the Content Library

The vSphere Content Library consists of a local HAProxy library containing the OVA and a subscription Tanzu Kubernetes Grid library for deployment. The HAProxy OVA must be uploaded on a Datastore accessible by the tanzu-cluster.

- Tanzu Kubernetes Grid (Subscription content library).
- HAProxy (Local content library). You must Import Library Item: haproxy-v0.2.0.ova

To create vSphere Content Libraries, navigate to the **Menu** drop down from vSphere Client and select **Content Libraries** (Figure 4).

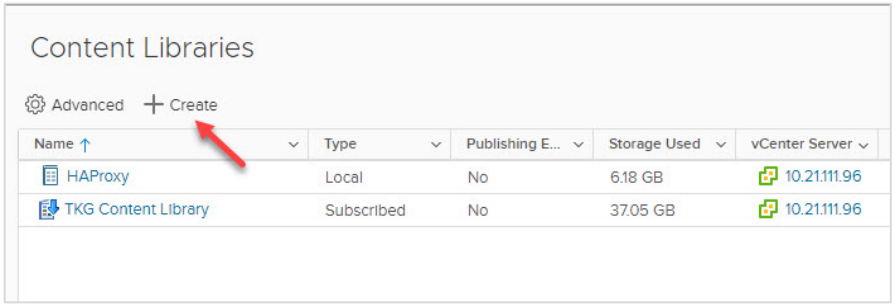


Figure 4: Content Library

Deployment and Application Validation

This section describes the deployment of vSphere Workload Management, Namespaces, Tanzu Kubernetes Clusters, and Persistent Volume Claims with FlashArray vVols Datastore. It provides guidelines for installing and configuring HAProxy, the Workload Management environment, and deploying a Tanzu Kubernetes Cluster.

Once the cluster is deployed, the guide will then step through implementing Portworx Enterprise Data Platform to manage persistent storage operations using vVol-backed vSphere CSI volumes. This brings enterprise-class data services to the Azure Arc and Tanzu platforms that are on par with the service levels that are established standards in traditional environments.

Solution validation is an operational deployment of Azure Arc-enabled data services with Portworx Enterprise providing persistent storage.

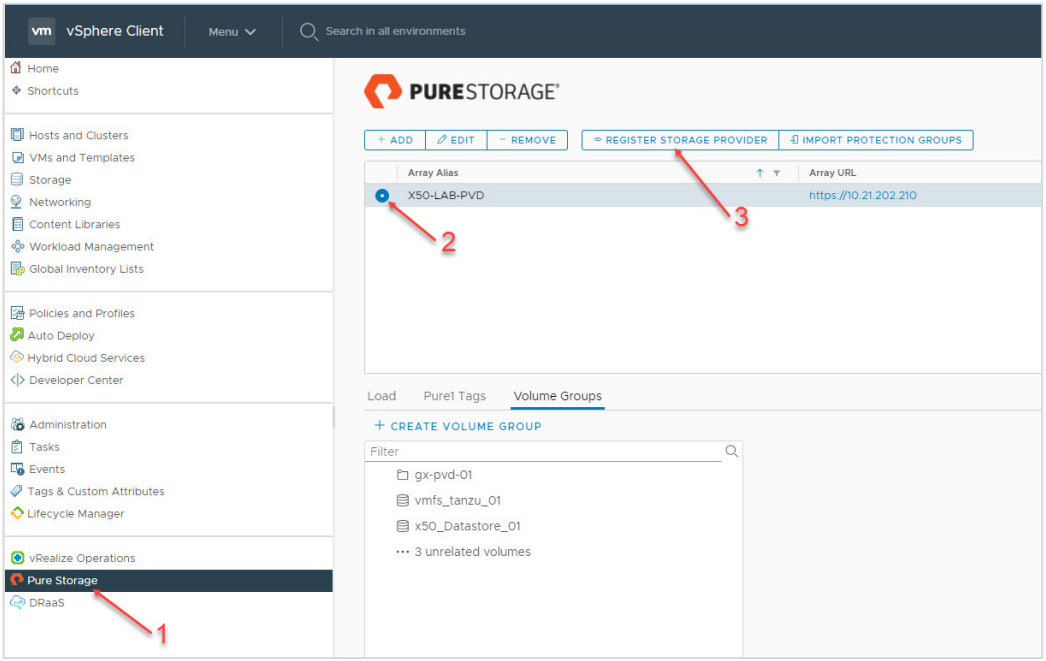
Deployment

You should install VMware vSphere with Tanzu on dedicated physical industry-standard x86 servers and Pure FlashArray hardware. Pure Storage FlashArray is a high-performance platform that supports mixed workloads for shared storage efficiency. It is common to use FlashArray for hosting VMware vSphere workloads with both VMFS and vVols.

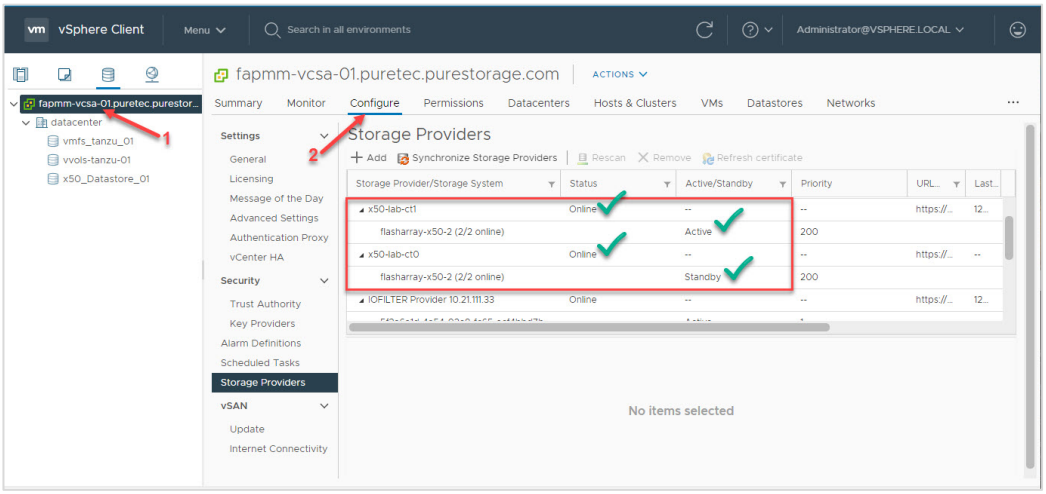
vSphere Client VASA FlashArray Storage Providers Registration

VASA registration can be accomplished in multiple ways, i.e., PowerShell, FlashArray UI, or vRealize Orchestrator. The example here uses vSphere Client with the FlashArray Plugin. Once the FlashArray providers are registered, creating and mounting a vVols datastore is a simple and seamless process.

1. With an existing and registered FlashArray via Plugin, navigate to the plugin.



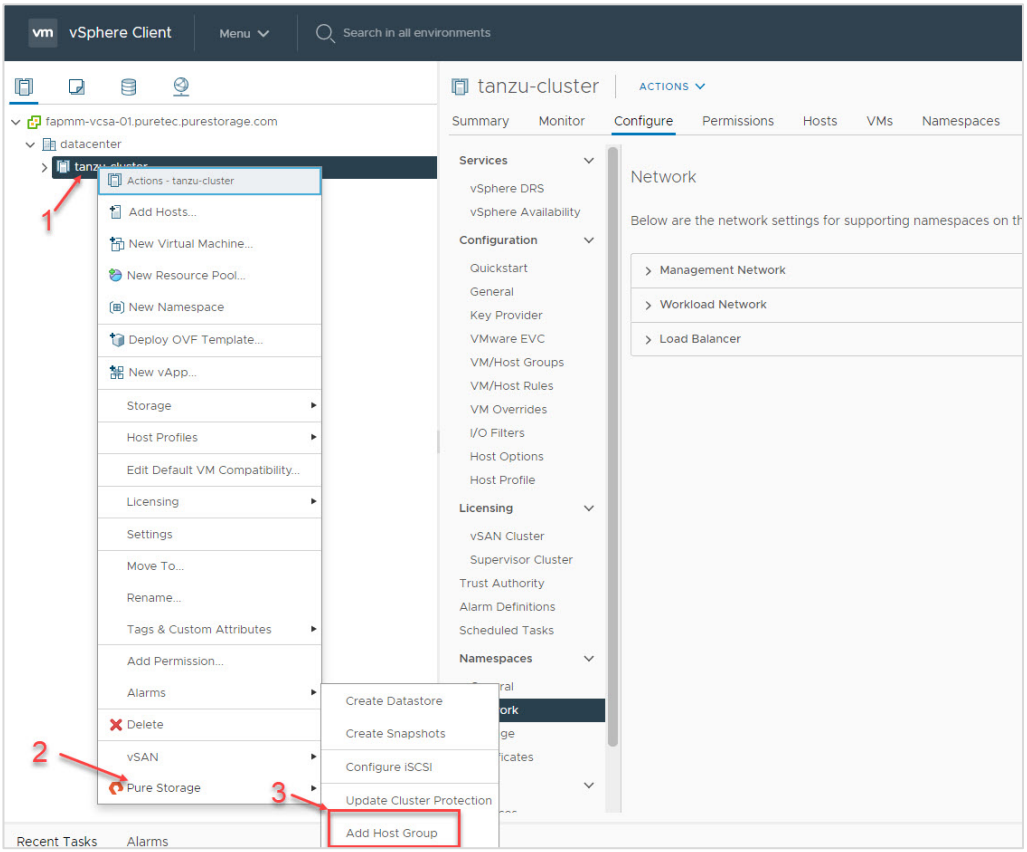
2. Once you have successfully registered the Storage Provider, navigate to the Storage Providers configuration page to confirm both providers are online and healthy.



Creating a FlashArray Host Group

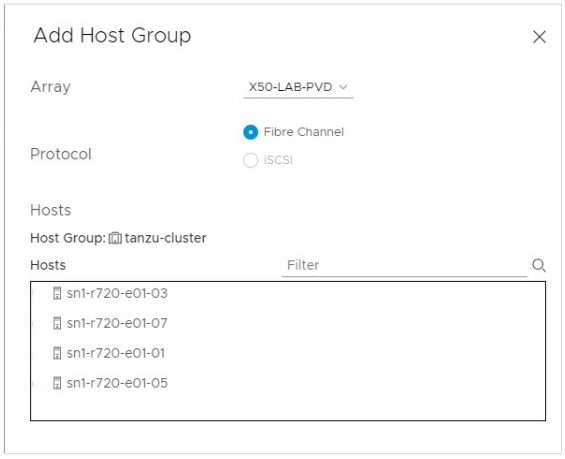
The vSphere Plugin for the vSphere Client will be used to streamline the creation of a host group to the cluster named *tanzu-cluster*. The best practice is to create and mount the vVol Datastore with the ESXi cluster mapped to FlashArray Host Groups.

1. Navigate to the tanzu-cluster and right-click to bring up the **Actions** menu, hover down to **Pure Storage** and click **Add Host Group** wizard.



2. Select the FlashArray and the appropriate protocol that you are using for the deployment. Review and confirm the available hosts and click **Create**.

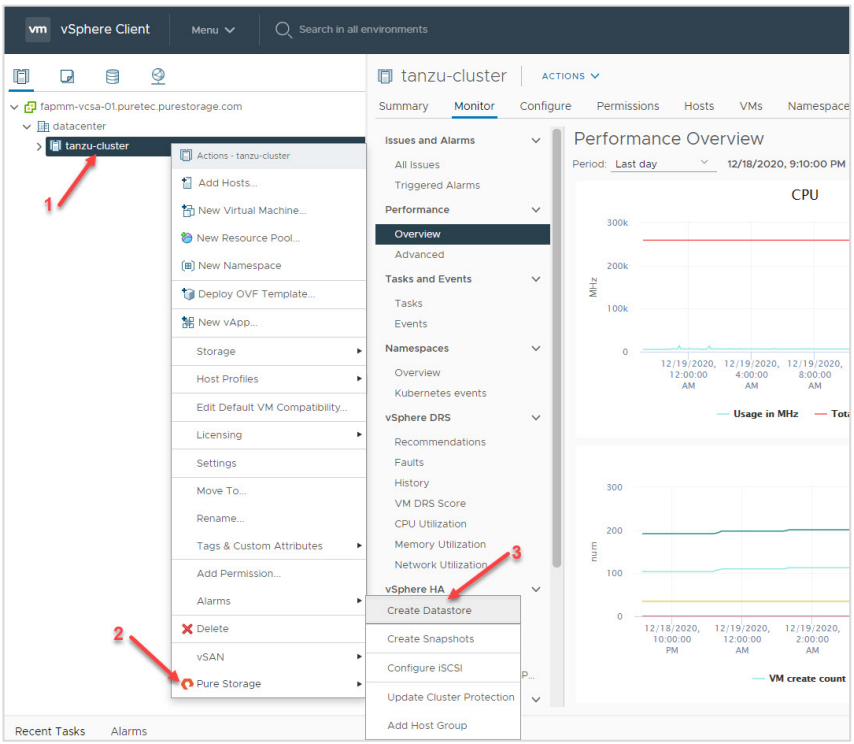
   NOTE: iSCSI deployments will default and Configure iSCSI Initiators on Hosts.



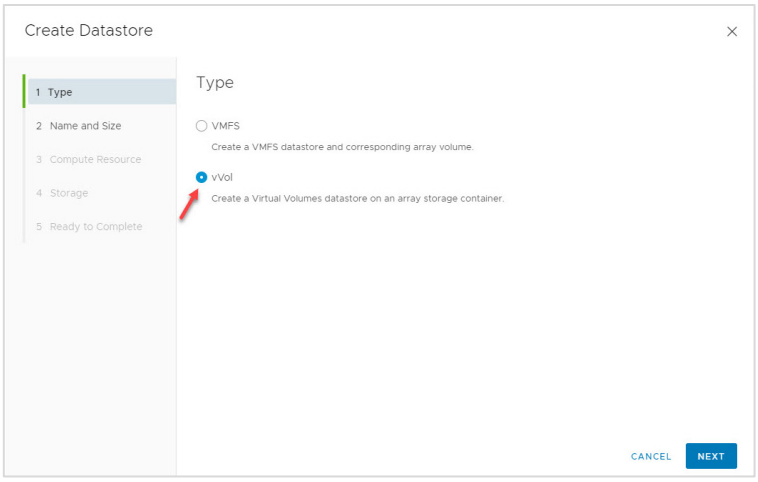
Creating and Mounting the vVols Datastore

Use the vSphere Plugin for the vSphere Client to streamline the provisioning process of the vVols datastore to the tanzu-cluster. The best practice is to create and mount the vVol datastore with the ESXi cluster mapped to a FlashArray host group.

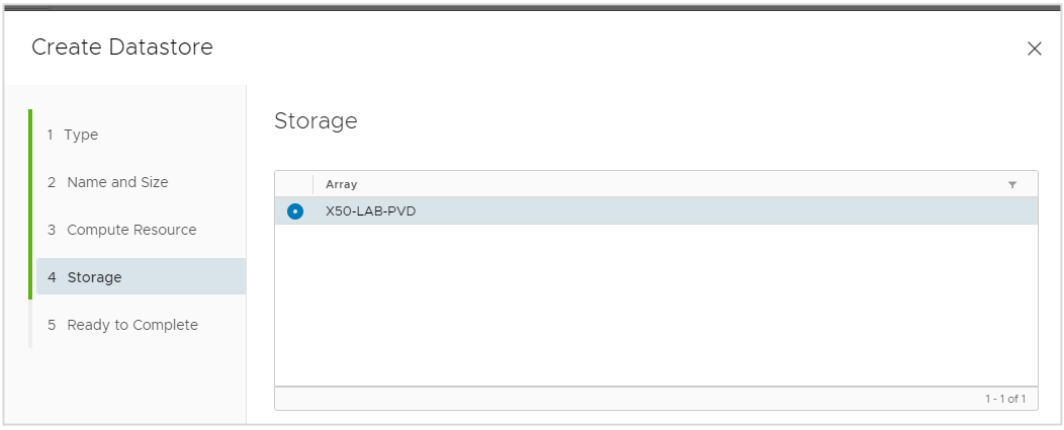
1. Navigate to the *tanzu-cluster* and right-click to bring up the **Actions** menu, hover down to **Pure Storage** and click **Create Datastore** wizard.



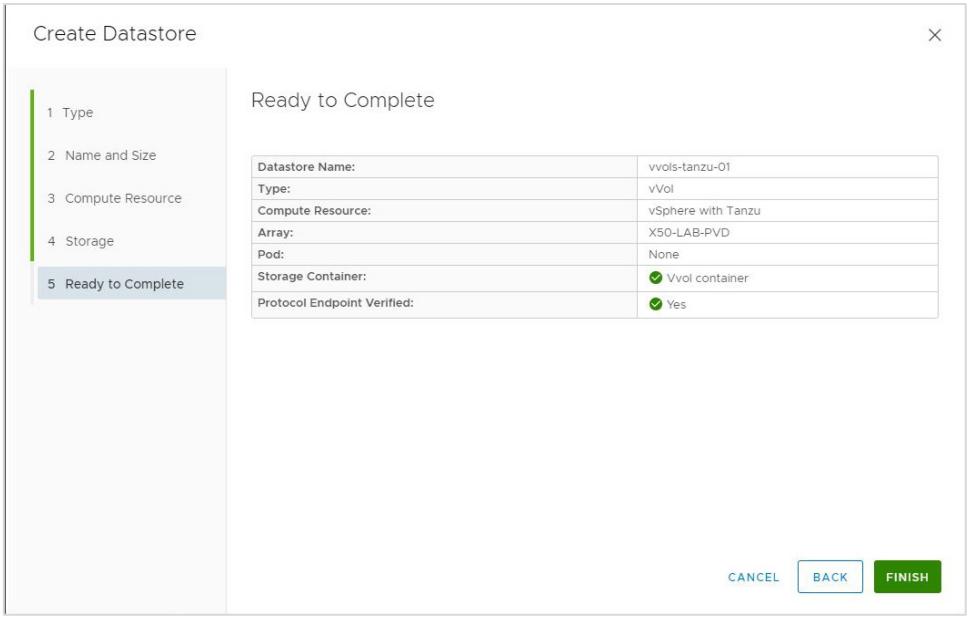
2. Select “vVol” as the type of datastore and proceed through the wizard. The following screens will ask for a datastore name. A vVol datastore is at a defaulted size of 8PB and can be modified by Pure Support as required. Select tanzu-cluster as the compute resource.



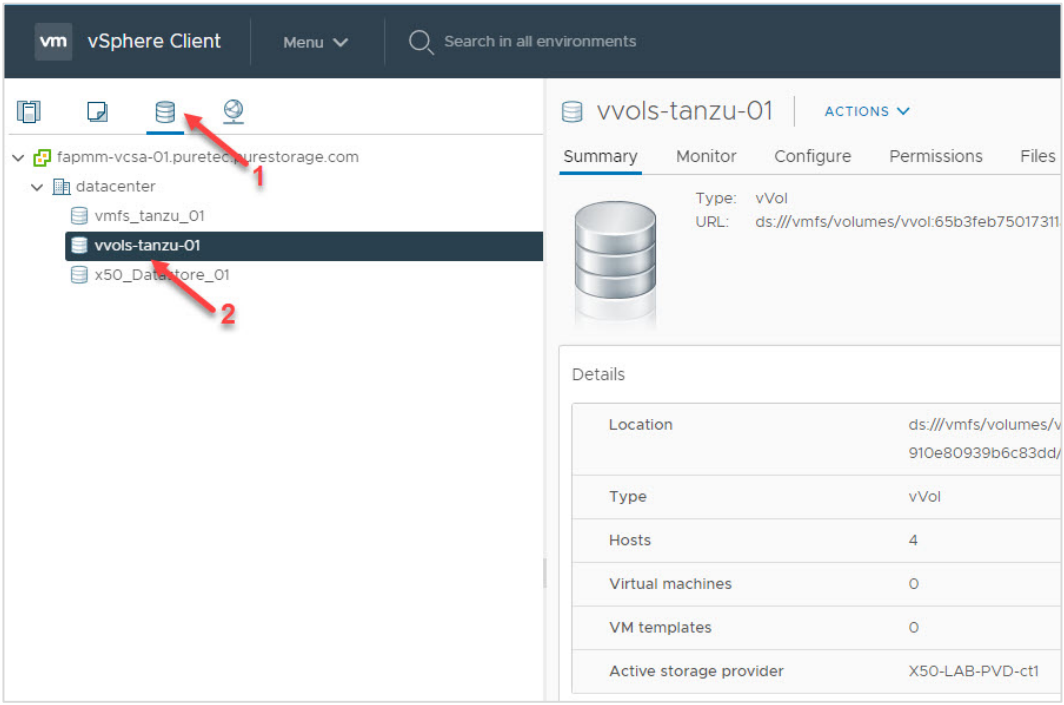
3. Select the FlashArray for the project.



4. Review your entries, click **Finish** to complete the task.



5. Navigate to the Datastore page and select the vVol Datastore for a summary.



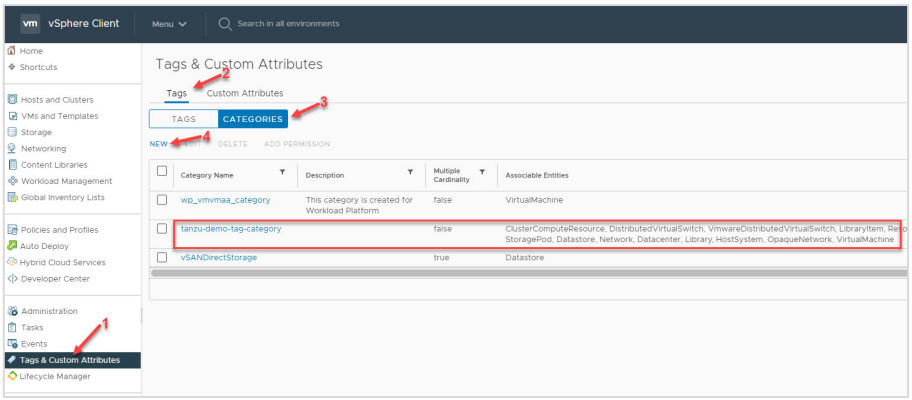
VMware vSphere Client Tasks

Creating Storage Policies

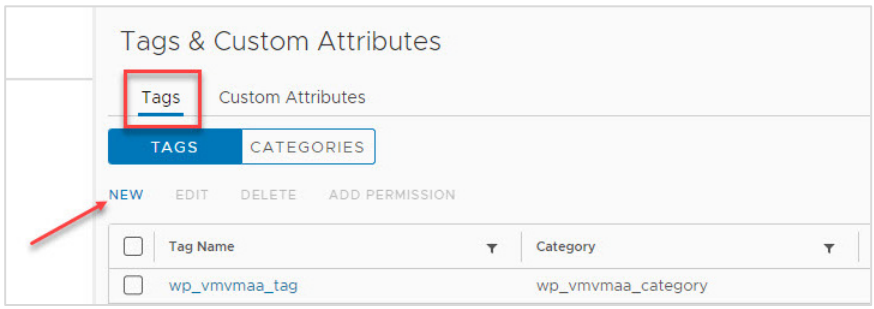
This deployment uses two storage policies. The first is a vVols policy (example name: vsphere-tanzu-gold-policy) for persistent storage and the second is for VMFS policy (example name: vsphere-tanzu-vmfs-policy) which will store VMs. These policies translate to storage classes in Kubernetes which will be inserted in the design files during the Tanzu Kubernetes cluster, application, and database deployment.

The FlashArray vVols datastore policy will use VMware's native Storage Policy-Based Management (SPBM) with a simple definition. A tag-based policy will be created for the FlashArray VMFS datastore.

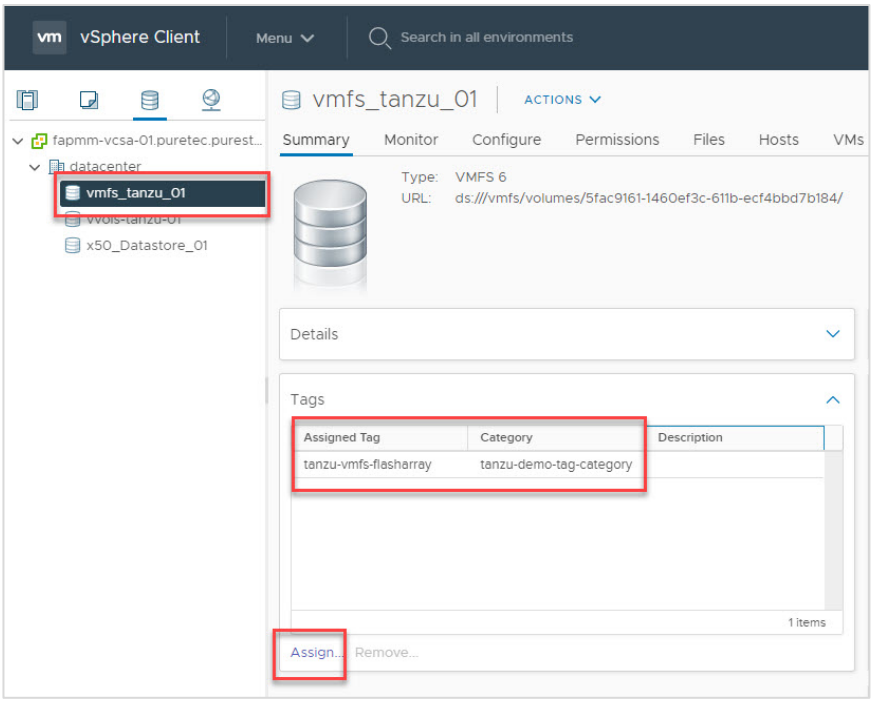
1. Create a new Category Name with default values. This deployment uses the name "tanzu-demo-tag-category" for ease of identification.



2. Additionally, create a new tag to be assigned to the FlashArray VMFS datastore.

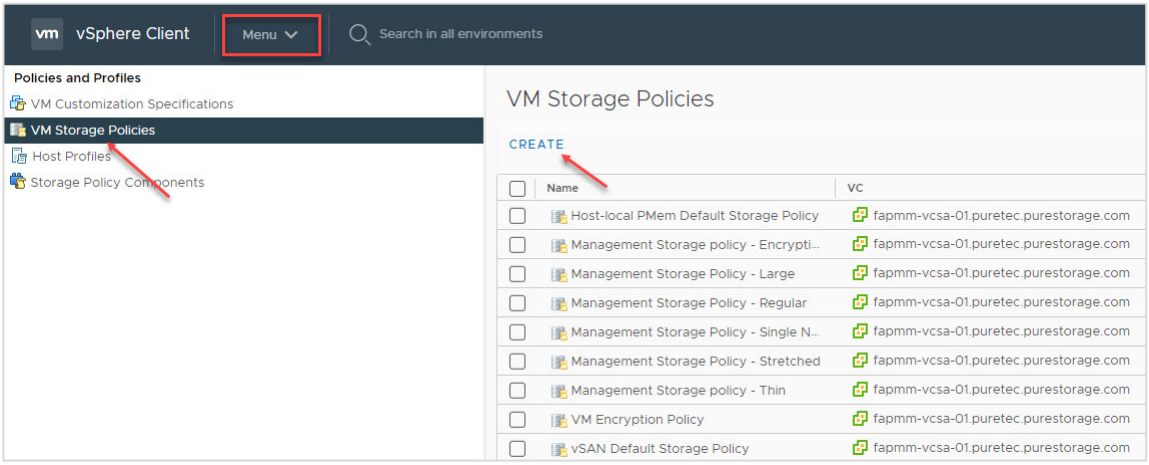


3. Navigate to the datastores and assign the new tag to the FlashArray VMFS Datastore.

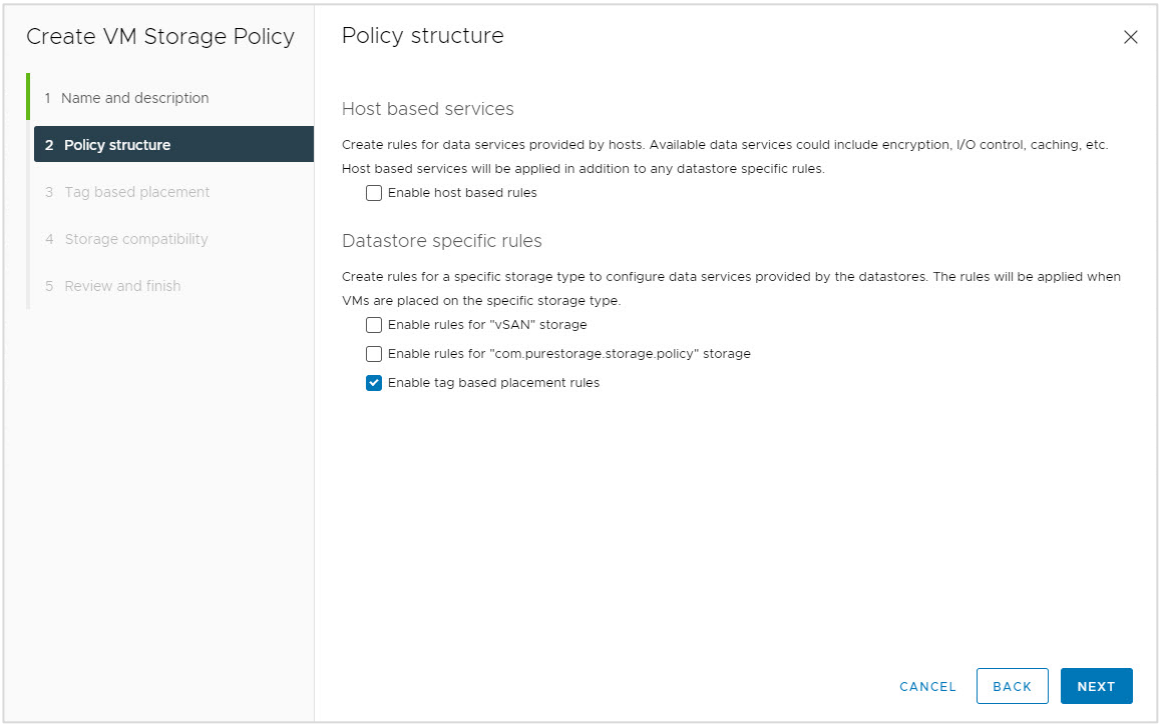


Create a Storage Policy for the FlashArray VMFS datastore.

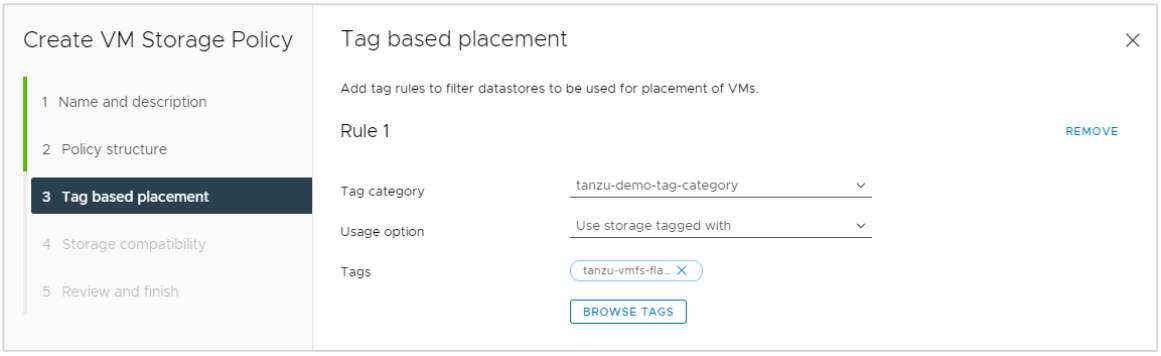
1. Navigate to the vSphere Client Menu, click **Policies and Profiles**, select **VM Storage Policies,** and click **Create**.



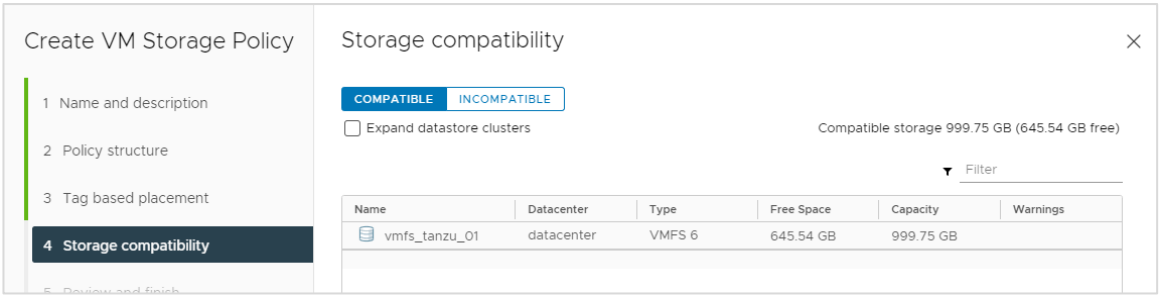
2. Add a description if desired and click **Next**. The policy structure will be based on datastore-specific rules, so check the box for **Enable tag-based placement rules** and click **Next**.



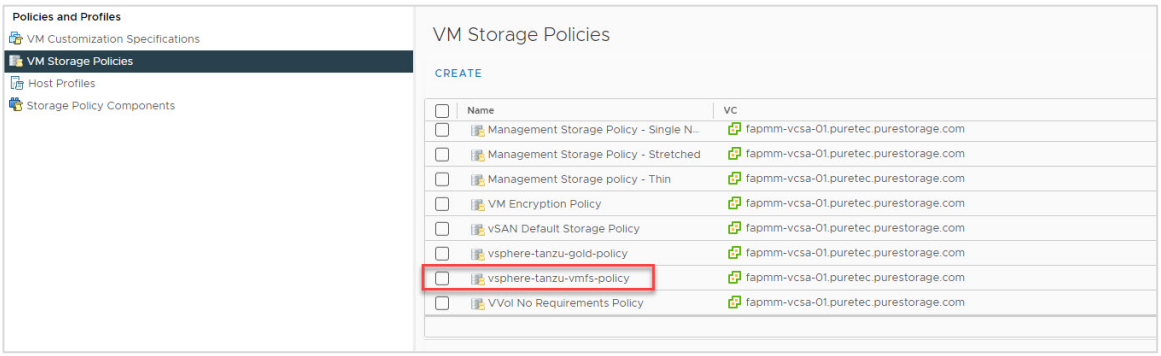
3. Select the tag category for the rule and add the appropriate tag with **Browse tags**. Click **Next**.



4. Review and select the compatible Datastore type and click next to review and finish.

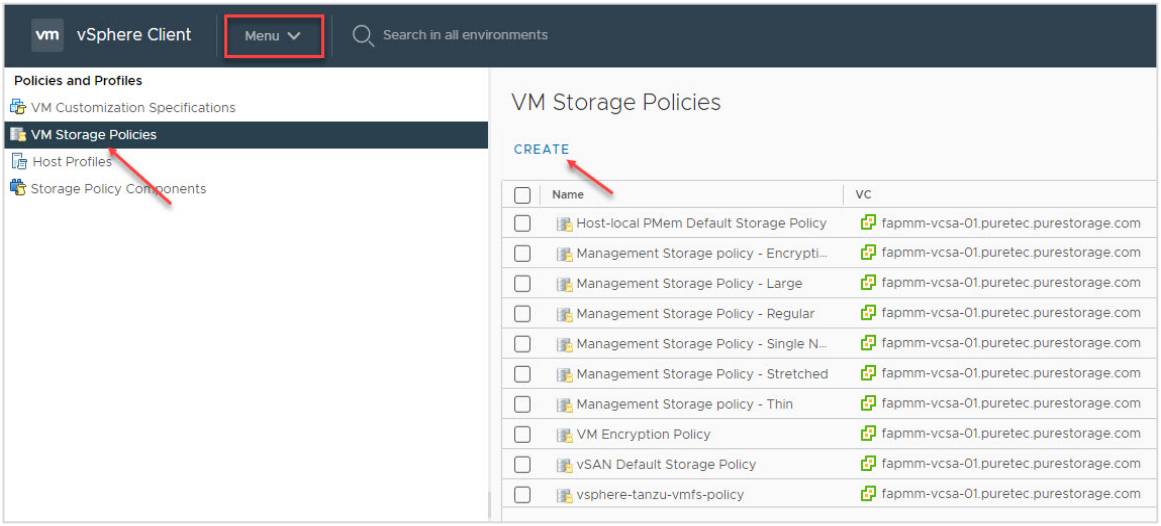


5. Review the list for the new VMFS policy.

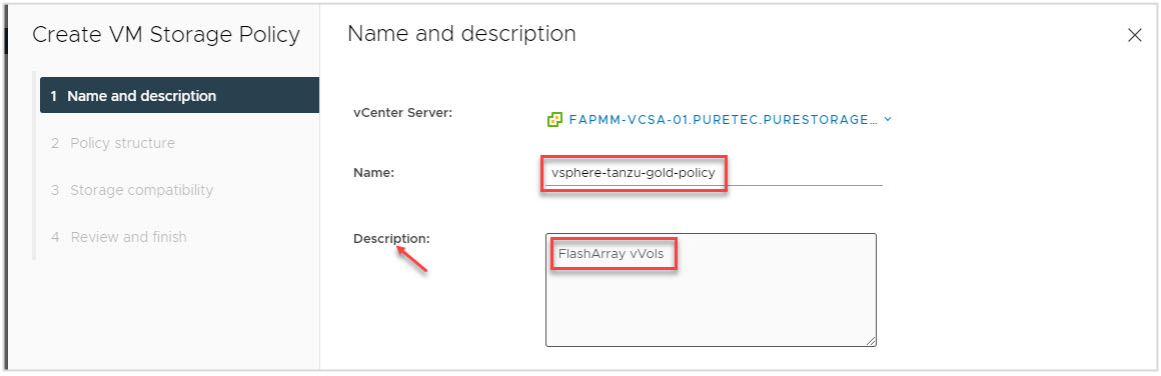


The next step is to create a storage policy for the FlashArray vVols Datastore, using Storage Policy-Based Management (SPBM) with a simple rule.

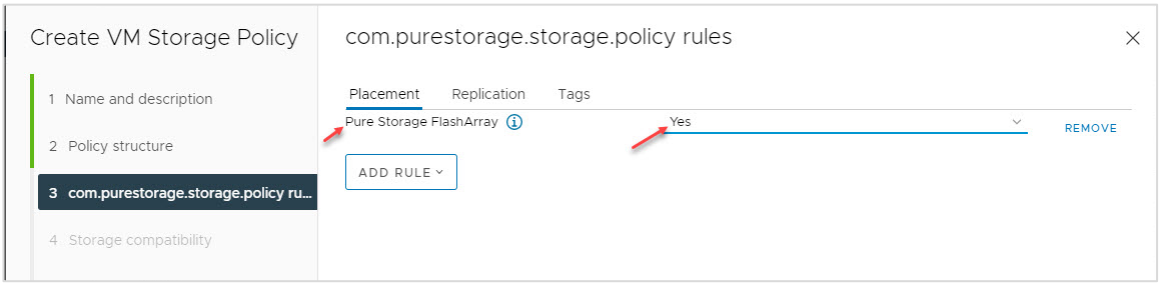
1. Navigate to the vSphere Client Menu, click **Policies and Profiles**, select **VM Storage Policies,** and click **Create**.



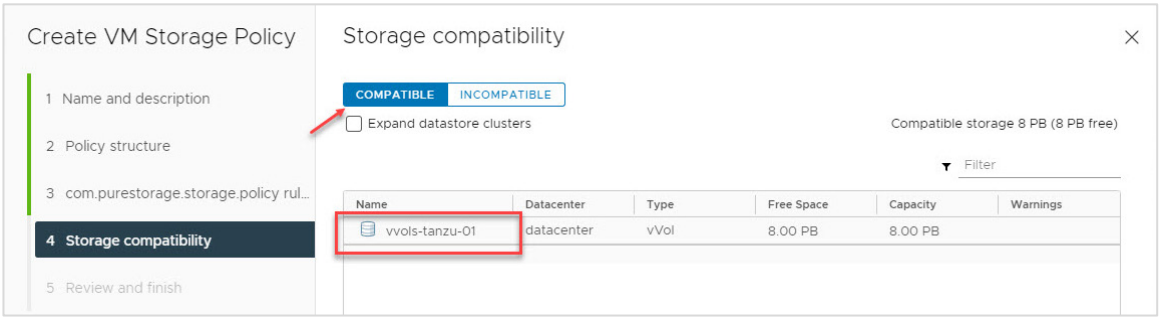
2. Use a descriptive Name with a Description (optionally) and click **Next**. The policy structure will be based on datastore-specific rules, check the box for Enable rules for "com.purestorage.storage.policy" storage and click **Next**.



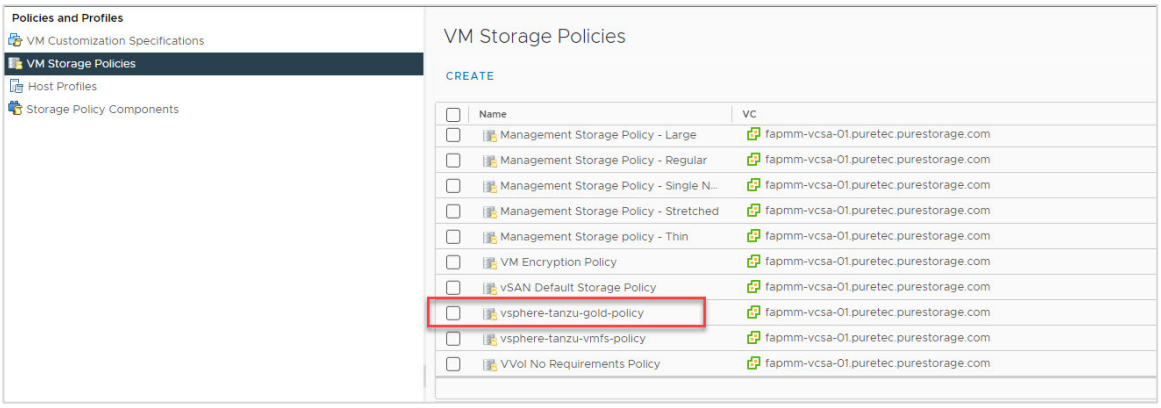
3. Using the **Add Rule** drop-down menu, select **Pure Storage FlashArray** rule with **Yes** value and click **Next**.



4. Select the FlashArray vVols datastore and click **Next** onto the Review screen and **Finish**.



5. Review the list for the new vVols policy.

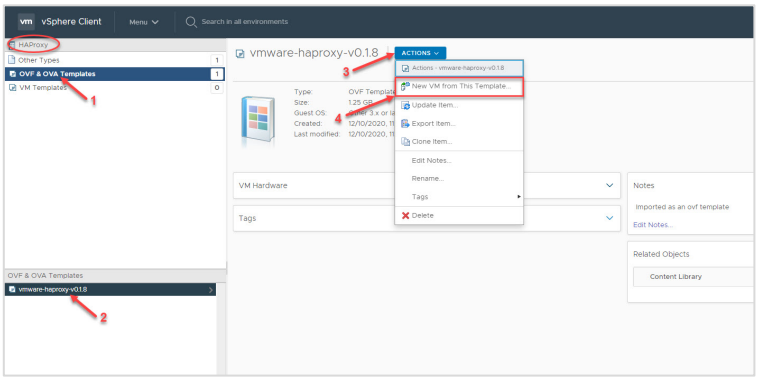


HAProxy

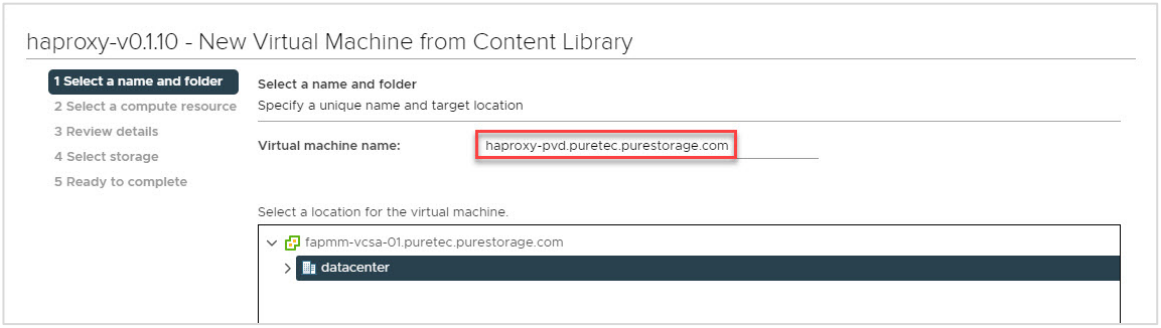
The deployment of the HAProxy Load Balancer is wizard-driven and requires attention with IP Address entries. While it is possible to modify the HAProxy post-deployment, we recommend deleting the HAProxy VM and re-deploying it to eliminate complexities.

Navigate to the Content Libraries through the **vCenter Server Menu** drop down and expand the HAProxy Library.

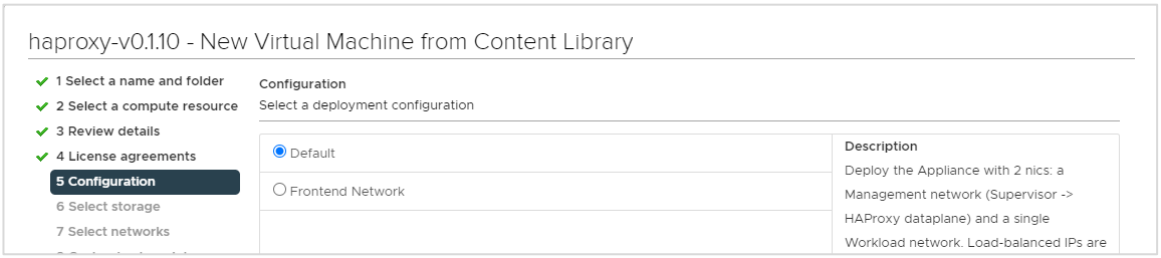
1. Select OVF & OVA Templates, select the haproxy-v0.2.0 OVA and select New VM from This Template in the **Actions** drop-down.



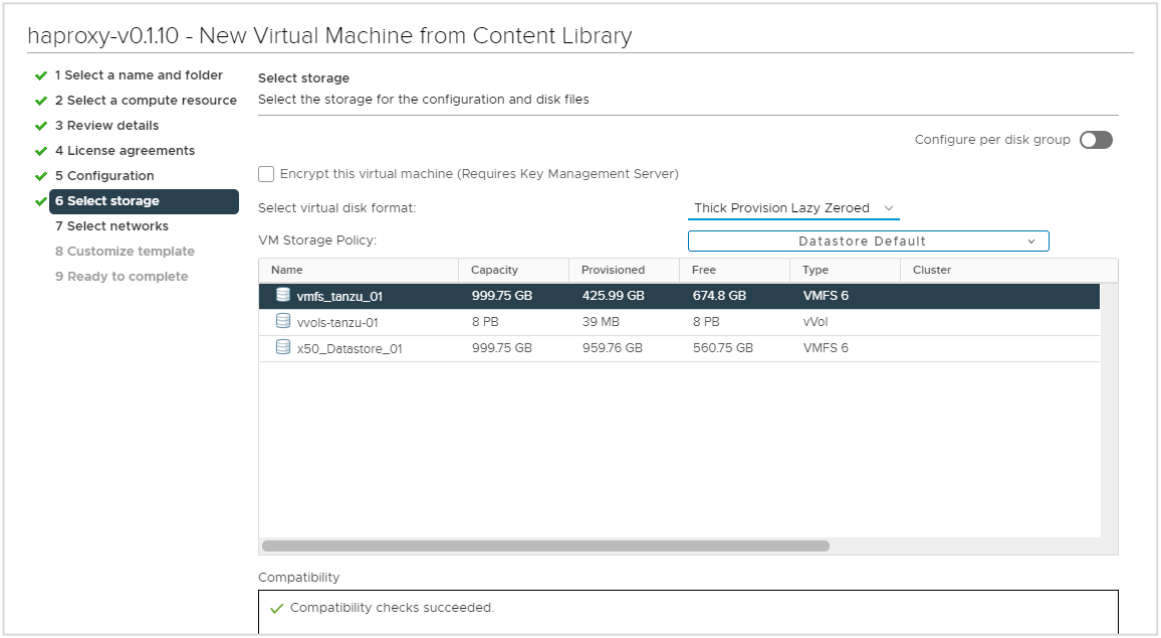
2. Enter a virtual machine name for your haproxy and select the location.
3. Select destination cluster for the project tanzu-cluster.
4. Review details and accept the license agreements.



5. Select the **Default** option to deploy the HAProxy with two network interface controllers.

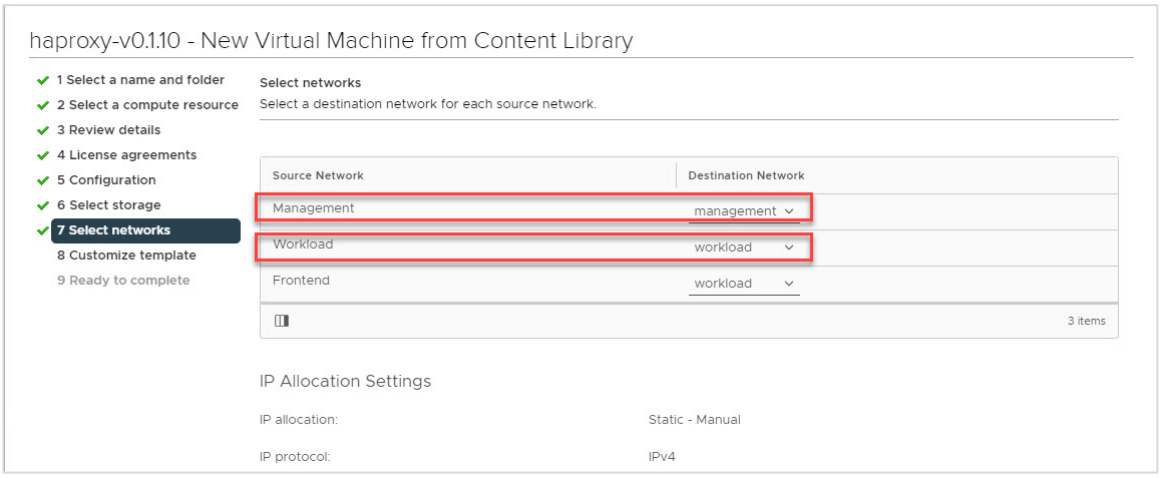


6. Select the vmfs_tanzu_01 Datastore storage option for the HAProxy. We strongly recommended keeping the virtual disk format at the default setting of **Thick Provisioned Lazy Zeroed**.



7. Select each destination network to match the source network. The HAProxy's management Interface and IP Address on the management source network **must** be connected to the management network on the destination network. The workload must be connected to the destination network.

   NOTE: The frontend source network does not require a change in Destination Network selection.

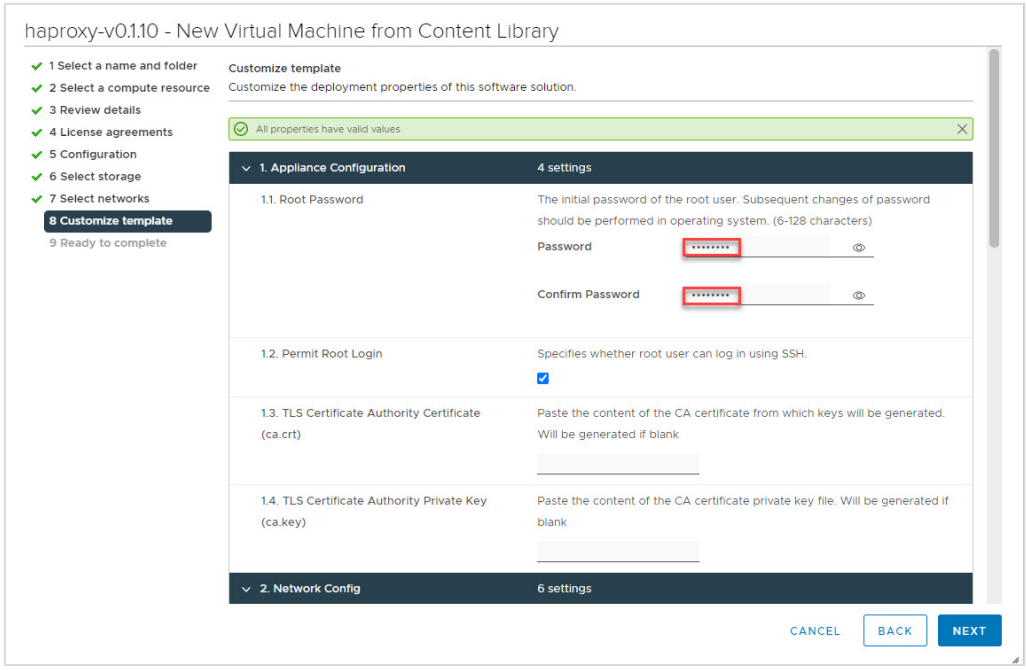


8. Configure the appliance as follows:

   a. 1.1: Enter a password for the root user to manage the HAProxy via the Management network. Take note of this password, as it will be used with the *root* user once the VM is powered on for additional tasks.

   b. 1.2: Permit Root Login: Default as enabled.

   c. 1.3-1.4: Do not require an entry.

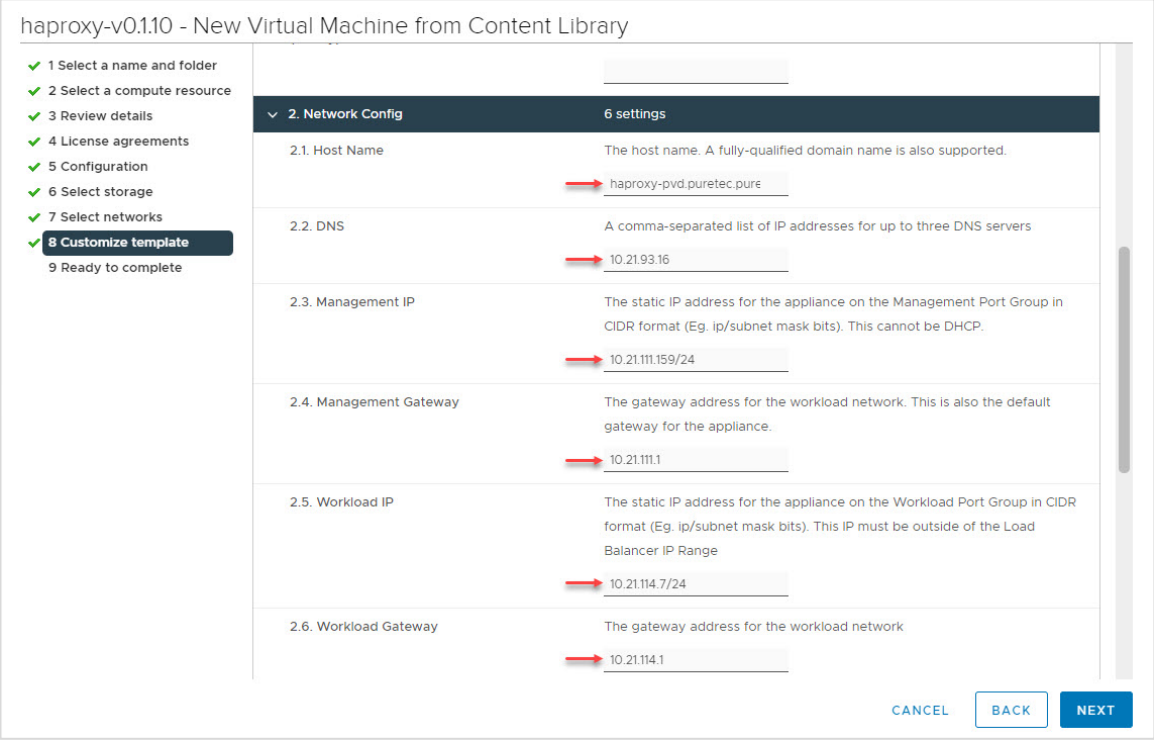


9. Network configuration entries here must be entered and reviewed thoroughly to ensure accuracy based on the worksheet. It is important to note that entries for sections 2.3 and 2.5 require the proper CIDR format based on the IP Address, for example: 10.21.111.159/24 (e.g., IP/subnet mask bits), /24 is a subnet mask of 255.255.255.0

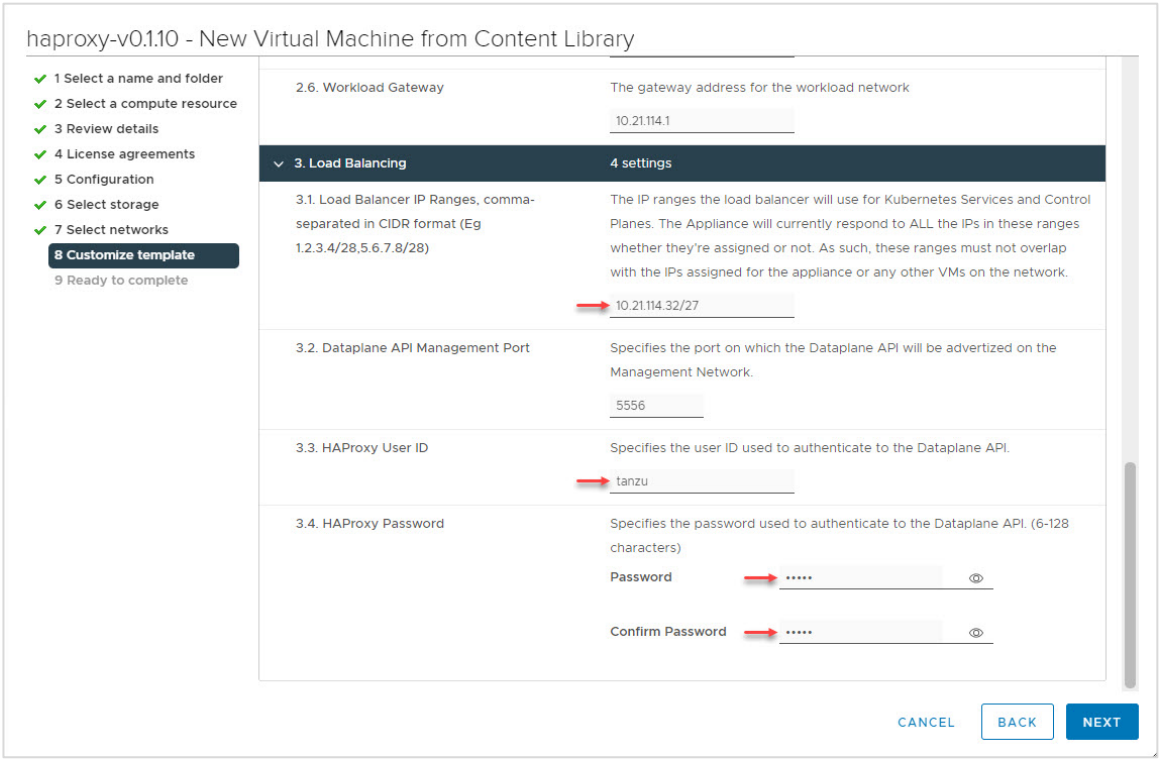
   a. 2.1: It is best practice to use an FQDN.

   b. 2.3-2.5: The management and workload IP address entry must include the proper CIDR format.

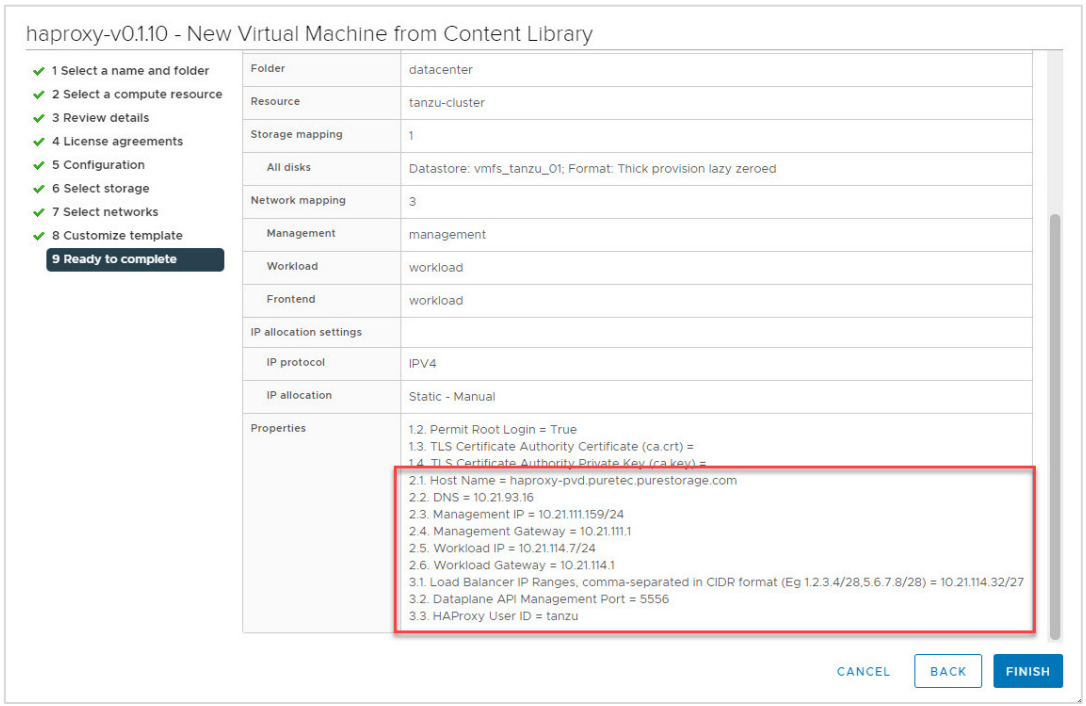
   c. 2.4-2.6: The gateway IP address entry does not require a CIDR format.



10. Load balancing entries must be entered and reviewed thoroughly to ensure accuracy based on the worksheet. Entry for section 3.1 requires the proper CIDR format based on the first IP address of the load balancer IP range from the worksheet in CIDR format. Example: 192.168.114.32/27 with /27 CIDR range of 192.168.114.32-195.168.114.63



11. Review your entries and take note of the IP Address on Section 2.3, Management IP. It will be used to log into the HAProxy to confirm the installation and pull the certificate authority to enable Workload Management. Review thoroughly and click **Finish** to start the HAProxy build.



12. Check the status of the OVA build in vCenter, Power-On HAProxy once the build is complete.
13. You must be able to ping the management IP, workload IP, and the load balancer IP range to enable workload management.
    a. If you are able to ping all the IP addresses, you have successfully installed the HAProxy and will be able to enable Workload Management.
        i. Retrieve the Server Certificate Authority from the HAProxy.
            1. SSH to the HAProxy management IP address using root user and the password from step 1.1
        ii. Once logged in as root run `cat /etc/haproxy/ca.crt`, copy the entire contents and save for the workload management installation.
    b. If you are unable to ping all IP addresses, give the HAProxy a little more time to start all of its services while rechecking your entries for accuracy with CIDR entry requirements.
        iii. Validate physical core networking for any restrictions in the network.
        iv. Check vDS DSwitch port group settings for appropriate VLAN ID.

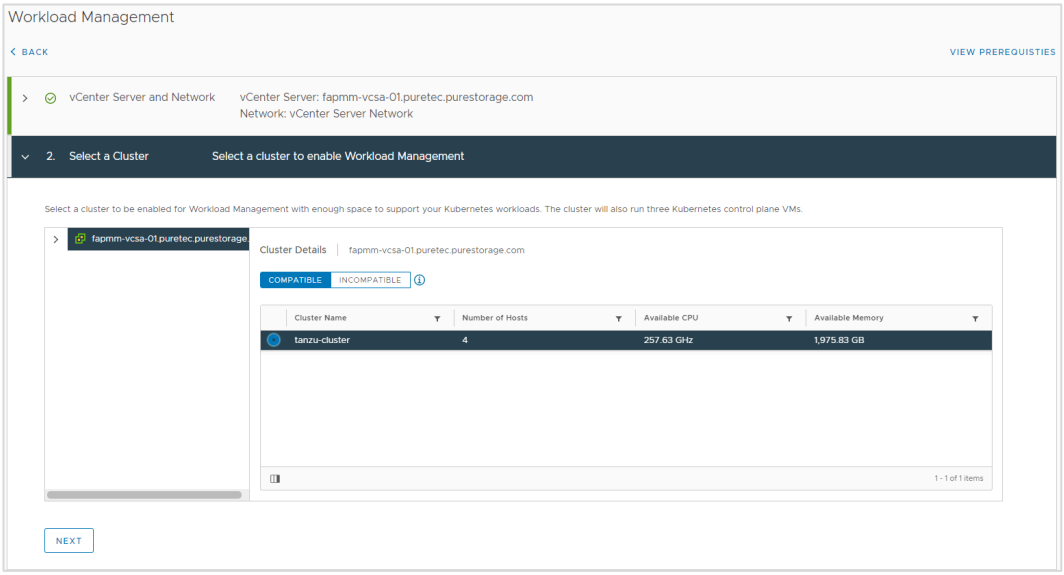
Enable Workload Management and Create a Namespace

Enabling the vSphere Workload Management requires a fully functional HAProxy, which is the ability to ping the management IP, workload IP, and the entire load balancer IP range(s). Create the Namespace once Workload Management is enabled.

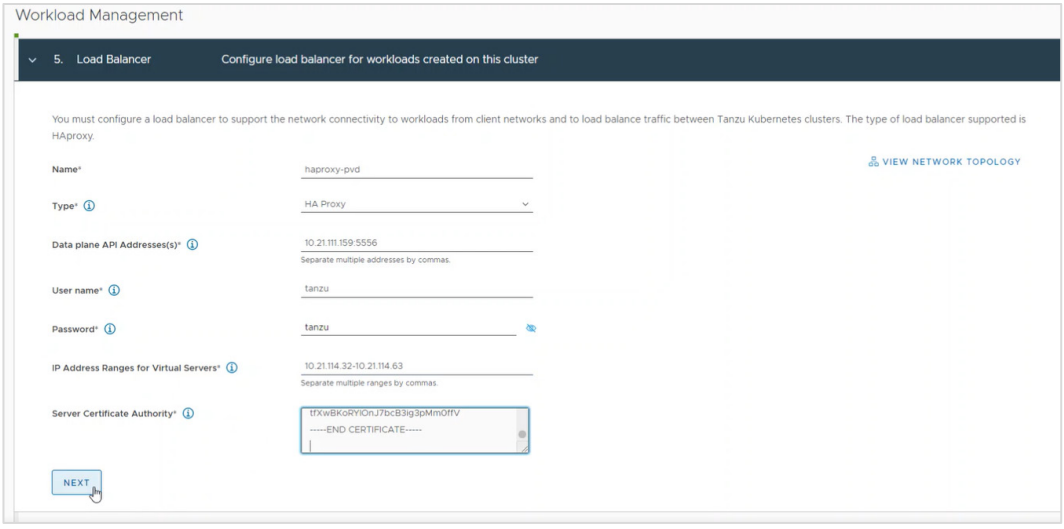
Enable Workload Management

Navigate to vSphere Client Menu and select **Workload Management**. Click **Get Started** to build the Cluster of Supervisors.

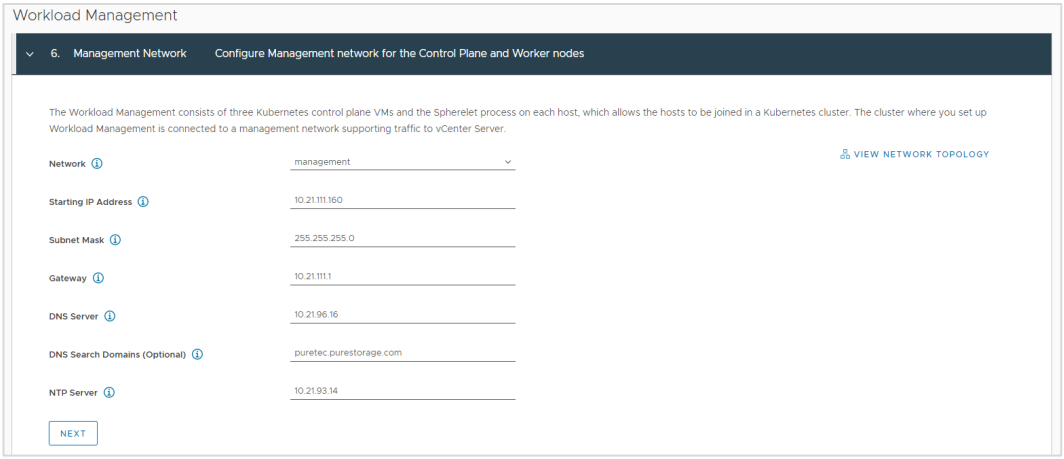
1. Select vCenter Server Network and click **Next**.
2. Select a Compatible Cluster and click **Next**.



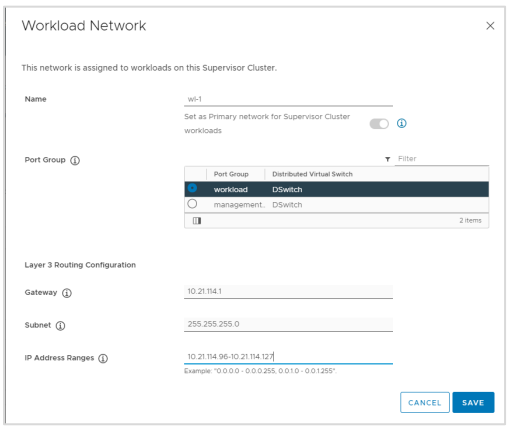
3. Select a Deployment Size and click **Next**.
4. Using the drop-down menu, select vsphere-tanzu-vmfs-policy for the Control Plane VMs and click **Next**.
   a. Configure the Load Balancer.
   b. IP Address Range is based on your CIDR LB Range Example: 10.21.114.32/27 would be 10.21.114.32-10.21.114.63.
   c. Server Certificate Authority: You must retrieve from HAProxy via HAProxy Management IP Address as root. Run `cat /etc/haproxy/ca.crt`.



5. Enter management network data of Supervisor Control Plane virtual machines from the worksheet. The **Starting IP Address** entry is the first of the five supervisor control plane IP addresses.



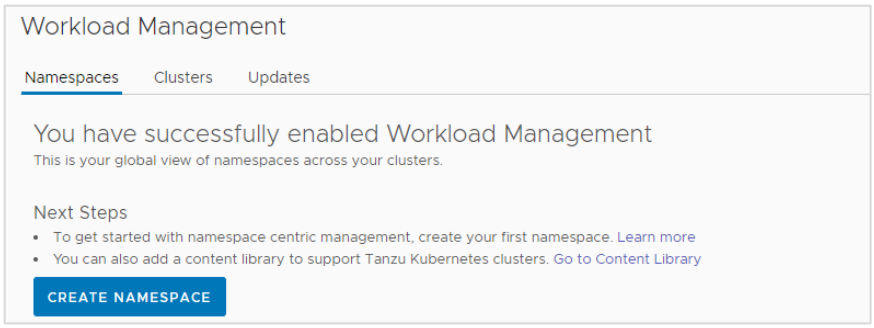
6. For the Workload network, use default IP Address for services, enter the DNS server IP and add the workload network.
   a. Workload network: Create a name, select the workload port group, and provide the gateway and subnet of the workload network. The "IP Address Ranges" is a range for virtual machines and must be outside of the load balancer range. The total number of IP addresses for this "IP Address Ranges" is 30. Click **Save** and continue.



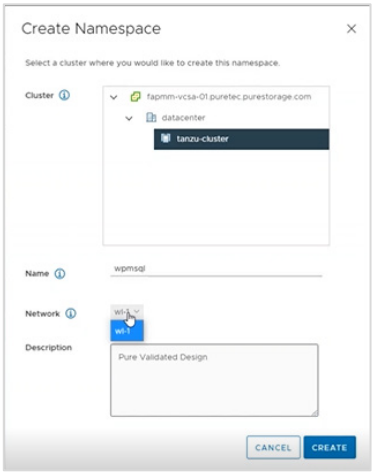
7. Select the TKG Library. Click **OK** to continue.
8. Review and confirm, then select **Finish** to enable workload management.

Create and Configure the vSphere Namespace

Navigate to the vCenter Server menu and select **Workload Management**. Select **Namespaces** to create a namespace.

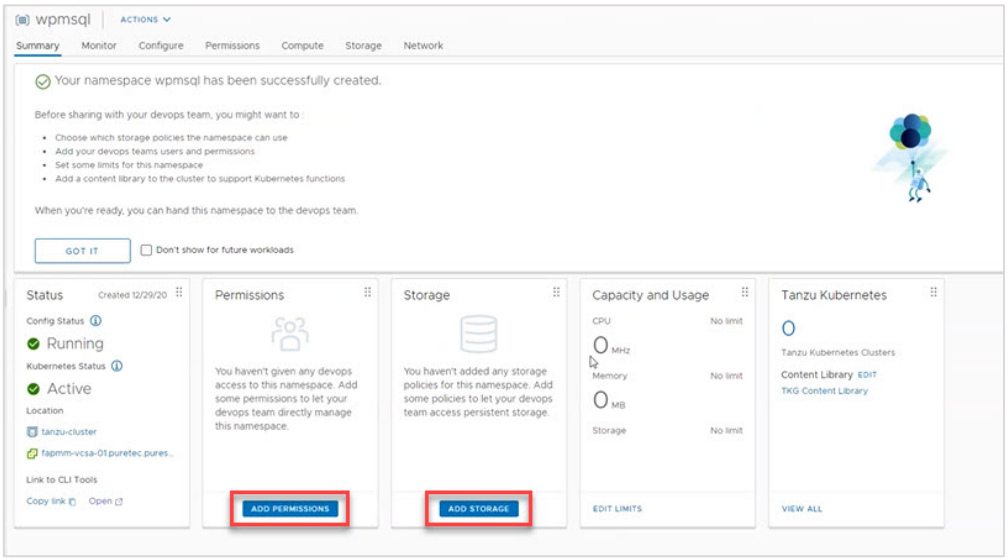


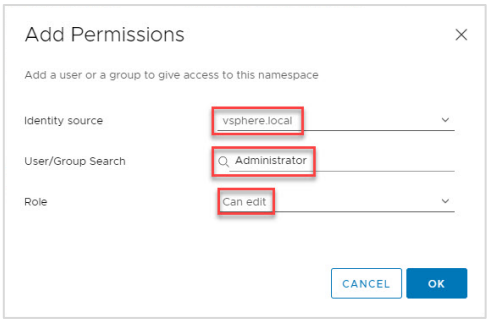
1. Select the designated vSphere Cluster, assign a name to the new namespace and choose the designated network to support Tanzu workloads. Click **Create** and the new namespace will be created.

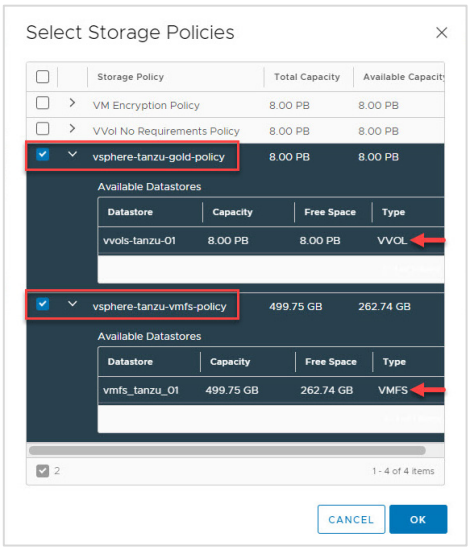


2. Add Administrator@vsphere.local Permissions and assign storage policies.

   a. Add appropriate user as authorized and applicable for the project, and authentication to the Namespace is via vSphere Plugin for the kubectl CLI Tool, and vCenter **Single Sign-On Credentials**.

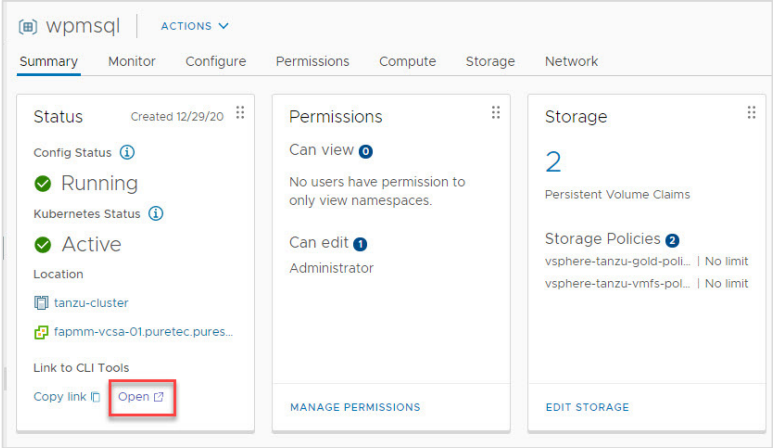






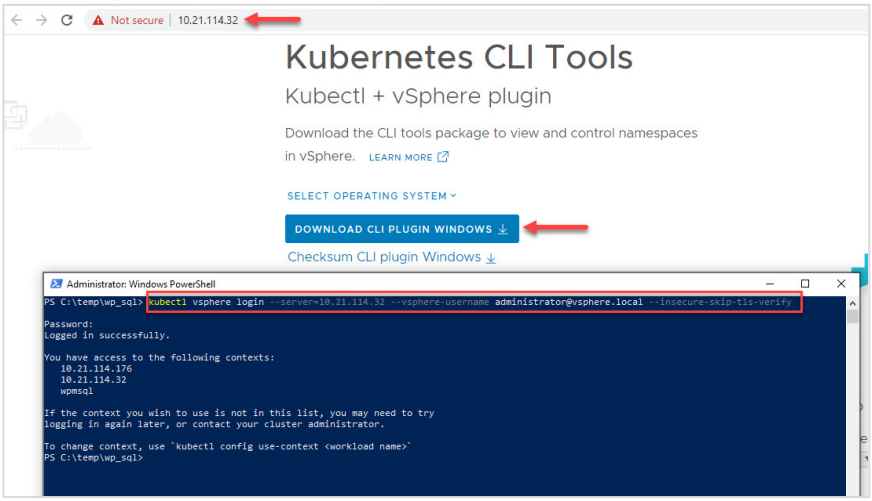
Tanzu Kubernetes Cluster Deployment

An instance of Tanzu Kubernetes Cluster will be deployed to the *wpmsql* workspace using the kubectl CLI tool.

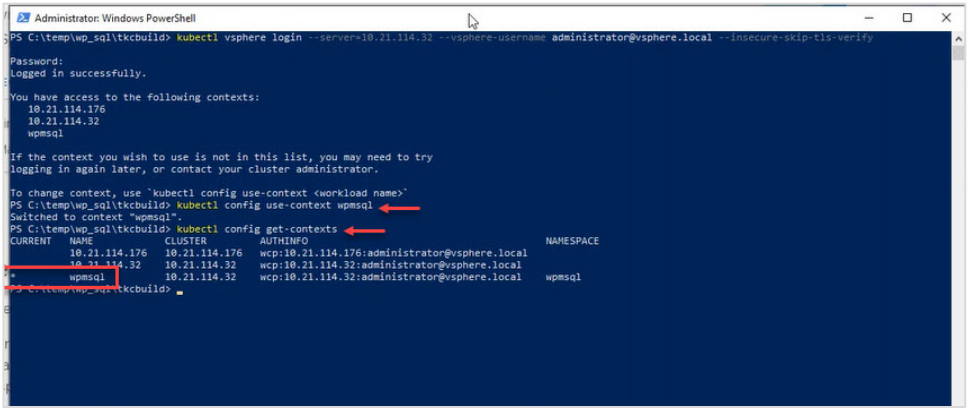


1. Once Kubernetes CLI Tools is installed, log into the Namespace. Example:

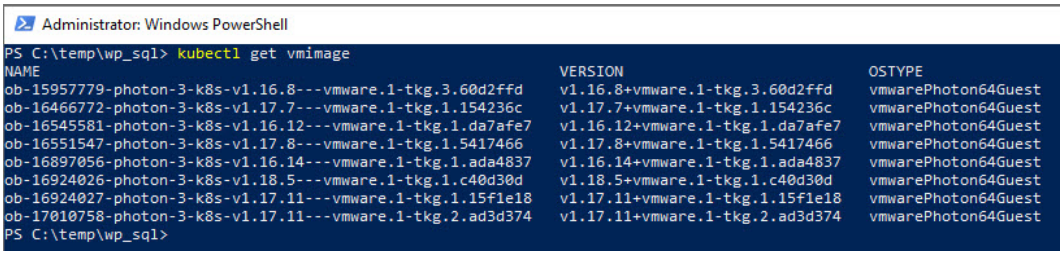
   ```
   kubectl vsphere login --server=10.21.114.32 --vsphere-username administrator@vsphere.local --insecure-skip-tls-verify
   ```



2. Use *wpmsql* Namespace. Example: `kubectl config use-context wpmsql`. Run, for example: `kubectl config get-contexts`. The asterisk indicates the current context in which you are logged into.



3. Use `kubectl get vmimage` to see the list of available Tanzu Kubernetes Cluster versions.



4. Using a source-code editor such as Visual Studio Code, build a design yaml file to deploy the Tanzu Kubernetes Cluster with `storageClass: vsphere-vmfs-policy`.

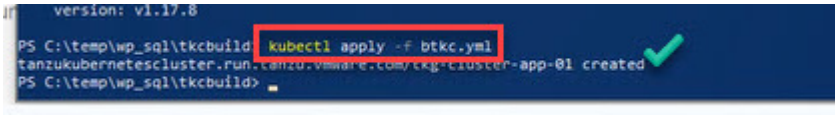
   NOTE: Tanzu Kubernetes Cluster deployments are currently only supported on FlashArray VMFS Datastores.

Tanzu Kubernetes Cluster design yaml file example:

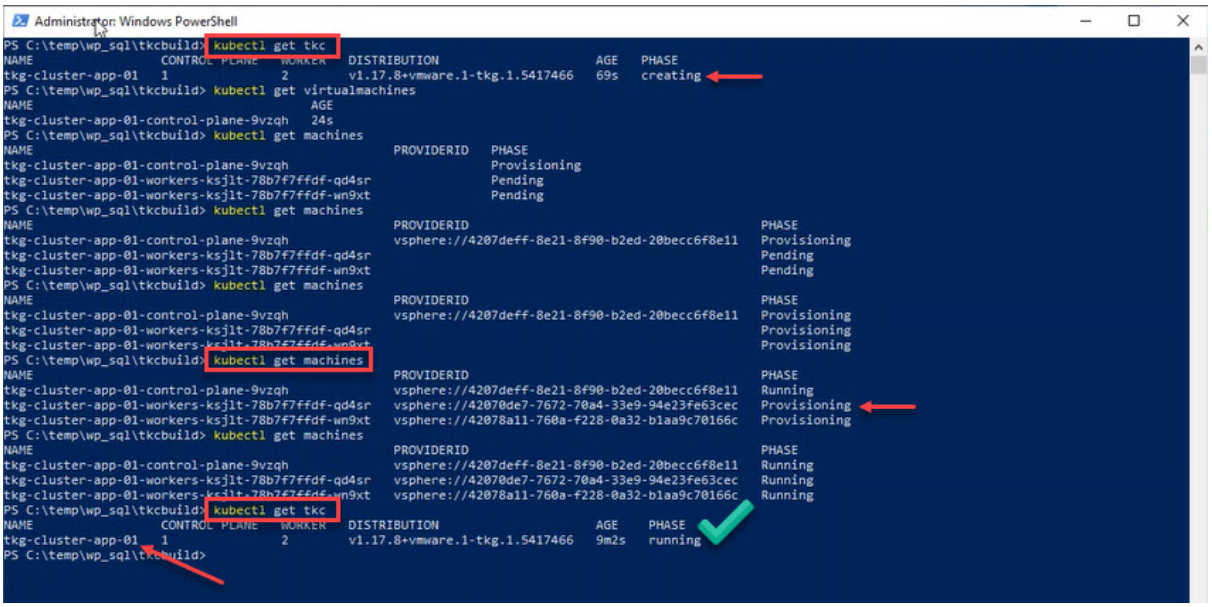
```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkg-cluster-app-01
spec:
  topology:
    controlPlane:
      count: 3
      class: best-effort-medium
      storageClass: vsphere-tanzu-vmfs-policy
    workers:
      count: 3
      class: best-effort-large
      storageClass: vsphere-tanzu-vmfs-policy
  distribution:
```

```
      version: v1.20
```

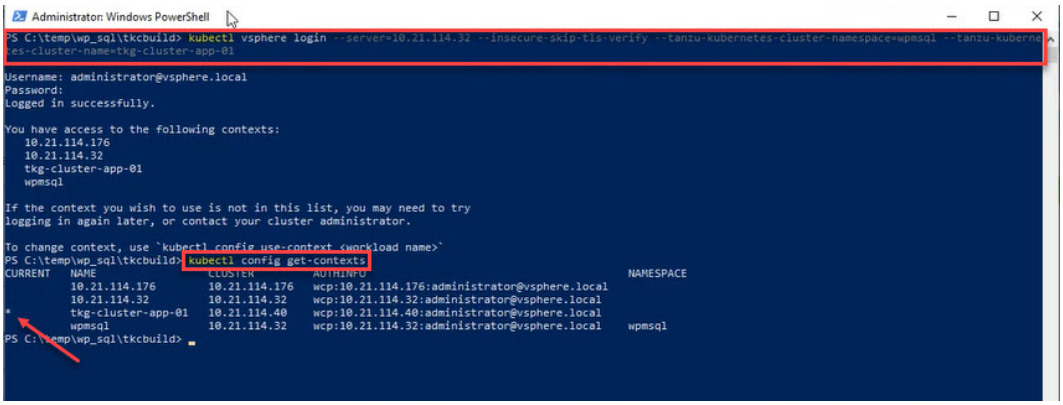
5. From the directory of the TKC, design the yaml file and manifest the deployment, Example: `kubectl apply -f btkc.yml`



6. Check the status of the Tanzu Kubernetes Cluster deployment `kubectl get tkc`. Once the TKC is running, proceed to the next step.



7. Log into tkg-cluster-app-01. Example: `kubectl vsphere login --server=10.21.114.32 --insecure-skip-tls-verify --tanzu-kubernetes-cluster-namespace=wpmsql --tanzu-kubernetes-cluster-name=tkg-cluster-app-01`
8. Use `kubectl config get-contexts` to validate with the asterisk that you are logged into the Tanzu Kubernetes Cluster.

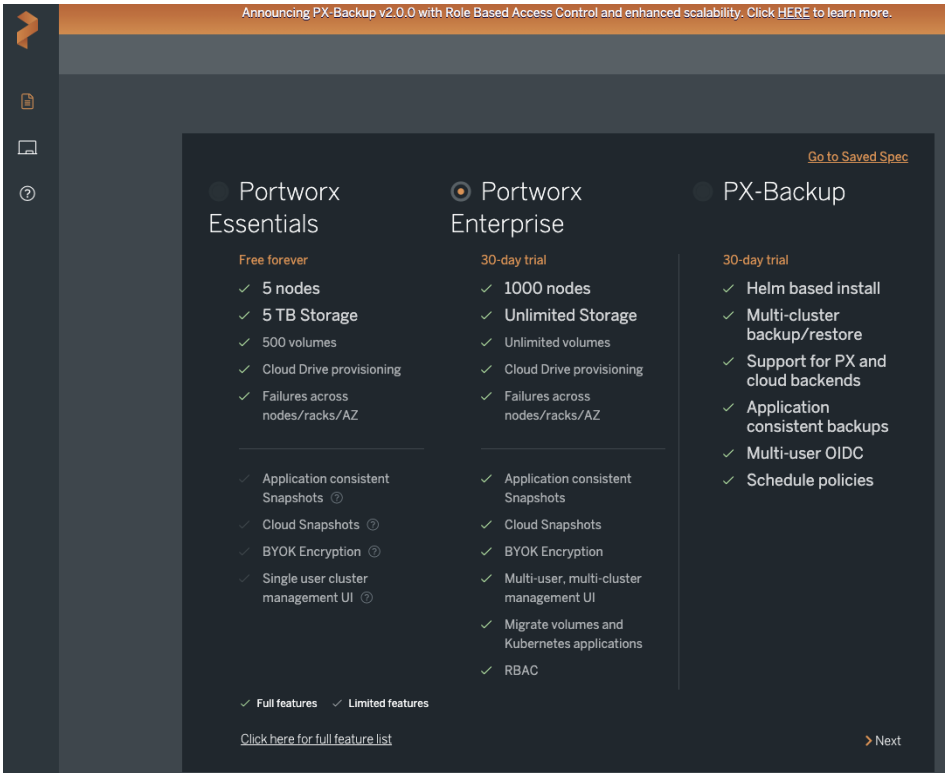


Deploy Portworx Enterprise to the Tanzu Cluster

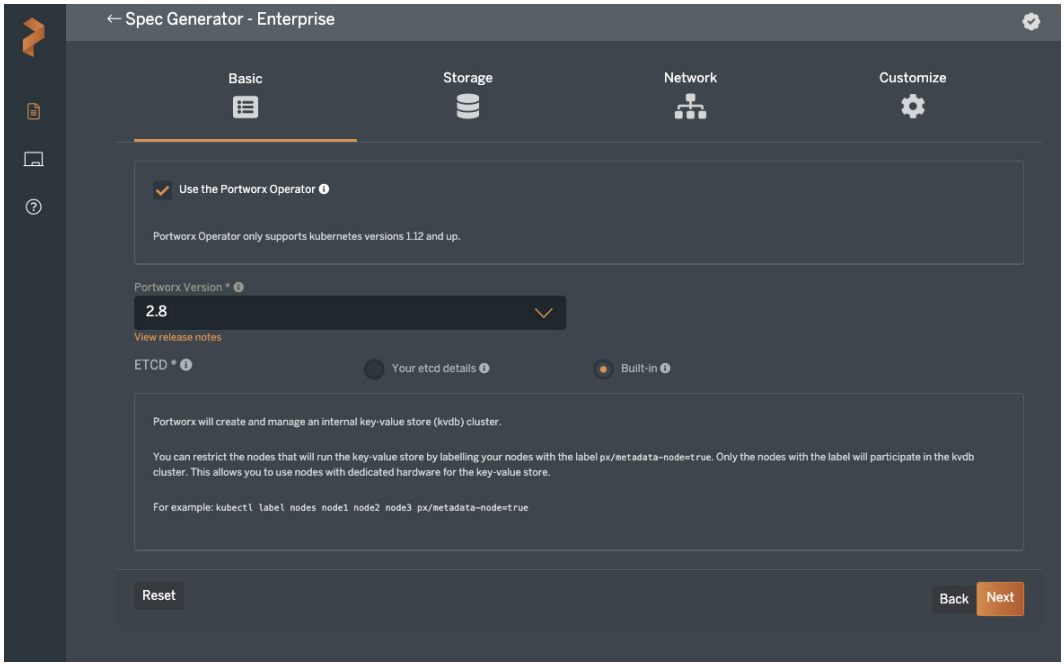
Preparing the Portworx Storage Cluster Manifest

Once you have completed deploying your Tanzu cluster, you are ready to move on to deploying Portworx. Portworx by Pure Storage provides a wizard driven website to generate the appropriate deployment manifests. To prepare the manifests follow the screen shots below, some selections used should be changed to align with your environment.

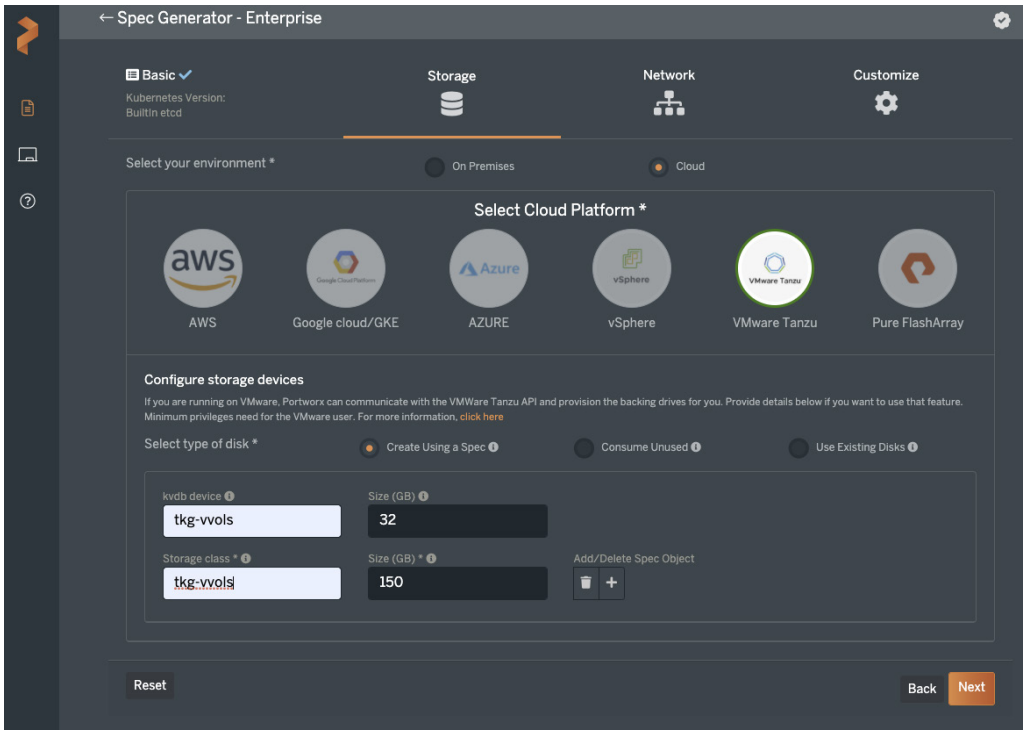
1. Navigate to https://central.portworx.com and either create a new account or use one of the supported Identity providers.
2. Once logged into the page, you will be presented with the "Install and Run" page. We will start the process here.
3. On the install and run page, select Enterprise Edition then click "Next."



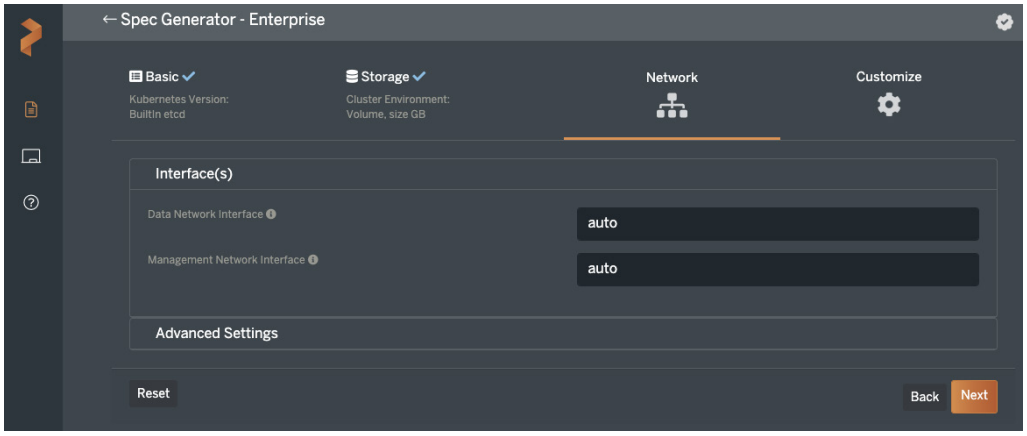
4. On the next page we will begin configuring Portworx. To follow this guide, it is advised to make the same selections.



5. The next screen provides the options to configure a "On-Premises" or "Cloud" environment. Portworx supports VMware vSphere as well as Tanzu Kubernetes as a Cloud provider, so we will select "Cloud" and complete the page with the settings below.

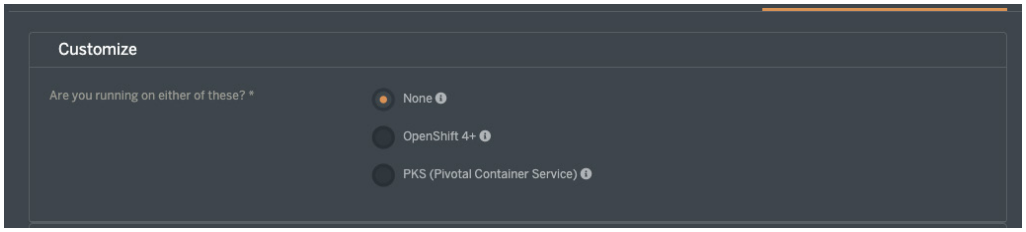


6. On the "Network" page, we can leave the settings at default and simply click "Next".

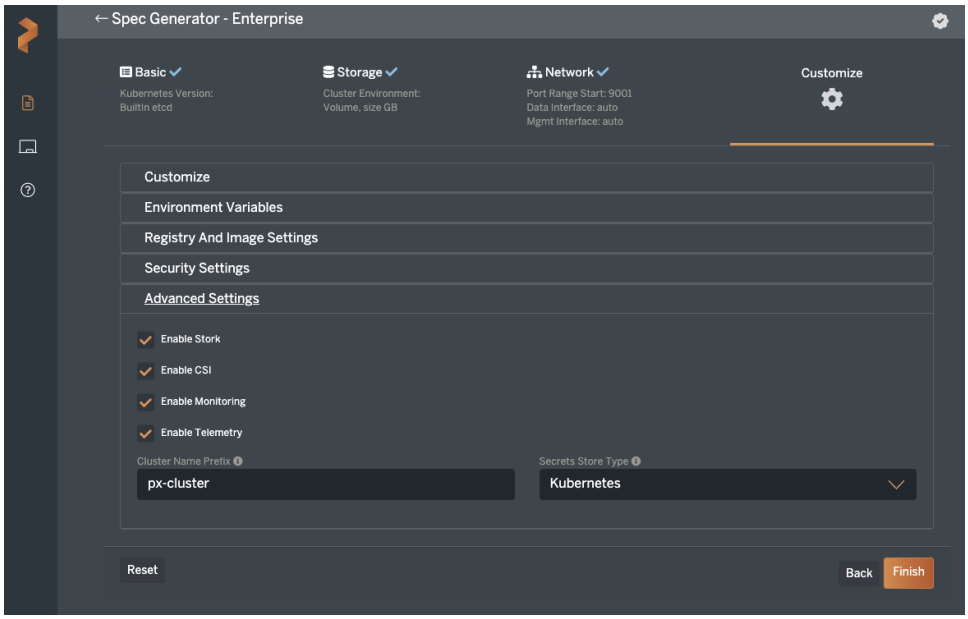


7. On the "Customize" page, we will make the following selections:

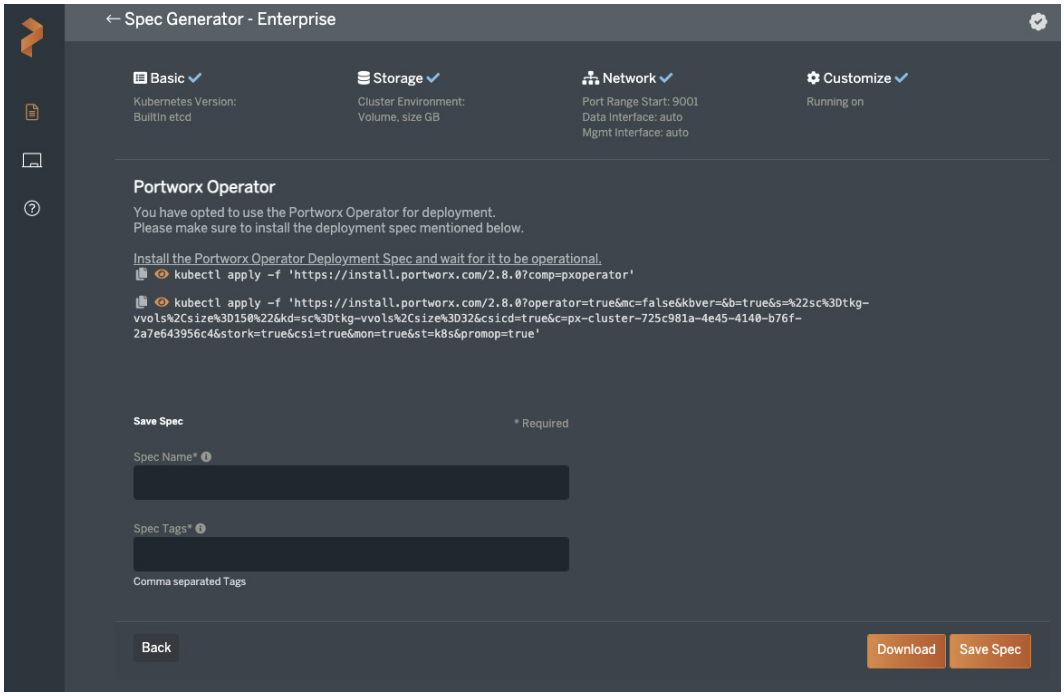
   a. Under "Customize" select None.



   b. Unless you need to use a Custom Registry or enable RBAC within Portworx, select the "Advanced Settings" section and verify all options are selected.



8. Once you complete the wizard, click "Finish". You will be asked to read and agree to the End User agreement. Once done, you are presented with the following page:



9. Click the icon that looks like two pieces of paper to copy the commands to your "Clipboard."

   a. First Deploy the Portworx Operator – You can also manually copy or type the kubectl command displayed.

      i. After deploying the operator, verify it is running by issuing `kubectl get pods -n kube-system`. Among the pods listed, you will see Portworx-operator-xxxxx with the 'x' representing the hash created by Kubernetes found in almost all pod names. When the operator is running, we can now deploy the Storage Cluster Manifest.

   b. Next copy and apply the second line to complete installing Portworx on your Tanzu cluster.

   c. Finally, it is recommended to save each spec created for future reference, so provide a name and optional tag, then click "Save Spec".

Below is an example of the manifest created at https://central.portworx.com for a vSphere with Tanzu deployment.

```
# SOURCE: https://install.portworx.com/?operator=true&mc=false&kbver=&b=true&s=%22sc%3Dtkg-
vvols%2Csize%3D150%22&kd=sc%3Dtkg-vvols%2Csize%3D32&csicd=true&c=px-cluster-725c981a-4e45-4140-b76f-
2a7e643956c4&stork=true&csi=true&mon=true&st=k8s&promop=true
kind: StorageCluster
apiVersion: core.libopenstorage.org/v1
metadata:
  name: px-cluster-725c981a-4e45-4140-b76f-2a7e643956c4
  namespace: kube-system
  annotations:
    portworx.io/install-source:
"https://install.portworx.com/?operator=true&mc=false&kbver=&b=true&s=%22sc%3Dtkg-
```

```
vvols%2Csize%3D150%22&kd=sc%3Dtkg-vvols%2Csize%3D32&csicd=true&c=px-cluster-725c981a-4e45-4140-b76f-
2a7e643956c4&stork=true&csi=true&mon=true&st=k8s&promop=true"
spec:
  image: portworx/oci-monitor:2.8.0
  imagePullPolicy: Always
  kvdb:
    internal: true
  cloudStorage:
    deviceSpecs:
    - sc=tkg-vvols,size=150
    kvdbDeviceSpec: sc=tkg-vvols,size=32
  secretsProvider: k8s
  stork:
    enabled: true
    args:
      webhook-controller: "false"
  autopilot:
    enabled: true
    providers:
    - name: default
      type: prometheus
      params:
        url: http://prometheus:9090
  monitoring:
    telemetry:
      enabled: true
    prometheus:
      enabled: true
      exportMetrics: true
  featureGates:
    CSI: "true"
```

After applying the above manifest to your Tanzu Kubernetes Cluster, open a second terminal window and issue the following command to monitor the progress of the deployment: `watch kubectl get pods -n kube-system -l name=portworx.`

When two out of three pods are running, issue the following command in your original terminal window:

```
kubectl get pods -n kube-system -l name=portworx |cut -f1 -d\  |\
while read pod; \
 do echo "$pod setting host firewall rules:";\
  kubectl exec -t $pod -n kube-system -- nsenter --mount=/host_proc/1/ns/mnt bash -c \
   "iptables -A INPUT -p tcp --match multiport --dports 9001:9020 -j ACCEPT &&\
 iptables -A INPUT -p tcp --match multiport --dports 1970 -j ACCEPT"
done
```

This will open the ports on the worker nodes used by Portworx. Once this command has finished, the installation will complete in approximately five to ten minutes.

Use `kubectl get pods -n kube-system` to check the progress of the deployment. Once all Portworx pods are running, proceed with deploying applications using Portworx as the storage orchestration layer.

The output of `kubectl get pods -n kube-system` should resemble this once Portworx has finished installing:

```
❯❯ kubectl get pods -n kube-system
NAME                                                  READY   STATUS    RESTARTS   AGE
autopilot-7b4f7f58f4-d2qx8                            1/1     Running   0          17h
calico-kube-controllers-c76779489-htl82               1/1     Running   0          18h
calico-node-bv749                                     1/1     Running   0          18h
calico-node-k4v46                                     1/1     Running   0          18h
calico-node-rfzpw                                     1/1     Running   0          18h
calico-node-smxbl                                     1/1     Running   0          18h
calico-node-sx5bd                                     1/1     Running   0          18h
calico-node-xn8c2                                     1/1     Running   0          18h
coredns-785f4496d4-4bmjw                              1/1     Running   0          18h
coredns-785f4496d4-nt2gh                              1/1     Running   0          18h
docker-registry-tkc-cluster01-control-plane-6qx9v     1/1     Running   0          18h
docker-registry-tkc-cluster01-control-plane-fl8hz     1/1     Running   0          18h
docker-registry-tkc-cluster01-control-plane-ggl7n     1/1     Running   0          18h
docker-registry-tkc-cluster01-workers-brdgw-57f94bf685-gptxm   1/1     Running   0          18h
docker-registry-tkc-cluster01-workers-brdgw-57f94bf685-sstth   1/1     Running   0          18h
docker-registry-tkc-cluster01-workers-brdgw-57f94bf685-v2vbc   1/1     Running   0          18h
etcd-tkc-cluster01-control-plane-6qx9v                1/1     Running   0          18h
etcd-tkc-cluster01-control-plane-fl8hz                1/1     Running   0          18h
etcd-tkc-cluster01-control-plane-ggl7n                1/1     Running   0          18h
kube-apiserver-tkc-cluster01-control-plane-6qx9v      1/1     Running   0          18h
kube-apiserver-tkc-cluster01-control-plane-fl8hz      1/1     Running   0          18h
kube-apiserver-tkc-cluster01-control-plane-ggl7n      1/1     Running   0          18h
kube-controller-manager-tkc-cluster01-control-plane-6qx9v   1/1     Running   1          18h
kube-controller-manager-tkc-cluster01-control-plane-fl8hz   1/1     Running   0          18h
kube-controller-manager-tkc-cluster01-control-plane-ggl7n   1/1     Running   0          18h
kube-proxy-72q8w                                      1/1     Running   0          18h
kube-proxy-742b8                                      1/1     Running   0          18h
kube-proxy-c82p8                                      1/1     Running   0          18h
kube-proxy-dh4sv                                      1/1     Running   0          18h
kube-proxy-qzjhp                                      1/1     Running   0          18h
kube-proxy-wghjh                                      1/1     Running   0          18h
kube-scheduler-tkc-cluster01-control-plane-6qx9v      1/1     Running   1          18h
kube-scheduler-tkc-cluster01-control-plane-fl8hz      1/1     Running   0          18h
kube-scheduler-tkc-cluster01-control-plane-ggl7n      1/1     Running   0          18h
portworx-api-9lbqs                                    1/1     Running   0          17h
portworx-api-lb2mg                                    1/1     Running   0          17h
portworx-api-twz8h                                    1/1     Running   0          17h
portworx-kvdb-4jn62                                   1/1     Running   0          17h
portworx-kvdb-jcw4k                                   1/1     Running   0          17h
portworx-kvdb-zbrpz                                   1/1     Running   0          17h
portworx-operator-65c7c7bb5b-9hhfc                    1/1     Running   0          18h
prometheus-px-prometheus-0                            3/3     Running   1          17h
px-csi-ext-5686675c58-rlpq5                           3/3     Running   0          17h
px-csi-ext-5686675c58-vzjgx                           3/3     Running   0          17h
px-csi-ext-5686675c58-vztd4                           3/3     Running   0          17h
px-lighthouse-7dc48b77c8-x9vvt                        3/3     Running   0          17h
px-prometheus-operator-8c88487bc-mktp5                1/1     Running   0          17h
stork-755f9d6f5c-j7h5s                                1/1     Running   0          17h
stork-755f9d6f5c-vcx4q                                1/1     Running   0          17h
stork-755f9d6f5c-zcwvc                                1/1     Running   0          17h
stork-scheduler-988d5dbdb-cj5rv                       1/1     Running   0          17h
stork-scheduler-988d5dbdb-j6v69                       1/1     Running   0          17h
stork-scheduler-988d5dbdb-vlrbb                       1/1     Running   0          17h
tkc-cluster01-7hmjx                                   3/3     Running   0          17h
tkc-cluster01-h6lr8                                   3/3     Running   0          17h
tkc-cluster01-qtqn6                                   3/3     Running   0          17h
```

Figure 5: The output of `kubectl get pods -n kube-system`

Deploying Azure Arc-enabled Data Services

Prerequisites

- A client machine to install the CLI per Microsoft's installation instructions
- A context to connect the target Kubernetes cluster, which is obtained by running the kubectl vsphere login command
- Optionally, a machine with an operating system that supports a graphical user interface for installation of Azure Data Studio, e.g., Windows, Linux, or macOS
- If you are using Azure Data Studio, ensure that the Azure Arc Extension is installed.
- A Kubernetes namespace for installing the Azure Arc-enabled data services controller and database instances into.
- Azure CLI

- A Kubernetes storage class
- An Azure subscription
- An Azure resource group

Storage Class Considerations

Portworx makes data highly available across any Kubernetes cluster by using replication. This is specified at the storage class level per the example below:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
Metadata:
    name: portworx-sc
provisioner: kubernetes.io/portworx-volume
Parameters:
   repl: "2"
   io_profile: "db_remote"
allowVolumeExpansion: true
```

It is recommended that you use:

- A minimum replication factor of 2
- The io_profile of db_remote because it tunes storage access in a manner best suited to databases while factoring for data durability with replication
- An allowVolumeExpansion set to true
- A dedicated storage class for SQL Managed Server Instance backups, such that backups ultimately reside on storage which is separate from that used to store the database data and log files

Note that while Availability Groups protect data at the Managed Instance database level, all data associated with controllers, Managed Instances—both user and system databases and PostgreSQL Hyperscale instances—are fully protected by Portworx storage replication when a replication factor of two or more is specified.

The storage experience for users of Azure Arc-enabled data services can be further enhanced using performance-based storage quality-of-service, facilitated by adding io_priority to the storage class manifest:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
Metadata:
    name: portworx-sc
provisioner: kubernetes.io/portworx-volume
Parameters:
   repl: "2"
   io_profile: "db_remote"
```

```
    io_priority: <low|medium|high>
allowVolumeExpansion: true
```

The io_priority parameter can be used to direct storage classes to allocate persistent volume claims from pools of storage with different performance characteristics. For example, low-performance pools (io_priority: low) might be best suited for development purposes, and high-performance pools (io_priority: high) might be more suitable for applications running in production environments.

Deploying a Data Controller

1. Install the Azure CLI as per Microsoft's documentation.
2. Log into Azure via the Azure CLI: `az login`
3. Create an Azure resource group to deploy the controller to, this example will use AzureArc as the resource group and eastus as the region: `az group create --name AzureArc --location eastus`

   Configure the environment variables for the controller

   ```
   export AZDATA_USERNAME=azuser
   export AZDATA_PASSWORD=S0mεstr0ngP@ssw0rd1!
   ```
4. Deploy the data controller:

```
az arcdata dc create --profile-name azure-arc-kubeadm \
                     --k8s-namespace arc-ds \
                     --name ds-controller \
                     --subscription {Azure subscription ID} \
                     --resource-group AzureArc \
                     --location eastus \
                     --storage-class portworx-sc \
                     --connectivity-mode indirect \
                     --infrastructure onpremises \
                     --use-k8s
```

Once the controller has been successfully deployed the following message will appear: `Data controller successfully deployed.`

Deploying an Arc Managed SQL Server Instance

Once a data controller has been deployed, an Arc managed SQL Server instance can be deployed, this example uses portworx-sc as the storage class:

```
az sql mi-arc create -n sqlmi1 \
                     --k8s-namespace arc-ds \
                     --admin-login-secret {admin password} \
```

```
                    --dev \
                    --cores-limit 4 \
                    --cores-request 4 \
                    --memory-limit 4Gi \
                    --memory-request 4Gi \
                    --replicas 1 \
                    --storage-class-backups portworx-sc \
                    --volume-size-backups 8Gi \
                    --storage-class-data portworx-sc \
                    --volume-size-data 4Gi \
                    --storage-class-datalogs portworx-sc \
                    --volume-size-datalogs 4Gi \
                    --storage-class-logs portworx-sc \
                    --volume-size-logs 4Gi \
                    --use-k8s
```

Deploying a Postgres Hyperscale Server Group

Similarly, a Postgres Hyperscale server group can be deployed, this example also uses portworx-sc as the storage class:

```
az postgres arc-server create -n postgres-hs-sg --memory-limit "coordinator=2Gi,w=1Gi"
  --k8s-namespace arc-ds --cores-limit 8 --cores-request 8 --memory-request 1Gi
  --storage-class-data portworx-sc --storage-class-logs portworx-sc
  --storage-class-backups portworx-sc --volume-size-data 10Gi --volume-size-logs 2Gi
  --volume-size-backups 10Gi --workers 2 --use-k8s
```

Conclusion

Portworx can easily provide the Enterprise Grade data services needed to run reliable production databases built around Azure Arc in the VMware Tanzu ecosystem at any scale. Solving for speed, density, and scale, Portworx not only enables efficient provisioning, High Availability, and data that is as mobile as the containers it fuels. Portworx also provides customers a complete Disaster Recovery and Business Continuity solution. Simply add the Disaster Recovery option and enable Metro-DR for Zero RPO, or Async-DR for longer distances with a low RPO of 10 minutes. If your business just cannot be down, Portworx Enterprise paired with VMware Tanzu Kubernetes Grid service are the tools for the job.

Additional Documentation

- FastStart Azure Arc-enabled data services and BDC deployment GitHub.

Product Support

Pure and Microsoft will support their customers following each respective company's normal support process. When the need arises, the Pure and Microsoft support teams will engage each other to collaborate. Pure offers support services over the phone, by email, and through our web portal

To contact pure storage support:

- Web: pure1.purestorage.com/support
- Email: support@purestorage.com
- Phone (US): +1 (866) 244-7121 or +1 (650) 729-4088
- Phone (international): support.purestorage.com/pure1/support

Document Updates

We are always looking to improve the quality of our content and documentation and welcome your feedback. Please send us your comments at pvd-documents@purestorage.com.

Document Revisions

Rev #	Description	Date
1.0	Initial Publication	August 2021

Pure Storage, Inc.
650 Castro Street, #400
Mountain View, CA 94041