Dell EMC PowerStore: Microsoft SQL Server 2019 Big Data Clusters

February 2022

H18231.3

White Paper

Abstract

This document includes architecture and deployment guidance for Microsoft SQL Server 2019 Big Data Clusters with Dell EMC PowerStore.

Dell Technologies

Copyright

The information in this publication is provided as is. Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Copyright © 2020-0222 Dell Inc. or its subsidiaries. All Rights Reserved. Dell Technologies, Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Intel, the Intel logo, the Intel Inside logo and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries. Other trademarks may be trademarks of their respective owners. Published in the USA February 2022 H18231.3.

Dell Inc. believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

Contents

Executive summary

Overview

The Microsoft SQL Server 2019 release introduced the data platform of SQL Server 2019 Big Data Clusters. This platform has different requirements compared to traditional versions of SQL Server. This document provides recommendations, tips, and other guidelines for architecting and deploying SQL Server 2019 Big Data Clusters on Dell EMC PowerStore.

Audience

This document is intended for IT administrators, storage architects, partners, and Dell Technologies employees. This audience also includes any individuals who may evaluate, acquire, manage, operate, or design a Dell EMC networked storage environment using PowerStore systems.

Revisions

Date	Description
April 2020	Initial release: PowerStoreOS 1.0
February 2021	Legal disclaimer update
May 2021	Updated for PowerStoreOS 2.0, Kubespray, vSphere CSI, and SQL Server 2019 CU10
February 2022	Updated template

Note: This document may contain language from third-party content that is not under Dell Technologies' control and is not consistent with current guidelines for Dell Technologies' own content. When such third-party content is updated by the relevant third parties, this document will be revised accordingly.

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on this document. Contact the Dell Technologies team by email.

Author: Doug Bernhardt

Note: For links to other documentation for this topic, see the PowerStore Info Hub.

Introduction

PowerStore overview

Dell EMC PowerStore is a robust and flexible storage and compute option that is well suited for SQL Server 2019 Big Data Clusters. PowerStore achieves new levels of operational simplicity and agility. It uses a container-based microservices architecture, advanced storage technologies, and integrated machine learning to unlock the power of your data. PowerStore is a versatile platform with a performance-centric design that delivers multidimensional scale, always-on data reduction, and support for next-generation media.

PowerStore brings the simplicity of public cloud to on-premises infrastructure, streamlining operations with an integrated machine-learning engine and seamless automation. It also offers predictive analytics to easily monitor, analyze, and troubleshoot the environment. PowerStore is highly adaptable, providing the flexibility to host specialized workloads directly on the appliance and modernize infrastructure without disruption. It also offers investment protection through flexible payment solutions and data-in-place upgrades.

PowerStore allows up to four appliances to be configured into a cluster. The PowerStore clustering capability provides the power for the most demanding Big Data Cluster workloads. Since data growth is inevitable, you can non-disruptively expand a PowerStore cluster to meet capacity and performance requirements.

SQL Server 2019 Big Data Clusters overview

SQL Server 2019 introduced a groundbreaking data platform with SQL Server 2019 Big Data Clusters (BDC). Designed to address big data challenges in a unique way, BDC solves many of the traditional challenges with building big-data and data-lake environments. See an overview of SQL Server 2019 Big Data Clusters on the Microsoft page SQL Server 2019 Big Data Cluster Overview and on the GitHub page SQL Server Big Data Cluster Workshops.

In addition to the product documentation, the following subsections cover specific benefits when deploying BDC on PowerStore.

Platform choice

SQL Server 2019 BDC deploys on the Kubernetes (K8) platform. This means that several different distributions for Kubernetes are supported, and various Linux distributions run Kubernetes. While you can deploy SQL Server 2019 BDC either in the public cloud or on-premises, this paper focuses on the PowerStore on-premises deployment. Dell Technologies also provides many Kubernetes hosting platforms and validated designs, depending on the required solution, besides the design addressed in this paper. Regardless of the deployment, cluster management and user experience are largely the same.

For administrators and IT professionals transitioning from Microsoft SQL Server on Windows Server, the Kubernetes platform can make the transition to SQL Server BDC a bit daunting. At the time of publication, there are over 65 certified Kubernetes offerings from the Cloud Native Computing Foundation. Also, the Kubernetes platform is rapidly evolving, and updates are generally published on a quarterly basis. These factors can make finding, setting up, and running a solution extremely challenging.

Dell Technologies has conquered this challenge by providing step-by-step instructions on how to set up and deploy a Big Data Cluster on PowerStore X models. The PowerStore X AppsOn feature provides a fully integrated compute and storage cluster that allows simplified deployment and management. This process is fully documented in PowerStore X deployment examples.

Scale

When planning a big data environment, scaling can sometimes be an afterthought. When scalability is not planned for an environment that will inevitably grow, this scenario can create problems in the future. SQL Server 2019 BDC software is designed with scalability in mind. The default installation creates a cluster of three nodes, enabling performance and scalability from the start. Using proven components such as SQL Server, Apache Spark, and Kubernetes provides massive power and scale. To add power to the Kubernetes cluster, just add nodes to the cluster. This ability complements the clustered scale-out architecture of PowerStore which allows up for four appliances in a cluster, making PowerStore an excellent choice for this solution.

Deployment

Building out a big-data environment typically requires defining a stack of products that provide the required capabilities. It also involves configuring multiple components such as Apache Hadoop and Spark, and selecting and installing monitoring and analytical components. SQL Server 2019 BDC simplifies a complex deployment process. Using a containerized architecture on the Kubernetes platform can simplify deployment, since Kubernetes manages networking, resiliency, and load balancing. The SQL Server 2019 BDC installation tools enable deploying an entire BDC cluster on Kubernetes with a single command. The capabilities of PowerStore and Ansible support allow automated deployment of Kubernetes clusters.

Working with Big Data Clusters

Several client tools exist for working with and managing BDC. Some tools are familiar such as Microsoft Azure Data Studio, and some tools are new such as Azure Data CLI (azdata) which is used for deployment of BDC. A complete list of client tools and installation instructions is discussed on the Microsoft site Install big data tools - SQL Server Big Data Clusters.

Data movement and external data sources

Typically, in big-data and data-analytics environments, data must be prepared for analysis. Often, this preparation includes data extraction, transformation, and load (ETL) processes in a separate data store. These processes can be expensive and time consuming in terms of development, maintenance, and administration. The capabilities of SQL Server 2019 BDC enable choice in how to analyze data and access data with expanded PolyBase capabilities. You can use BDC as a data store and also to analyze data where it resides. This data could reside in existing relational databases, Hadoop clusters, or unstructured storage. This BDC capability enables scaling compute and storage separately, horizontally, and dynamically. With the abilities of PowerStore X models, you can place hosts running external data sources on the same cluster to optimize performance.

Terminology

The following table provides definitions for some of the terms that are used in this document.

Table 1. Terminology

Term	Definition
Appliance	Solution containing a base enclosure and attached expansion enclosures. The size of an appliance could be only the base enclosure or the base enclosure plus expansion enclosures.
Node	The component within the base enclosure that contains processors and memory. Each appliance consists of two nodes.
PowerStore cluster	Multiple PowerStore appliances in a single grouping. Clusters can consist of one to four appliances.
PowerStore Manager	The web-based user interface (UI) for storage management.

Deployment overview

Introduction

SQL Server 2019 BDC is a powerful data-analytics platform. It is ideal for a wide variety of big-data and data-analytics tasks including artificial intelligence (AI) and machine learning (ML). As organizations discover the various ways that BDC can be deployed and used, they can define the best compute and storage requirements for specific use cases.

Physical hosts and virtual machines

SQL Server BDC requires a minimum of three nodes, and planning a cluster deployment requires determining the number of nodes to be used. Since BDC is a relatively new product and workloads can vary, sizing a cluster may require some trial and error. Cluster resizing can be challenging on physical hosts, which makes virtualization a great choice, especially for initial deployments.

You can deploy SQL Server 2019 BDC either directly on PowerStore X models using AppsOn, or on a physical or hypervisor compute platform that consumes PowerStore storage. Since PowerStore X models combine compute, network, and storage, they provide the most benefits to SQL Server 2019 BDC. PowerStore X models can easily scale, resize, and oversubscribe VMs. This ability makes it ideal for working with virtual hosts when requirements may change.

You can also use PowerStore T models to present storage to external hosts through storage volumes and VMware vSphere Virtual Volumes (vVols).

Automation

You can automate many of the PowerStore deployment features. Automating deployments not only saves time, which is important when deploying several hosts, but improves consistency by avoiding human error. Besides improving deployment, automation also aids in DevOps deployments and organizations moving toward infrastructure as code (IaC).

PowerStore supports industry-leading automation tools such as Ansible (see github.com/dell) to create new environments rapidly or to reconfigure and scale existing environments. Virtualized compute, networking, and storage in PowerStore X models simplify the complete automation of a Kubernetes cluster deployment. The deployment examples in this paper use automation in various deployment aspects.

Hypervisor architecture

The architecture of PowerStore X models provides a hypervisor-centric platform that brings storage and compute resources together to improve performance for data-centric applications. Applications such as SQL Server 2019 BDC can require high-performance compute, storage, networking, or all three elements. Data-centric applications such as SQL Server 2019 BDC can achieve efficiency by combining these resources and growing compute and storage at separate rates. This capability makes BDC an optimal choice for the PowerStore X model architecture.

After you have made initial sizing choices, you can create a template for the chosen configuration. Then, you can create VMs in the cluster based on a template to minimize manual configuration. If the goal is to create a large environment, creating templates and deploying through automation is optimal and moves the organization towards IaC.

Deploying Kubernetes

Although several distributions of Kubernetes (K8s) exist, Microsoft supports only some distributions for a Big Data Cluster deployment. See the SQL Server Big Data Clusters release notes for the complete list.

First, deploy a Linux operating system across several hosts, and then deploy Kubernetes to create a K8s cluster. After you install the Linux operating system, some customizations are typically required before you install Kubernetes and configure the cluster. Those customizations, and instructions for installing and configuring Kubernetes, are detailed in the Microsoft article Configure Kubernetes on multiple machines for SQL Server big data cluster deployments.

After completing these deployment instructions and before deploying SQL Server 2019 BDC on Kubernetes, configure persistent storage.

Configuring persistent storage

Default Kubernetes storage is ephemeral. For data to persist beyond the lifetime of a pod, which is a requirement for most data applications including BDC, persistent storage is required. For more information about data persistence in Kubernetes in the context of SQL Server 2019 BDC, see the Microsoft article Data persistence with SQL Server Big Data Clusters on Kubernetes.

Configure persistent storage for Kubernetes through the Container Storage Interface (CSI). A CSI driver implements functionality that is outlined in the CSI specification to enable storage provisioning tasks. When consuming PowerStore storage as a traditional block volume, use the Dell EMC PowerStore CSI driver. When PowerStore storage resides on VMware vVols such as PowerStore X AppsOn, use the VMware vSphere CSI driver.

CSI driver installation

PowerStore CSI driver

To provision block storage for PowerStore, first ensure that the PowerStore appliance is visible to the hosts. You must configure the Fibre Channel or iSCSI network, including multipath, when applicable. The next step differs in that the hosts are not added manually to PowerStore. During the deployment of the PowerStore CSI driver, it connects to the PowerStore appliance and adds the host entries. For information about connecting hosts to PowerStore, see the *Dell EMC PowerStore Host Connectivity Guide* on the PowerStore Info Hub.

To use the PowerStore CSI driver, download and install it into the Kubernetes cluster. For more information about the PowerStore CSI driver, including a product guide, deployment, and installation instructions, see the GitHub page for the PowerStore CSI Driver. Before installing the PowerStore CSI driver, you must configure all nodes in the Kubernetes cluster for host access to PowerStore.

After you complete the PowerStore CSI driver, a default storage class is created.

VMware vSphere CSI Driver

When using PowerStore X AppsOn or consuming PowerStore vVols in an external vSphere cluster, use the vSphere CSI Driver. The vSphere CSI Driver implements the CSI specification to provision storage based on a vSphere Storage Policy. See more details about the vSphere CSI Driver in the article Introduction · vSphere CSI Driver (k8s.io). In the context of PowerStore, a vVol datastore is presented to vSphere, then a vSphere Storage Policy that contains the PowerStore vVol is referenced when deploying the Kubernetes Storage Class.

Creating a vSphere CSI Storage Class is an additional step after deployment. An example definition of a vSphere Storage Class is in Appendix A: Sample configuration files.

Kubernetes Storage Class

Configuring persistent storage creates a StorageClass within the Kubernetes cluster. This StorageClass is used for dynamic storage volume provisioning with K8s and is used during the deployment of SQL Server BDC. During BDC installation, you can specify the StorageClass either as an input parameter or in a configuration file, depending on the installation method. For complete instructions about deploying SQL Server 2019 BDC, see the Microsoft article How to deploy SQL Server Big Data Clusters on Kubernetes > Deployment overview section.

Big Data Cluster deployment

After you configure the K8s environment and persistent storage, you can deploy a Microsoft Big Data Cluster. Use the Azdata tool to deploy the Big Data Cluster as a set of resources into a K8s namespace. As part of this deployment, all storage provisioning is completed dynamically. Unlike traditional SQL Server deployments, there is no requirement to provision storage before deployment. For complete instructions, see the Microsoft article How to deploy SQL Server Big Data Clusters on Kubernetes.

PowerStore X deployment examples

Introduction

The following deployment examples use the AppsOn features of PowerStore X to provide a powerful, scalable, and flexible platform for BDC. Managing and tuning a multi-node cluster can be challenging, but PowerStore X AppsOn provides a complete storage and compute platform which simplifies deployment of clustered applications such as BDC. Also, AppsOn features automatically manage and distribute the workload across appliance nodes in the cluster. This ability ensures that cluster resources are optimally balanced, even as workload demands change.

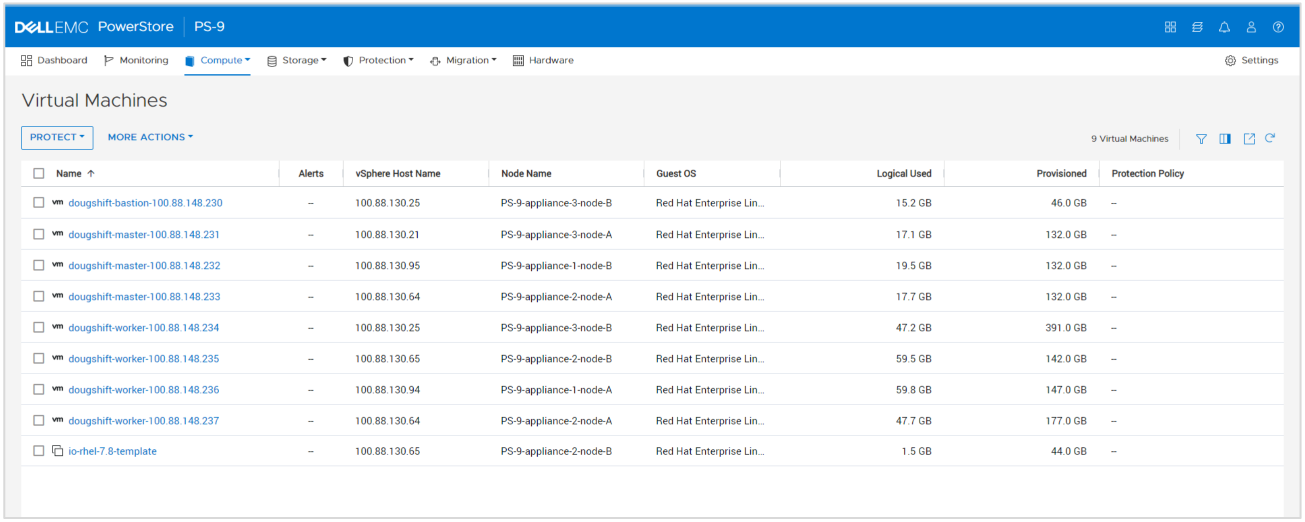
Deploying a typical on-premises Big Data Cluster on Kubernetes can be a time-consuming and manual process. The cluster requires a minimum of three nodes but can scale up to hundreds of nodes. Automating the process is key to consistent, efficient deployments. Dell Technologies has assembled and tested examples of SQL Server Big Data Cluster deployments using the capabilities of PowerStore using both OpenShift and open-source Kubernetes.

Red Hat OpenShift on PowerStore X

Red Hat OpenShift has become a leader in the Kubernetes space supporting multiple deployment models and offering powerful platform management tools and enterprise support.

For mission-critical workloads, many customers insist on enterprise support for all components in the application stack. At the time of this publication, Red Hat OpenShift is the only on-premises Kubernetes platform supported by Microsoft BDC that offers end-to-end enterprise support. Dell Technologies, Microsoft, and Red Hat offer full enterprise support for all components in the SQL Server Big Data Cluster stack, allowing you to deploy Microsoft BDC workloads with confidence.

PowerStore X AppsOn features complement the powerful tools and features of OpenShift to provide a feature-rich platform for deploying Microsoft BDC.

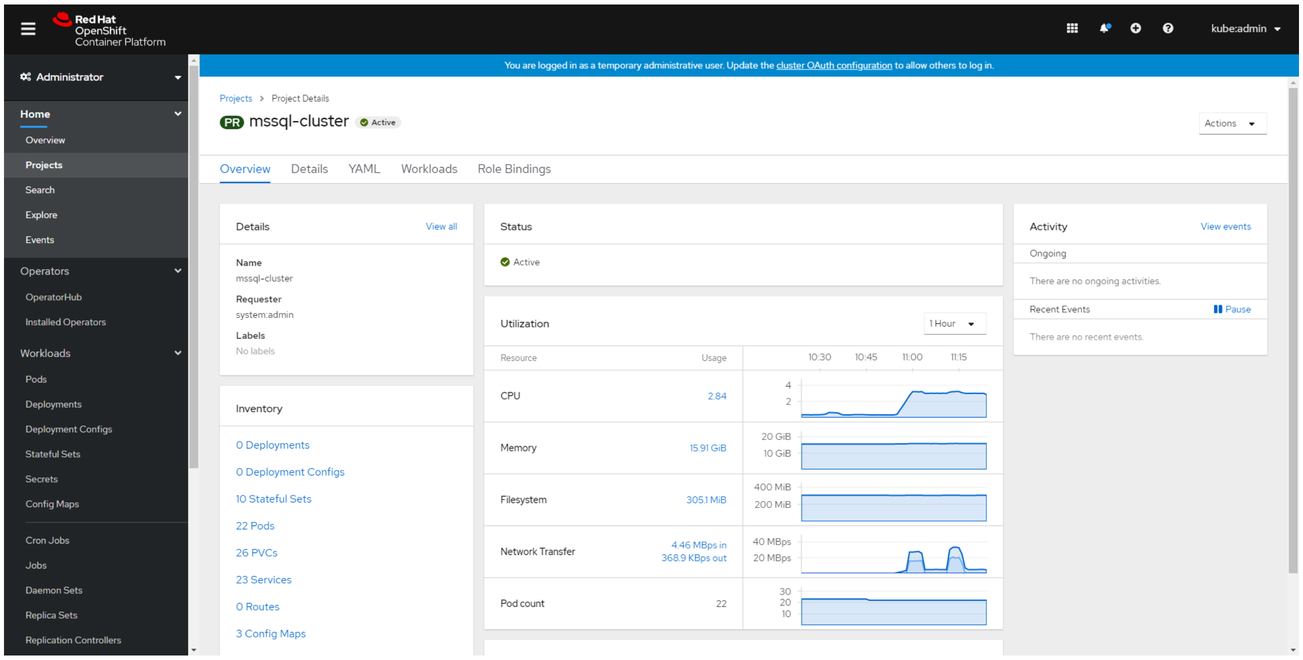


Virtual Machines

9 Virtual Machines

Name	Alerts	vSphere Host Name	Node Name	Guest OS	Logical Used	Provisioned	Protection Policy
dougshift-bastion-100.88.148.230	--	100.88.130.25	PS-9-appliance-3-node-B	Red Hat Enterprise Lin...	15.2 GB	46.0 GB	--
dougshift-master-100.88.148.231	--	100.88.130.21	PS-9-appliance-3-node-A	Red Hat Enterprise Lin...	17.1 GB	132.0 GB	--
dougshift-master-100.88.148.232	--	100.88.130.95	PS-9-appliance-1-node-B	Red Hat Enterprise Lin...	19.5 GB	132.0 GB	--
dougshift-master-100.88.148.233	--	100.88.130.64	PS-9-appliance-2-node-A	Red Hat Enterprise Lin...	17.7 GB	132.0 GB	--
dougshift-worker-100.88.148.234	--	100.88.130.25	PS-9-appliance-3-node-B	Red Hat Enterprise Lin...	47.2 GB	391.0 GB	--
dougshift-worker-100.88.148.235	--	100.88.130.65	PS-9-appliance-2-node-B	Red Hat Enterprise Lin...	59.5 GB	142.0 GB	--
dougshift-worker-100.88.148.236	--	100.88.130.94	PS-9-appliance-1-node-A	Red Hat Enterprise Lin...	59.8 GB	147.0 GB	--
dougshift-worker-100.88.148.237	--	100.88.130.64	PS-9-appliance-2-node-A	Red Hat Enterprise Lin...	47.7 GB	177.0 GB	--
io-rhel-7.8-template	--	100.88.130.65	PS-9-appliance-2-node-B	Red Hat Enterprise Lin...	1.5 GB	44.0 GB	--

Figure 1. **PowerStore Manager view of OpenShift virtual machines**



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Projects > Project Details

PR mssql-cluster Active

Details

Name
mssql-cluster
Requester
system:admin
Labels
No labels

Inventory

0 Deployments
0 Deployment Configs
10 Stateful Sets
22 Pods
26 PVCs
23 Services
0 Routes
3 Config Maps

Status

Active

Utilization

Resource	Usage
CPU	2.84
Memory	15.91 GiB
Filesystem	305.1 MiB
Network Transfer	4.46 MBps in 368.9 KBps out
Pod count	22

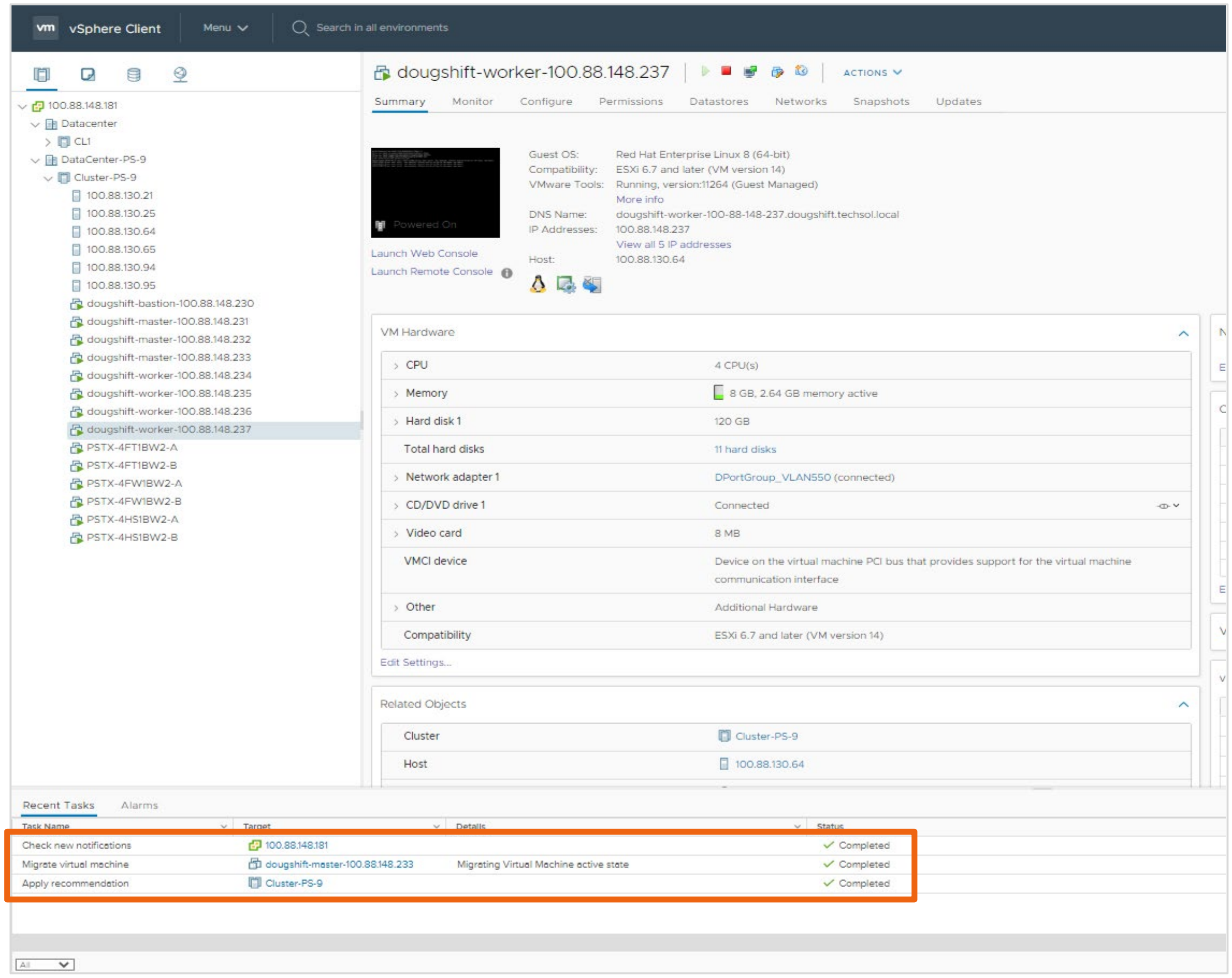
Activity

Ongoing
There are no ongoing activities.
Recent Events
There are no recent events.

Figure 2. **OpenShift management view of Big Data Cluster on PowerStore X models**

Deploy OpenShift on VMware

After the PowerStore X appliance deployment is complete, deploy the OpenShift environment. Complete details about deploying OpenShift on VMware are in the OpenShift article [Installing a cluster on vSphere](). If your organization has standard deployment scripts for deploying an OpenShift environment, you can use those scripts also. The following figure shows PowerStore X deployed as Datacenter-PS-9 within vSphere.



- 100.88.148.181
  - Datacenter
    - CL1
  - DataCenter-PS-9
    - Cluster-PS-9
      - 100.88.130.21
      - 100.88.130.25
      - 100.88.130.64
      - 100.88.130.65
      - 100.88.130.94
      - 100.88.130.95
      - dougshift-bastion-100.88.148.230
      - dougshift-master-100.88.148.231
      - dougshift-master-100.88.148.232
      - dougshift-master-100.88.148.233
      - dougshift-worker-100.88.148.234
      - dougshift-worker-100.88.148.235
      - dougshift-worker-100.88.148.236
      - dougshift-worker-100.88.148.237
      - PSTX-4FT1BW2-A
      - PSTX-4FT1BW2-B
      - PSTX-4FW1BW2-A
      - PSTX-4FW1BW2-B
      - PSTX-4HS1BW2-A
      - PSTX-4HS1BW2-B

dougshift-worker-100.88.148.237

Guest OS:	Red Hat Enterprise Linux 8 (64-bit)
Compatibility:	ESXi 6.7 and later (VM version 14)
VMware Tools:	Running, version:11264 (Guest Managed) More info
DNS Name:	dougshift-worker-100-88-148-237.dougshift.techsol.local
IP Addresses:	100.88.148.237 View all 5 IP addresses
Host:	100.88.130.64

VM Hardware

CPU	4 CPU(s)
Memory	8 GB, 2.64 GB memory active
Hard disk 1	120 GB
Total hard disks	11 hard disks
Network adapter 1	DPortGroup_VLAN550 (connected)
CD/DVD drive 1	Connected
Video card	8 MB
VMCI device	Device on the virtual machine PCI bus that provides support for the virtual machine communication interface
Other	Additional Hardware
Compatibility	ESXi 6.7 and later (VM version 14)

Related Objects

Cluster	Cluster-PS-9
Host	100.88.130.64

Task Name	Target	Details	Status
Check new notifications	100.88.148.181		✓ Completed
Migrate virtual machine	dougshift-master-100.88.148.233	Migrating Virtual Machine active state	✓ Completed
Apply recommendation	Cluster-PS-9		✓ Completed

Figure 3. **Workload balancing by automating virtual machine migration**

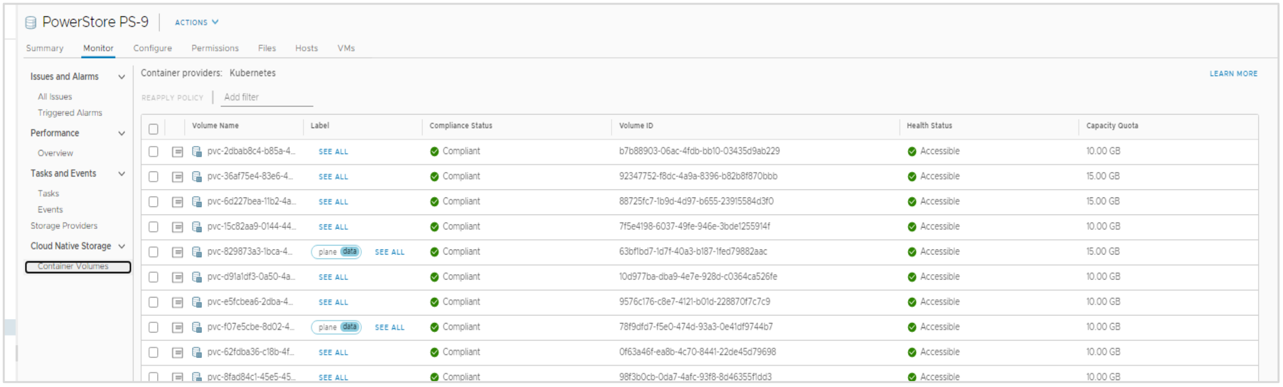
Deploy vSphere CSI Driver

We recommend using the vSphere CSI Driver for provisioning storage on PowerStore X models. Complete instructions on how to the deploy and configure the vSphere CSI Driver are in the OpenShift article Installing and configuring vSphere CSI Driver.

When configuring the vSphere CSI driver, use a VM Storage Policy that is compatible with PowerStore. The following yaml file creates the csi-vsphere Storage Class using a Storage Policy of **HighPerf**.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-vsphere
  annotations:
    storageclass.kubernetes.io/is-default-class: "false"
```

```
provisioner: csi.vsphere.vmware.com
parameters:
  StoragePolicyName: "HighPerf"
```



Container providers: Kubernetes

Volume Name	Label	Compliance Status	Volume ID	Health Status	Capacity Quota
pvc-2dbab8c4-b85a-4...	SEE ALL	Compliant	b7b88903-06ac-4fdb-bb10-03435d9ab229	Accessible	10.00 GB
pvc-36af75e4-83e6-4...	SEE ALL	Compliant	92347752-f8dc-4a9a-8396-b82b8f870bbb	Accessible	15.00 GB
pvc-6d227bea-11b2-4a...	SEE ALL	Compliant	88725fc7-1b9d-4d97-b655-23915584d3f0	Accessible	15.00 GB
pvc-15c82aa9-0144-44...	SEE ALL	Compliant	7f5e4198-6037-49fe-946e-3bde1255914f	Accessible	10.00 GB
pvc-829873a3-1bca-4...	plane data SEE ALL	Compliant	63bf1bd7-1d7f-40a3-b187-1fed79882aac	Accessible	15.00 GB
pvc-d91a1df3-0a50-4a...	SEE ALL	Compliant	10d977ba-dba9-4e7e-928d-c0364ca526fe	Accessible	10.00 GB
pvc-e5fcbea6-2dba-4...	SEE ALL	Compliant	9576c176-c8e7-4121-b01d-228870f7c7c9	Accessible	10.00 GB
pvc-f07e5cbe-8d02-4...	plane data SEE ALL	Compliant	78f9dfd7-f5e0-474d-93a3-0e41df9744b7	Accessible	10.00 GB
pvc-62fdba36-c18b-4f...	SEE ALL	Compliant	0f63a46f-ea8b-4c70-8441-22de45d79698	Accessible	10.00 GB
pvc-8fad84c1-45e5-45...	SEE ALL	Compliant	98f3b0cb-0da7-4afc-93f8-8d46355f1dd3	Accessible	10.00 GB

Figure 4. **PowerStore X container volumes for a Big Data Cluster deployment**

Deploy SQL Server Big Data Cluster

After you configure the Kubernetes Storage Class for PowerStore X, you can deploy the Microsoft Big Data Cluster. Complete instructions are in the Microsoft article Deploy SQL Server Big Data Clusters on OpenShift on-premises. While the instructions are detailed, pay attention to the following items.

The deployment of BDC on OpenShift is based on the OpenShift **nonroot** Security Context Constraint (SCC). As a result, you may encounter deployment issues that are likely security related. Keep this in mind and double-check the security settings when troubleshooting BDC deployment issues.

During the deployment, the azdata utility uses configuration files for deploying the Big Data Cluster. Modify the **control.json** file, and change the **className** parameter to the storage class created above and the **size** parameter to the volume size you require. PowerStore uses thin provisioning, so it is recommended to create files with ample space.

Kubernetes on PowerStore X

The PowerStore 9000X model is used for this solution to meet the demanding compute and storage requirements of BDC.

This section provides a step-by-step reference for deploying SQL Server 2019 BDC on a PowerStore X model using Ansible. Ansible is a tool that is used for software and configuration deployment and management. Since Ansible relies on SSH for configuration, its agentless architecture and minimal footprint make it easy to use. Ansible is designed to achieve idempotency, meaning that Ansible scripts are written to achieve a certain state. This characteristic is important, because in traditional scripting frameworks, scripts may require alteration or to be run as individual pieces. Ansible provides the framework to rerun, if necessary, skipping previously completed work, to achieve a specific state.

The steps in this section are customizable to create BDCs of various sizes and configurations. The configuration and deployment steps include options to customize these values.

Some basic knowledge with VMware and vSphere is assumed, such as the ability to create and clone virtual machines in vSphere. For instructions on how to complete these tasks, see VMware vSphere Documentation.

Note: Other security steps may be required in a production environment and are outside the scope of this document.

Configure Ansible environment

The first step in this deployment process is to ensure that the Ansible environment exists and is suitable for the tools and versions used. If an existing Ansible environment exists, you can skip the remainder of this section. For environments that do not use Ansible, the steps to configure an Ansible workstation environment are provided for convenience.

For the Ansible workstation that is used in this deployment process, a Red Hat Enterprise Linux 7 deployment VM is used. However, any Linux version could be used that supports the required Ansible and Python version. For this deployment, Ansible 2.9.20 and Python 2.7.5 are used. Other versions of Ansible and Python may produce different results.

The following commands install Ansible.

```
# yum install ansible
# yum install git
```

Once Ansible is installed, confirm that the version is correct:

```
# ansible --version
```

Install pip and the required libraries.

```
# curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
# python get-pip.pypip install requests
# pip install PyVmomi
```

Create template VM

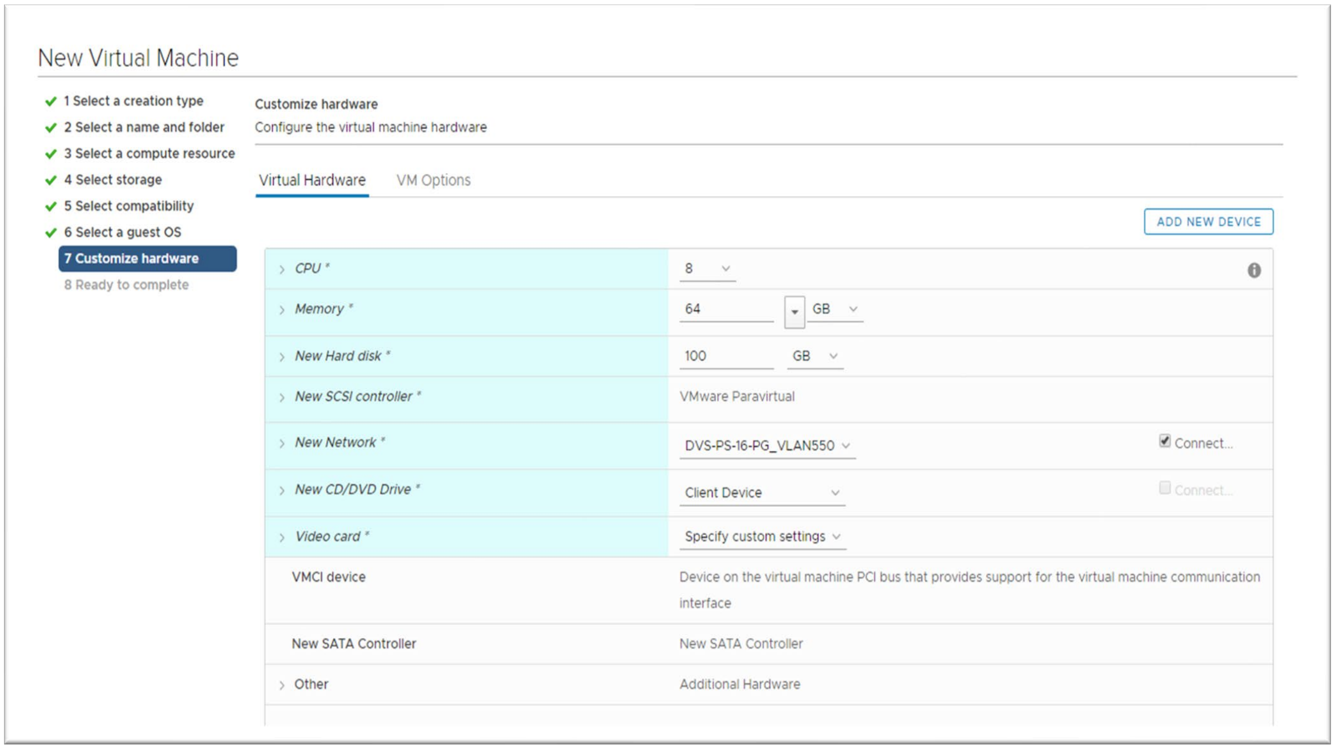
The template VM is the foundation of the BDC. This template VM serves as the basis for all nodes in the cluster. This example uses a VM template with the specifications in Table 2, which are the minimum hardware requirements for SQL Server 2019 BDC. The values are specified at the time of VM creation.

In the vSphere client, select the PowerStore X cluster and create a new virtual machine.

1. Accept the defaults for compute and storage.
2. Set the **Compatibility level** to **ESXi 6.2 Update 2 and later**.
3. Under **Guest OS**, click **Linux** > **Red Hat Enterprise 7**.
4. Under **Customize hardware**, set the values shown in the following table.

Table 2. VM template values

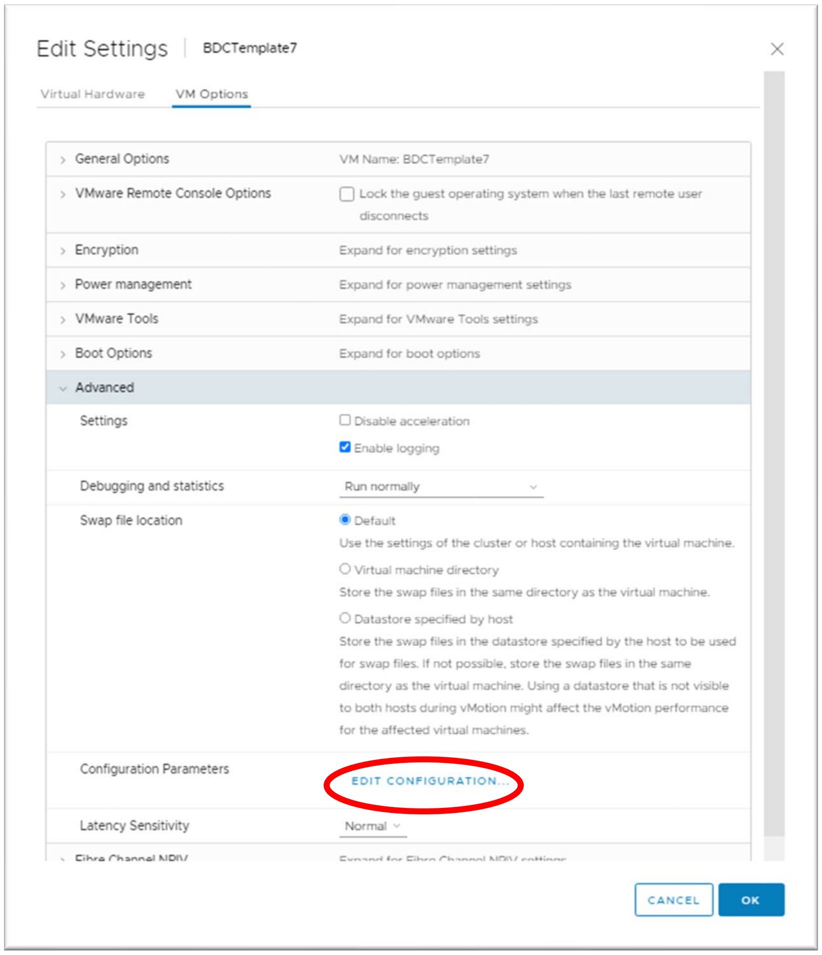
VM hardware	Value
CPU	8
Memory	64 GB
New Hard disk	100 GB
New Network	Name of port group created for workload traffic per PowerStore configuration instructions
Configuration Parameter	disk.EnableUUID=TRUE



CPU *	8
Memory *	64 GB
New Hard disk *	100 GB
New SCSI controller *	VMware Paravirtual
New Network *	DVS-PS-16-PG_VLAN550 ✓ Connect...
New CD/DVD Drive *	Client Device ☐ Connect...
Video card *	Specify custom settings
VMCI device	Device on the virtual machine PCI bus that provides support for the virtual machine communication interface
New SATA Controller	New SATA Controller
Other	Additional Hardware

Figure 5. Virtual machine settings

Next, add the **disk.EnableUUID=TRUE** parameter as shown in the following figure.

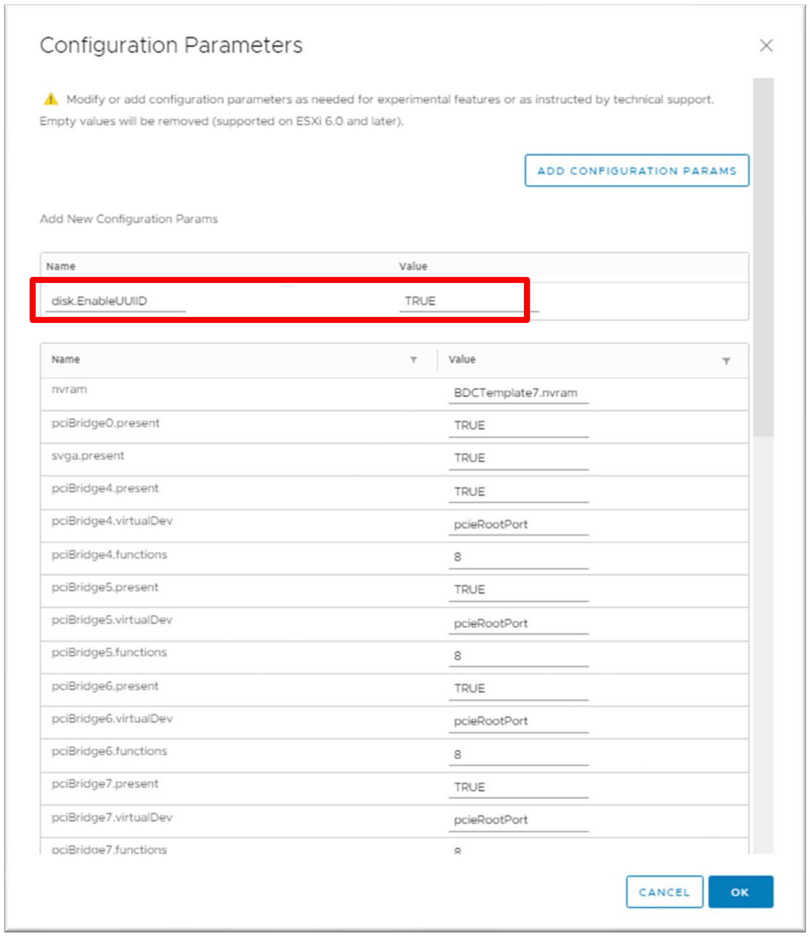


Edit Settings | BDCTemplate7

Virtual Hardware   VM Options

General Options	VM Name: BDCTemplate7
VMware Remote Console Options	☐ Lock the guest operating system when the last remote user disconnects
Encryption	Expand for encryption settings
Power management	Expand for power management settings
VMware Tools	Expand for VMware Tools settings
Boot Options	Expand for boot options
Advanced	
Settings	☐ Disable acceleration ☑ Enable logging
Debugging and statistics	Run normally
Swap file location	● Default Use the settings of the cluster or host containing the virtual machine. ○ Virtual machine directory Store the swap files in the same directory as the virtual machine. ○ Datastore specified by host Store the swap files in the datastore specified by the host to be used for swap files. If not possible, store the swap files in the same directory as the virtual machine. Using a datastore that is not visible to both hosts during vMotion might affect the vMotion performance for the affected virtual machines.
Configuration Parameters	EDIT CONFIGURATION...
Latency Sensitivity	Normal

Figure 6. **VM > Edit Settings > Edit Configuration**



Configuration Parameters

Modify or add configuration parameters as needed for experimental features or as instructed by technical support. Empty values will be removed (supported on ESXi 6.0 and later).

ADD CONFIGURATION PARAMS

Add New Configuration Params

Name	Value
disk.EnableUUID	TRUE

Name	Value
nvram	BDCTemplate7.nvram
pciBridge0.present	TRUE
svga.present	TRUE
pciBridge4.present	TRUE
pciBridge4.virtualDev	pcieRootPort
pciBridge4.functions	8
pciBridge5.present	TRUE
pciBridge5.virtualDev	pcieRootPort
pciBridge5.functions	8
pciBridge6.present	TRUE
pciBridge6.virtualDev	pcieRootPort
pciBridge6.functions	8
pciBridge7.present	TRUE
pciBridge7.virtualDev	pcieRootPort
pciBridge7.functions	8

CANCEL OK

Figure 7. **Clicking Add Configuration Params, and adding disk.EnableUUID with a value of TRUE**

Configure template VM

After you create the VM, install Red Hat Enterprise Linux 7.9 and accept the installation defaults. After installation, you must install sshpass, docker, and git prerequisites. Docker initially to pulls down the container images, and Kubespray uses sshpass when deploying the Kubernetes cluster. To pull down files from GitHub repositories, use git.

```
# yum install sshpass git
```

In the template VM, disable the firewall so the Kubespray deployment can succeed. You can reenable the firewall later if needed.

```
# sudo systemctl disable firewalld
```

Install azdata, which is a Microsoft tool that is used to configure and manage BDCs.

```
# sudo yum install -y curl
# sudo rpm --import
https://packages.microsoft.com/keys/microsoft.asc
# sudo curl -o /etc/yum.repos.d/mssql-server.repo
https://packages.microsoft.com/config/rhel/7/prod.repo
# sudo yum install azdata-cli
```

See the complete installation instructions in the Microsoft article Install azdata with yum.

After you complete the VM configuration, shut down the VM, remove the network adapter, and convert it to a VMware template.

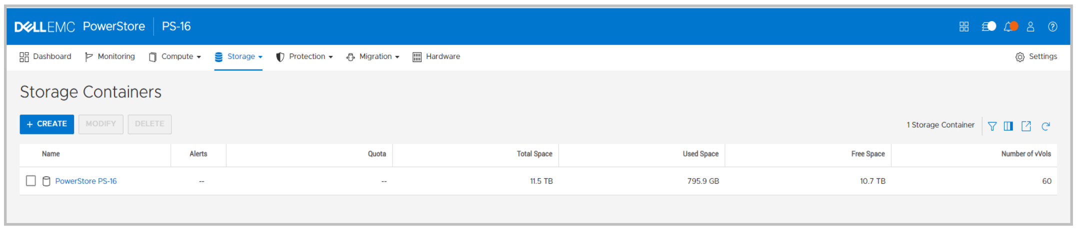
Deploy cluster VMs

BDC requires a minimum of three nodes. This example deploys five nodes. However, using the same deployment scripts, you can deploy any number of nodes of any size. Download the Ansible playbooks from the GitHub page Deploy Multiple VMware VM with Ansible. Perform the following steps on the Ansible workstation.

Use the following command to pull down the files from the GitHub repository:

```
# git clone https://github.com/cloudmaniac/ansible-deploy-vmware-
vm.git
```

Next, edit the **ansible-deploy-vmware-vm/vms-to-deploy** file, and change the hostnames, IP addresses, and datastore name to the names that match your environment. Locate the available datastores in the PowerStore Manager UI under **Storage** > **Storage Containers**.



Name	Alerts	Quota	Total Space	Used Space	Free Space	Number of vVols
PowerStore PS-16	--	--	11.5 TB	795.9 GB	10.7 TB	60

Figure 8. PowerStore X Storage Containers

Edit the **ansible-deploy-vmware-vm/answerfile.yml** file and complete all fields according to your environment. This location is where all vSphere and host configuration information is stored. In this example, **deploy_vsphere_resourcepool** is not used and can be removed. The **guest_id** must be a valid VMware guest id. Valid values are listed in the Ansible page Ansible - Manage virtual machines in vCenter.

In the following examples, **guest_memory** and **guest_cpu** are set to the recommended minimum but can be increased as needed. The value for **guest_template** is the name of the VM template that was created earlier.

```
guest_memory: ‘65536’
guest_vcpu: ‘8’
guest_template: 'RHEL-Template-BDC'
```

Edit **roles/deploy-vsphere-template/tasks/main.yml**, and under **disk:**, change **size_gb:** to **500**.

After the configuration file is complete, deploy the VMs with the following command:

```
# ansible-playbook -i vms-to-deploy deploy-kubernetes-prod.yml
```

This command instructs Ansible to run the deploy.yml playbook for the list of VMs in the vms-to-deploy file.

Deploy Kubernetes using Kubespray

Kubespray is a flexible way to deploy a Kubernetes cluster using Ansible. Most of the complexity and manual steps are eliminated, which makes it ideal for repetitive or large deployments. Find complete information https://kubespray.io/.

This document uses the following steps to deploy a Kubernetes cluster with Kubespray. In this example, Kubernetes v1.20.6 was used.

1. Clone the Kubespray script repository from GitHub:

   ```
   # git clone https://github.com/kubernetes-sigs/kubespray.git
   ```

2. Install the dependencies:

   ```
   # sudo pip install -r requirements.txt
   # yum install python3
   # sudo pip3.6 install ruamel.yaml
   ```

3. Copy the sample scripts to customize:

   ```
   # cp -rfp inventory/sample inventory/mycluster
   ```

4. Update the Ansible inventory file with inventory builder.

   The following example provides five IP addresses, but any number of addresses could be used, depending on the number of VMs deployed to the cluster.

   ```
   # declare -a IPS=(xxx.xxx.xxx.221 xxx.xxx.xxx.222
   xxx.xxx.xxx.223 xxx.xxx.xxx.224 xxx.xxx.xxx.225)
   # CONFIG_FILE=inventory/mycluster/hosts.yaml python3
   contrib/inventory_builder/inventory.py ${IPS[@]}
   ```

5. Update **kubespray/inventory/mycluster/hosts.yaml** with the correct host information.

   This example passes in the root user and password in the configuration file. Other, more-secure methods of accomplishing this task are described in Kubespray documentation.

   ```
   all:
     hosts:
       node1:
         ansible_host: xxx.xxx.xxx.221
         ip: xxx.xxx.xxx.221
         access_ip: xxx.xxx.xxx.221
         ansible_ssh_user: root
         ansible_ssh_pass: SecretPassword
       node2:
         ansible_host: xxx.xxx.xxx.222
         ip: xxx.xxx.xxx.222
         access_ip: xxx.xxx.xxx.222
         ansible_ssh_user: root
         ansible_ssh_pass: SecretPassword
       node3:
   ```

```
      ansible_host: xxx.xxx.xxx.223
      ip: xxx.xxx.xxx.223
      access_ip: xxx.xxx.xxx.223
      ansible_ssh_user: root
      ansible_ssh_pass: SecretPassword
    node4:
      ansible_host: xxx.xxx.xxx.224
      ip: xxx.xxx.xxx.224
      access_ip: xxx.xxx.xxx.224
      ansible_ssh_user: root
      ansible_ssh_pass: SecretPassword
    node5:
      ansible_host: xxx.xxx.xxx.225
      ip: xxx.xxx.xxx.225
      access_ip: xxx.xxx.xxx.225
      ansible_ssh_user: root
      ansible_ssh_pass: SecretPassword
  children:
    kube-master:
      hosts:
        node1:
        node2:
    kube-node:
      hosts:
        node1:
        node2:
        node3:
        node4:
        node5:
    etcd:
      hosts:
        node1:
        node2:
        node3:
    k8s-cluster:
      children:
        kube-master:
        kube-node:
    calico-rr:
      hosts: {}
```

6. Run the following commands to deploy the Kubernetes cluster.

   ```
   # cd kubespray
   # ansible-playbook -i inventory/mycluster/hosts.yaml -b -v
   cluster.yml
   ```

7. Run the following command.

   ```
   kubectl describe nodes | grep "ProviderID"
   ```

   The output should look like the following (1 ProviderID for each K8s cluster node):

```
ProviderID:       vsphere://422E92D9-29E1-C699-6AD7-3AEC71BDF1DB
ProviderID:       vsphere://422E07C2-D29F-6B70-F5CA-B64DAAB2EBB4
ProviderID:       vsphere://422EFDA5-2AF4-B3B6-3C22-A05EA26572A7
ProviderID:       vsphere://422E2011-F8EF-39D8-D3EA-AE9436AFA9C4
ProviderID:       vsphere://422E406E-01F5-F4B9-B2A1-CDC0579C5617
```

If the ProviderID is populated, go to step 8. If it returns no output, you must patch the ProviderID value using the following steps:

a. Install jq and wget:

```
#yum install wget jq -y
```

b. Deploy govc which is used to get VM info:

```
# wget
https://github.com/vmware/govmomi/releases/download/v0.2
0.0/govc_linux_amd64.gz
# gunzip govc_linux_amd64.gz
# mv govc_linux_amd64 govc
# sudo chown root govc
# sudo chmod ug+r+x govc
# sudo mv govc /usr/local/bin/.
```

c. Run the script to patch the provider id:

```
# chmod +x patchProvider.sh
# ./patchProvider.sh
```

8. Deploy the vSphere CSI Driver. This deployment is based on the same instructions as discussed previously for OpenShift, using kubectl instead of oc (https://www.openshift.com/blog/installing-and-configuring-vsphere-csi-driver-on-openshift-4.3). It is also possible to deploy the vSphere CSI driver with Kubespray.

```
# kubectl create secret generic vsphere-config-secret --from-
file=csi-vsphere.conf --namespace=kube-system

# kubectl apply -f
https://raw.githubusercontent.com/kubernetes-sigs/vsphere-
csi-driver/master/manifests/v2.1.0/vsphere-
7.0u1/rbac/vsphere-csi-controller-rbac.yaml
# kubectl apply -f
https://raw.githubusercontent.com/kubernetes-sigs/vsphere-
csi-driver/master/manifests/v2.1.0/vsphere-
7.0u1/deploy/vsphere-csi-node-ds.yaml
# kubectl apply -f
https://raw.githubusercontent.com/kubernetes-sigs/vsphere-
csi-driver/master/manifests/v2.1.0/vsphere-
7.0u1/deploy/vsphere-csi-controller-deployment.yaml
```

Deploy SQL Server 2019 Big Data Cluster

Use the following steps to deploy a Big Data Cluster. Detailed instructions are described in the Microsoft article How to deploy SQL Server Big Data Clusters on Kubernetes.

SSH to one of the master nodes (one of the first two nodes specified previously) in the cluster, and perform the following steps:

1. Verify the Kubernetes cluster:

   ```
   # kubectl get nodes
   ```

   Output similar to the following should be displayed:

   ```
   NAME     STATUS   ROLES    AGE    VERSION
   node1    Ready    master   3d1h   v1.16.6
   node2    Ready    master   3d1h   v1.16.6
   node3    Ready    <none>   3d1h   v1.16.6
   node4    Ready    <none>   3d1h   v1.16.6
   node5    Ready    <none>   3d1h   v1.16.6
   ```

2. In the following example file, change the storage policy name (storagepolicyname: "HighPerf") to the name of the vSphere storage policy to use. Save the following to the file **sc.yaml.** This file is used to create the storage class.

   ```
   kind: StorageClass
   apiVersion: storage.k8s.io/v1
   metadata:
     name: fast
     annotations:
       storageclass.kubernetes.io/is-default-class: "true"
   provisioner: csi.vsphere.vmware.com
   parameters:
     storagepolicyname: "HighPerf"
   ```

3. Create the storage class on the Kubernetes cluster.

   ```
   # kubectl create -f sc.yaml
   ```

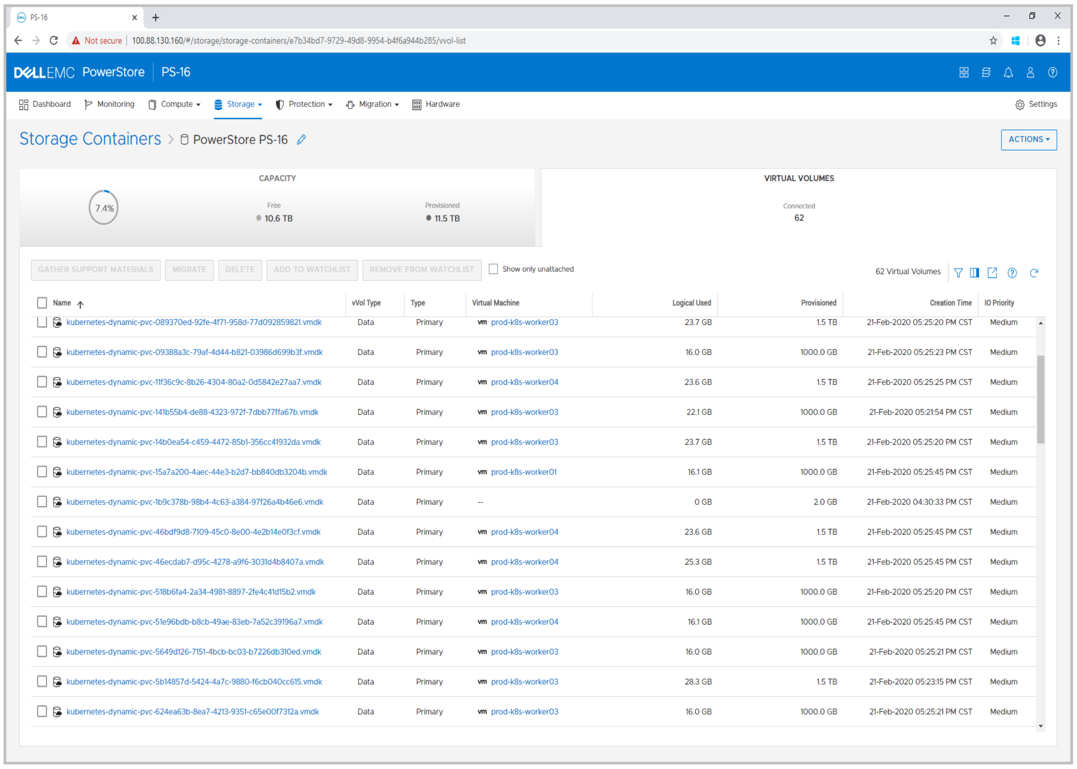
4. Run azdata to create a deployment profile.

   ```
   # azdata bdc config init --source kubeadm-dev-test --target
   custom
   ```

5. Edit the **custom/control.json** file and the size parameter to the size of the data and log file storage required.
6. Change the **className** to **fast**. This storage class was created with the **sc.yaml** file. Another file, **custom/bdc.json,** contains parameters for tuning other cluster objects. See the BDC documentation for guidance about modifying this file.
7. Deploy the Big Data Cluster on the infrastructure created previously.

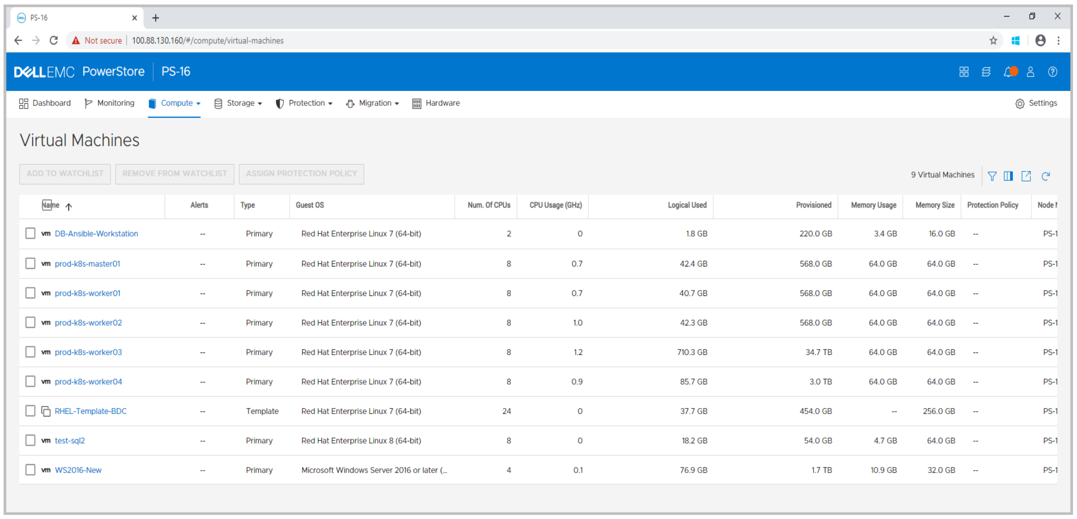
   ```
   # azdata bdc create --config-profile custom --accept-eula yes
   ```

During the deployment, the storage volumes are created dynamically on the PowerStore X model system. After the deployment is complete, several volumes are created, as shown in the following figures.



Name	vVol Type	Type	Virtual Machine	Logical Used	Provisioned	Creation Time	IO Priority
kubernetes-dynamic-pvc-089370ed-92fe-4f71-958d-77d092859821.vmdk	Data	Primary	prod-k8s-worker03	23.7 GB	1.5 TB	21-Feb-2020 05:25:20 PM CST	Medium
kubernetes-dynamic-pvc-09388a3c-79af-4d44-b821-03986d699b3f.vmdk	Data	Primary	prod-k8s-worker03	16.0 GB	1000.0 GB	21-Feb-2020 05:25:23 PM CST	Medium
kubernetes-dynamic-pvc-11f36c9c-8b26-4304-80a2-0d5842e27aa7.vmdk	Data	Primary	prod-k8s-worker04	23.6 GB	1.5 TB	21-Feb-2020 05:25:25 PM CST	Medium
kubernetes-dynamic-pvc-141b55b4-de88-4323-972f-7dbb77ffa67b.vmdk	Data	Primary	prod-k8s-worker03	22.1 GB	1000.0 GB	21-Feb-2020 05:21:54 PM CST	Medium
kubernetes-dynamic-pvc-14b0ea54-c459-4472-85b1-356cc41932da.vmdk	Data	Primary	prod-k8s-worker03	23.7 GB	1.5 TB	21-Feb-2020 05:25:20 PM CST	Medium
kubernetes-dynamic-pvc-15a7a200-4aec-44e3-b2d7-bb840db3204b.vmdk	Data	Primary	prod-k8s-worker01	16.1 GB	1000.0 GB	21-Feb-2020 05:25:45 PM CST	Medium
kubernetes-dynamic-pvc-1b9c378b-98b4-4c63-a384-97f26a4b46e6.vmdk	Data	Primary	--	0 GB	2.0 GB	21-Feb-2020 04:30:33 PM CST	Medium
kubernetes-dynamic-pvc-46bdf9d8-7109-45c0-8e00-4e2b14e0f3cf.vmdk	Data	Primary	prod-k8s-worker04	23.6 GB	1.5 TB	21-Feb-2020 05:25:45 PM CST	Medium
kubernetes-dynamic-pvc-46ecdab7-d95c-4278-a9f6-3031d4b8407a.vmdk	Data	Primary	prod-k8s-worker04	25.3 GB	1.5 TB	21-Feb-2020 05:25:45 PM CST	Medium
kubernetes-dynamic-pvc-518b6fa4-2a34-4981-8897-2fe4c41d15b2.vmdk	Data	Primary	prod-k8s-worker03	16.0 GB	1000.0 GB	21-Feb-2020 05:25:20 PM CST	Medium
kubernetes-dynamic-pvc-51e96bdb-b8cb-49ae-83eb-7a52c39196a7.vmdk	Data	Primary	prod-k8s-worker04	16.1 GB	1000.0 GB	21-Feb-2020 05:25:45 PM CST	Medium
kubernetes-dynamic-pvc-5649d126-7151-4bcb-bc03-b7226db310ed.vmdk	Data	Primary	prod-k8s-worker03	16.0 GB	1000.0 GB	21-Feb-2020 05:25:21 PM CST	Medium
kubernetes-dynamic-pvc-5b14857d-5424-4a7c-9880-f6cb040cc615.vmdk	Data	Primary	prod-k8s-worker03	28.3 GB	1.5 TB	21-Feb-2020 05:23:15 PM CST	Medium
kubernetes-dynamic-pvc-624ea63b-8ea7-4213-9351-c65e00f7312a.vmdk	Data	Primary	prod-k8s-worker03	16.0 GB	1000.0 GB	21-Feb-2020 05:25:21 PM CST	Medium

Figure 9. **Dynamically created volumes by Big Data Cluster deployment on PowerStore X model system**



Name	Alerts	Type	Guest OS	Num. Of CPUs	CPU Usage (GHz)	Logical Used	Provisioned	Memory Usage	Memory Size	Protection Policy	Node N
DB-Ansible-Workstation	--	Primary	Red Hat Enterprise Linux 7 (64-bit)	2	0	1.8 GB	220.0 GB	3.4 GB	16.0 GB	--	PS-1
prod-k8s-master01	--	Primary	Red Hat Enterprise Linux 7 (64-bit)	8	0.7	42.4 GB	568.0 GB	64.0 GB	64.0 GB	--	PS-1
prod-k8s-worker01	--	Primary	Red Hat Enterprise Linux 7 (64-bit)	8	0.7	40.7 GB	568.0 GB	64.0 GB	64.0 GB	--	PS-1
prod-k8s-worker02	--	Primary	Red Hat Enterprise Linux 7 (64-bit)	8	1.0	42.3 GB	568.0 GB	64.0 GB	64.0 GB	--	PS-1
prod-k8s-worker03	--	Primary	Red Hat Enterprise Linux 7 (64-bit)	8	1.2	710.3 GB	34.7 TB	64.0 GB	64.0 GB	--	PS-1
prod-k8s-worker04	--	Primary	Red Hat Enterprise Linux 7 (64-bit)	8	0.9	85.7 GB	3.0 TB	64.0 GB	64.0 GB	--	PS-1
RHEL-Template-BDC	--	Template	Red Hat Enterprise Linux 7 (64-bit)	24	0	37.7 GB	454.0 GB	--	256.0 GB	--	PS-1
test-sql2	--	Primary	Red Hat Enterprise Linux 8 (64-bit)	8	0	18.2 GB	54.0 GB	4.7 GB	64.0 GB	--	PS-1
WS2016-New	--	Primary	Microsoft Windows Server 2016 or later (...	4	0.1	76.9 GB	1.7 TB	10.9 GB	32.0 GB	--	PS-1

Figure 10. **Big Data Cluster nodes on PowerStore X model system**

Now, the Big Data Cluster is successfully deployed. The azdata utility shows the following output:

```
The privacy statement can be viewed at:
https://go.microsoft.com/fwlink/?LinkId=853010

The license terms for SQL Server Big Data Cluster can be viewed
at:
Enterprise: https://go.microsoft.com/fwlink/?linkid=2104292
```

```
Standard: https://go.microsoft.com/fwlink/?linkid=2104294
Developer: https://go.microsoft.com/fwlink/?linkid=2104079

Cluster deployment documentation can be viewed at:
https://aka.ms/bdc-deploy

NOTE: Cluster creation can take a significant amount of time
depending on configuration, network speed, and the number of nodes
in the cluster.

Starting cluster deployment.
Cluster controller endpoint is available at xxx.xxx.xxx.223:30080.
Cluster control plane is ready.
Data pool is ready.
Storage pool is ready.
Master pool is ready.
Compute pool is ready.
Cluster deployed successfully.
```

OpenShift on PowerStore T storage

Introduction

Red Hat OpenShift running on physical servers is a popular on-premises option for Kubernetes due to its rich feature set and enterprise support. You can also provision PowerStore T storage to physical servers (bare metal) or vSphere clusters using vVols. You can use either of these options can be a platform for OpenShift Container Platform. The CSI driver that is used depends on whether the compute platform is on physical servers or on VMware vSphere.

Physical server deployments introduce many more variables compared to PowerStore X AppsOn. The general steps in this section describe using PowerStore T models with suggested physical and virtual deployments, including links to complete instructions.

Physical server deployments

PowerStore T models provide high-performance storage for Kubernetes and BDC running on physical servers (bare metal). You can provision storage through iSCSI or Fiber Channel to K8s clusters, including OpenShift. The CSI interface simplifies storage provisioning by handling host registration and volume mapping between OpenShift and the PowerStore appliance. This capability allows you to completely manage storage through the OpenShift control plane. The Dell EMC PowerStore CSI driver orchestrates storage volume operations such as creation, mapping, resizing, snapshots, and deletion. The CSI specification is evolving rapidly. See PowerStore CSI Features for the latest information about supported features:

Deploy OpenShift

The first step to using PowerStore T storage for a physical OpenShift environment is to provision OpenShift on bare metal. Complete instructions for several bare metal deployment scenarios are on the OpenShift site Installing OpenShift. As you review the instructions, remember that storage is provisioned after you configure the OpenShift cluster, so the storage provisioning steps in the OpenShift documentation do not apply.

After you configure the OpenShift cluster, use the PowerStore CSI driver for storage provisioning.

Deploy PowerStore CSI driver

The PowerStore CSI driver is used to provision storage for Kubernetes, including OpenShift. The configuration and installation of the CSI driver performs the host mapping and the storage-volume preparation. Once the PowerStore appliance is visible on the storage network over iSCSI or Fibre Channel, install and configure the PowerStore CSI driver. See PowerStore CSI installation for more information.

Deploy SQL Server Big Data Cluster

After you deploy the PowerStore CSI driver, you can deploy the SQL Server Big Data Cluster. Complete instructions are on the Microsoft website Deploy SQL Server Big Data Clusters on OpenShift on-premises. When provisioning the cluster, use the K8s storage class created during the PowerStore CSI installation.

VMware vSphere

vSphere is a popular OpenShift deployment platform that is fully supported by Red Hat. This platform is for customers wanting to use their existing investment in VMware infrastructure while taking advantage of the rich OpenShift environment. This architecture combines the power of two leading platforms. PowerStore provides storage support through vVols to the vSphere cluster hosting OpenShift.

Deploy OpenShift

Complete instructions for deploying OpenShift on vSphere are on the OpenShift site Installing OpenShift on vSphere.

Deploy vSphere CSI Driver

After you configure the OpenShift environment on vSphere, configure the vSphere CSI driver. To deploy the vSphere CSI driver on OpenShift, follow the instructions on the OpenShift page Installing and configuring vSphere CSI Driver. During the installation, a vSphere storage class that contains the PowerStore vVols is the class that is specified in the OpenShift storage class configuration.

Deploy SQL Server Big Data Cluster

After you deploy the PowerStore CSI driver, deploy the SQL Server Big Data Cluster. For complete instructions, go to the Microsoft page Deploy SQL Server Big Data Clusters on OpenShift on-premises. When provisioning the cluster, use the K8s storage class that is created during the vSphere CSI driver installation.

Appendix A: Sample configuration files

answerfile.yml

```
# Infrastructure
# - Defines the vCenter / vSphere environment
deploy_vsphere_host: 'db-vc01.techsol.local'
deploy_vsphere_user: 'administrator@vsphere.local'
deploy_vsphere_password: 'SomePassword'
deploy_vsphere_datacenter: 'DataCenter-PS-16'
deploy_vsphere_folder: '/'
deploy_vsphere_cluster: 'Cluster-PS-16'

# Guest
# - Describes virtual machine common options
guest_network: 'DVS-PS-16-PG_VLAN550'
guest_netmask: '255.255.240.0'
guest_gateway: 'xxx.xxx.144.1'
guest_dns_server: 'xxx.xxx.144.11'
guest_domain_name: 'techsol.local'
guest_id: 'rhel7_64Guest'
guest_memory: '65536'
guest_vcpu: '8'
guest_template: 'RHEL-Template-BDC'
```

main.yml

```
---
# Deploy a VM from a template using Ansible 'vmware_guest' module
- name: Deploy VM from template
  vmware_guest:
    hostname: '{{ deploy_vsphere_host }}'
    username: '{{ deploy_vsphere_user }}'
    password: '{{ deploy_vsphere_password }}'
    validate_certs: no
    datacenter: '{{ deploy_vsphere_datacenter }}'
    cluster: '{{ deploy_vsphere_cluster }}'
    #resource_pool: '{{ deploy_vsphere_resourcepool }}'
    folder: '{{ deploy_vsphere_folder }}'
    name: '{{ inventory_hostname }}'
    state: poweredon
    guest_id: '{{ guest_id }}'
    annotation: "{{ guest_notes }}"
    disk:
    - size_gb: 500
      type: thin
      datastore: '{{ deploy_vsphere_datastore }}'
    networks:
    - name: '{{ guest_network }}'
      ip: '{{ guest_custom_ip }}'
      netmask: '{{ guest_netmask }}'
      gateway: '{{ guest_gateway }}'
      dns_servers:
        - '{{ guest_dns_server }}'
      hardware:
        memory_mb: '{{ guest_memory }}'
        num_cpus: '{{ guest_vcpu }}'
      customization:
        dns_servers:
        - '{{ guest_dns_server }}'
        domain : '{{ guest_domain_name }}'
        hostname: '{{ inventory_hostname }}'
      template: '{{ guest_template }}'
      wait_for_ip_address: no
      state: poweredon
    delegate_to: localhost
```

deploy.yml

```
---
- hosts: all
  gather_facts: false
  vars_files:
    - answerfile.yml
  roles:
     - deploy-vsphere-template
```

sc.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi.vsphere.vmware.com
parameters:
  storagepolicyname: "HighPerf"
```

patchProvider.sh

```
export GOVC_USERNAME='administrator@vsphere.local'
export GOVC_INSECURE=1
export GOVC_PASSWORD='SomePassword'
export GOVC_URL='db-vc01.techsol.local'
DATACENTER='DataCenter-PS-16'
FOLDER=''
# In my case I'm using a prefix for the VM's, so grep'ing is
necessary.
# You can remove it if the folder you are using only contains the
machines you need.
VM_PREFIX='/'
IFS=$'\n'
for vm in $(govc ls "/$DATACENTER/vm/$FOLDER" | grep $VM_PREFIX); do
  MACHINE_INFO=$(govc vm.info -json -dc=$DATACENTER -vm.ipath="/$vm"
-e=true)
  # My VMs are created on vmware with upper case names, so I need to
edit the names with awk
```

```
  VM_NAME=$(jq -r ' .VirtualMachines[] | .Name' <<< $MACHINE_INFO |
awk '{print tolower($0)}')
  # UUIDs come in lowercase, upper case then
  VM_UUID=$( jq -r ' .VirtualMachines[] | .Config.Uuid' <<<
$MACHINE_INFO | awk '{print toupper($0)}')
  echo "$VM_NAME" | tr . -

  VM_NAME=$(echo "$VM_NAME" | tr . -)
  echo "Patching $VM_NAME with UUID:$VM_UUID"
  # This is done using dry-run to avoid possible mistakes, remove
when you are confident you got everything right.
  kubectl patch node $VM_NAME -p
"{\"spec\":{\"providerID\":\"vsphere://$VM_UUID\"}}"
done
```

csi-vsphere.conf

```
[Global]
cluster-id = "csi-vsphere-cluster"
[VirtualCenter "db-vc01.techsol.local"]
insecure-flag = "true"
user = "Administrator@vsphere.local"
password = "SomePassword"
port = "443"
datacenters = "DataCenter-PS-16"
```

Appendix B: Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

Storage technical documents and videos provide expertise that helps to ensure customer success on Dell EMC storage platforms.

The PowerStore Info Hub provides detailed documentation on how to install, configure, and manage Dell EMC PowerStore systems.

For more information, see the following related resources

- VMware Documentation
- SQL Server 2019 Big Data Clusters Overview
- vSphere Cloud Provider Documentation
- Kubernetes
- SQL Server Big Data Cluster Workshops
- Dell Technologies GitHub
- Dell Technologies SQL Server Solutions
- SQL Workshops – The definitive guide from the best Microsoft minds!!!!
- Install guidance for SQL on Linux – Microsoft Docs
- Install SQL Server and create a DB on Red Hat Linux – Microsoft Docs
- Deploy SQL Server Big Data Cluster on OpenShift on-premises
- Installing RHEL on VMware
- Red Hat Enterprise Linux Server – 30 day free eval
- Performing a standard RHEL install