DELL Technologies

Technical White Paper

Dell EMC PowerStore: Microsoft SQL Server 2019 Big Data Clusters

Abstract

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

February 2021

H18231.1

Revisions

Date	Description
April 2020	Initial release: PowerStoreOS 1.0
Feb 2021	Legal disclaimer update

Acknowledgments

Author: Doug Bernhardt

Table of contents

Executive summary

The Microsoft® SQL Server® 2019 release introduced the data platform of SQL Server 2019 Big Data Clusters. This platform has very 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.

1 Introduction

Dell EMC PowerStore is a robust and flexible storage and compute option that is well suited for SQL Server 2019 Big Data Clusters. This section provides an overview for PowerStore and SQL Server 2019 Big Data Clusters.

1.1 PowerStore overview

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.

1.2 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, Big Data Clusters solve 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.

1.2.1 Platform choice

SQL Server 2019 BDC deploys on the Kubernetes platform. This means that several different distributions for Kubernetes are supported, and various Linux distributions that run Kubernetes. While SQL Server 2019 BDC can be deployed 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 Big Data Clusters a bit daunting. At the time of publication, there are 90 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 with only three commands on PowerStore X models. This process is fully documented in section 3.

1.2.2 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 Big Data Clusters have been designed with scalability in mind. The default installation creates a cluster of three nodes, enabling performance and scale from the start. Using proven components such as SQL Server, Spark, and Kubernetes provides massive power and scale. To add power to the cluster, just add nodes to the cluster. This complements the clustered scale-out architecture of PowerStore, making PowerStore a natural choice for this solution.

1.2.3 Deployment

Building out a big data environment typically requires defining a stack of products that provide the capabilities that are required. It also involves configuring multiple components such as Apache® Hadoop® and Spark®, and selecting and installing monitoring and analytical components. SQL Server 2019 Big Data Clusters 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 X models and Ansible support allow automated deployment of Kubernetes clusters.

1.2.4 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 Big Data Clusters enable choice in how to analyze data and access data with expanded PolyBase capabilities. Big Data Clusters can be used as a data store, but they can also be used 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, hosts running external data sources can be placed on the same cluster to optimize performance.

1.3 Terminology

The following list includes terms that are used with PowerStore.

PowerStore Manager: The web-based user interface (UI) for storage management.

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 appliance or more. PowerStore T model clusters are expandable by adding more appliances (up to four).

Base enclosure: The enclosure containing both nodes (node A and node B) and the NVMe drive slots.

Expansion enclosure: Enclosures that can be attached to a base enclosure to provide more SAS-based drive slots.

2 Deployment overview

SQL Server 2019 Big Data Clusters is a powerful data analytics platform. It can be used 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. The agility and flexibility of PowerStore X models provide many benefits that are outlined in the following subsections.

2.1 Benefits

Installing SQL Server 2019 BDC starts with a minimum of three nodes and can scale to hundreds of nodes. The virtualization features of PowerStore X models allow virtual machines (VMs) to be created and deployed as templates, improving efficiency and reducing manual steps. Since the nodes in a Kubernetes cluster are likely sized equally, deploying VMs from a host template is ideal for a virtualized environment such as PowerStore.

2.1.1 Automation

Many of the deployment features of PowerStore X models can be automated. Automating deployments not only saves time, which is important when deploying several hosts, but improves consistency by avoiding human error. Automation not improves deployment, but also aids in DevOps deployments and organizations moving toward infrastructure as code (IaC).

PowerStore X models also have support for industry-leading automation tools such as Ansible (see github.com/dell) to start or stop environments rapidly or reconfigure and scale existing environments. Virtualized compute, networking, and storage in PowerStore X models enable the complete automation of a Kubernetes cluster deployment. See section 3 for an example of deploying SQL Server 2019 Big Data Clusters on a PowerStore X model system.

2.1.2 Hypervisor architecture

The architecture of PowerStore X models provides a hypervisor-centric platform. It 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.

2.2 Overview

For general guidance about deploying SQL Server BDC on various platforms, see the Microsoft article How to deploy SQL Server Big Data Clusters on Kubernetes. The following subsections provide extra guidance for deploying BDC on PowerStore.

2.2.1 Deploying Kubernetes

Kubernetes supports most major distributions of Linux®. The Linux distribution that is used depends on the persistent storage method that is chosen. After the Linux operating system is installed, some customizations are required before installing Kubernetes and configuring 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, persistent storage must be configured.

2.2.2 Configuring persistent storage

The options for Kubernetes persistent storage on PowerStore are discussed in section 2.2.1. Following this guidance, configure one of the persistent storage types. For PowerStore T models, this type is the PowerStore CSI driver. For PowerStore X models, the type is either the vSphere Cloud Provider or PowerStore CSI driver. 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 cluster in Kubernetes. Deploying SQL Server 2019 Big Data Clusters

Configuring persistent storage creates a StorageClass within the Kubernetes cluster. This StorageClass is used for dynamic storage volume provisioning during the deployment of SQL Server BDC. During BDC installation, the StorageClass is specified 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.

2.3 Planning

SQL Server 2019 BDC can be either deployed on PowerStore X models or PowerStore T models, which are both supported by the PowerStore CSI Driver. Since PowerStore X models combine compute, network, and storage, they provide the most benefits to SQL Server 2019 BDC. However, PowerStore T models can also be used to present storage to external hosts. This section discusses some decisions that must be made when planning for deployment.

2.3.1 Creating virtual machines

SQL Server Big Data Clusters require a minimum of three nodes, and planning a cluster deployment requires determining the number of nodes to be used. Since BDC is a new product and workloads can vary, sizing a cluster may require some trial and error. While resizing could introduce rework when using physical hosts, PowerStore X models can easily scale, resize, and oversubscribe VMs. This ability makes it ideal for working with virtual hosts when requirements may change.

Once the initial sizing choices are made, a template can be created of the chosen configuration. Then, VMs in the cluster can then be created based on a template to minimize manual configuration. PowerStore X models also have support for tools such as Ansible for automating deployment and configuration tasks. If the goal is to create a large environment, creating templates and deploying through automation is optimal and moves the organization towards IaC.

2.3.2 Choosing a persistent storage type

Kubernetes is a rapidly evolving platform and persistent storage for data applications such as SQL Server 2019 BDC is relatively new. The evolution in Kubernetes storage support has created two options: the VMware Cloud Provider™ or the PowerStore Container Storage Interface (CSI) Driver. When planning a deployment, the method must be chosen. Each method requires configuration after the Kubernetes cluster is deployed. These two options are based on the fundamental persistent storage options available in Kubernetes.

One option is to use the vSphere Cloud Provider which uses the older in-tree method of implementing persistent storage. The other method uses the PowerStore CSI Driver which is the preferred method.

However, choosing the method is not an automatic decision. Table 1 outlines the key differences between the two options. CSI is the method of choice for Kubernetes and supports newer features such as snapshots. However, if Ubuntu is required or if it is preferred not to configure block storage protocols, the vSphere Cloud Provider may still be used.

Table 1 Comparison of vSphere Cloud Provider and PowerStore CSI Driver

Consideration	vSphere Cloud Provider	PowerStore CSI Driver
Linux distributions supported	Photon, Ubuntu®, CoreOS, CentOS, Red Hat® Enterprise Linux, SUSE Linux Enterprise Server https://vmware.github.io/vsphere-storage-for-kubernetes/documentation/prerequisites.html	Red Hat Enterprise Linux, Centos
Requires iSCSI or FC	No	Yes
Deprecated	Yes	No
Snapshot support	No	Coming soon
PowerStore Storage	VMware vSphere Virtual Volumes™ (vVols)	Block volumes

2.3.3 vSphere Cloud Provider configuration

Since the vSphere Cloud Provider uses the Kubernetes in-tree deployment model, there are no other download or installation requirements for Kubernetes. The VMware guide vSphere Storage for Kubernetes includes an overview of containers and Kubernetes, prerequisites for configuring the vSphere Cloud Provider, and configuration and troubleshooting instructions.

2.3.4 PowerStore CSI Driver installation

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 and installation instructions, see the GitHub page for the PowerStore CSI Driver. Before installing the PowerStore CSI driver, all nodes in the Kubernetes cluster must be configured for host access to PowerStore. For information about connecting hosts to PowerStore, see the *Dell EMC PowerStore Host Connectivity Guide* on the PowerStore Info Hub.

2.3.5 Kubernetes versions

There are several different distributions and deployment options for Kubernetes, and Kubernetes publishes quarterly releases for new versions. Both Dell Technologies and Microsoft are rapidly deploying new features and capabilities to use the full power of Kubernetes. As a result, version requirements may change. See the SQL Server 2019 Big Data Cluster documentation and the storage provider documentation for the latest updates on Kubernetes version requirements. For the purposes of this document, it is assumed that Kubernetes v1.15.0 or higher (https://kubernetes.io/) is used.

3 Deployment example

Deploying a typical on-premises Big Data Cluster on Kubernetes can be a time-consuming and manual process. The cluster can contain a minimum of three nodes and up to hundreds of nodes. Automating the process is key to consistent, efficient deployments. Dell Technologies has assembled and tested an IaC process for performing automated Big Data Cluster deployments in a simplified way that uses the capabilities of PowerStore X models.

The PowerStore 9000X model was chosen for this solution to meet the demanding compute and storage requirements of Big Data Clusters.

This section provides a step-by-step reference for deploying SQL Server 2019 Big Data Clusters 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. The scripts can also be rerun, if necessary, to achieve a specific state.

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

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.

3.1 Deployment preparation

This section includes preparation steps to perform before BDC deployment.

3.1.1.1 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, the remainder of this section can be skipped. For environments that do not use Ansible, the steps to configure an Ansible workstation environment are provided for convenience.

For the Ansible workstation used in this deployment process, a Red Hat Enterprise Linux 7 deployment VM was used. However, any Linux version could be used that supports the required Ansible and Python version. For this deployment, Ansible 2.7.12 and Python 2.7.15+ were 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
```

3.1.1.2 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 following specifications (see Table 2) which are the minimum hardware requirements for SQL Server 2019 Big Data Clusters. 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**, select **Linux** > **Red Hat Enterprise 7**.
4. Under **Customize hardware**, set the values shown in Table 2.

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

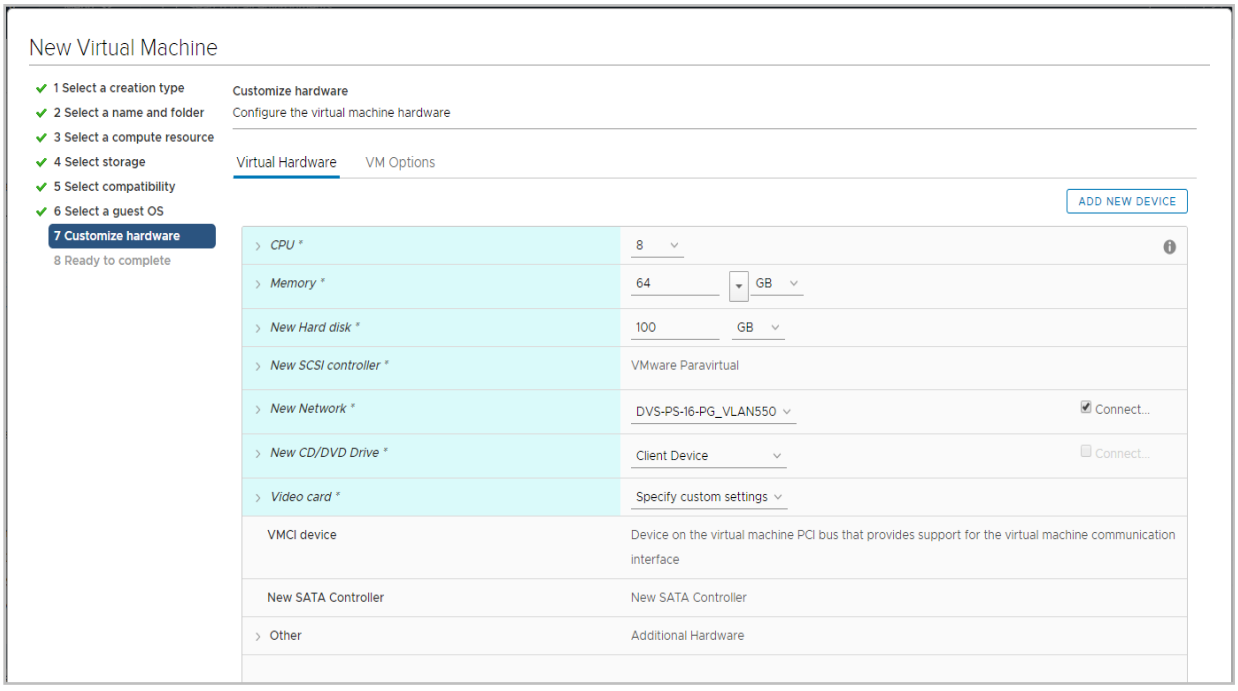

Figure 1 Virtual machine settings

3.1.1.3 Configure template VM

Once the VM is created, install Red Hat Enterprise Linux 7.7 and accept the installation defaults. After installation, sshpass docker, and git prerequisites must be installed. Docker® is used initially to pull down the container images, and sshpass is used by Kubespray when deploying the Kubernetes cluster. To pull down files from GitHub repositories, git is used.

```
# yum install sshpass docker git
```

Preloading the container images accelerates the time to deployment. Otherwise the images must be downloaded on demand as part of the deployment. While this process is completed automatically, in certain cases, the BDC deployment can time out depending on the speed of the download. To avoid timeouts and accelerate the deployment, preload the container images into the template VM. There are several ways to perform this process as outlined in the Microsoft article SQL Server Big Data Cluster Offline Deployment. In this example that is based on SQL Server 2019 CU1, the following Docker pull commands are used:

```
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-app-service-proxy:2019-
CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-control-watchdog:2019-CU1-
ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-controller:2019-CU1-
ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-dns:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-hadoop:2019-CU1-ubuntu-
16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-mleap-serving-
runtime:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-mlserver-py-runtime:2019-
CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-mlserver-r-runtime:2019-
CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-monitor-collectd:2019-CU1-
ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-monitor-
elasticsearch:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-monitor-fluentbit:2019-
CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-monitor-grafana:2019-CU1-
ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-monitor-influxdb:2019-CU1-
ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-monitor-kibana:2019-CU1-
ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-monitor-telegraf:2019-CU1-
ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-security-domainctl:2019-
CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-security-knox:2019-CU1-
ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-security-support:2019-CU1-
ubuntu-16.04
```

```
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-server:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-server-controller:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-server-data:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-ha-operator:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-ha-supervisor:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-service-proxy:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-ssis-app-runtime:2019-CU1-ubuntu-16.04
# sudo docker pull  mcr.microsoft.com/mssql/bdc/mssql-controller:2019-CU1-ubuntu-16.04
```

Note: Image tags reference Ubuntu 16.04. This image is the only choice available at the time of publication and is the correct image for Red Hat Enterprise Linux and Ubuntu.

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

```
# 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/mssql-server-2019.repo
# sudo yum install azdata-cli
```

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

Once the configuration of the VM is complete, shut down the VM, remove the network adapter, and convert it to a VMware template.

3.1.1.4 Deploy cluster VMs

Big Data Clusters require a minimum of three nodes. In this example, five nodes are deployed. However, using the same deployment scripts, any number of nodes of any size can be deployed. Download the Ansible playbooks from the following GibHub page: https://github.com/cloudmaniac/ansible-deploy-vmware-vm. The following steps are performed 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 available datastores in PowerStore UI under **Storage** > **Storage Containers**.

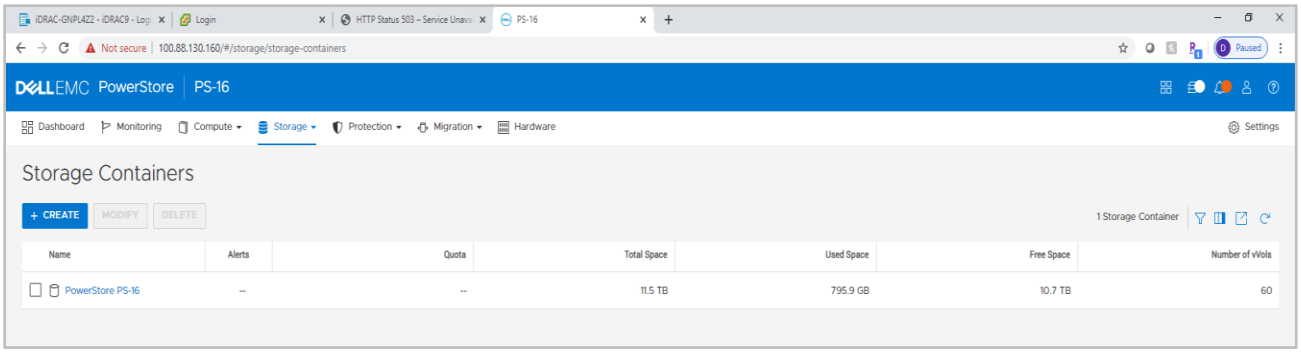

Figure 2 PowerStore X Storage Containers

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** was not used and can be removed. The **guest_id** must be a valid VMware guest id. Valid values are listed in the following Ansible documentation: https://docs.ansible.com/ansible/latest/modules/vmware_guest_module.html#parameter-guest_id.

In the following examples below, **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'

Save the following to a file named deploy.yml

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

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

Once the configuration file is complete, the VMs can be deployed with the following command:

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

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

Note: Ansible commands can be rerun without harm to achieve the required state.

3.1.1.5 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. Complete information can be found at https://kubespray.io/. The following steps are used in this guide to deploy a Kubernetes cluster with Kubespray.

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=(100.88.148.221 100.88.148.222 100.88.148.223
   100.88.148.224 100.88.148.225)
   # CONFIG_FILE=inventory/mycluster/hosts.yaml python3
   contrib/inventory_builder/inventory.py ${IPS[@]}
   ```

5. Edit the **kubespray/inventory.mycluster/group_vars/all/all.yml** file.

   For SQL Server Big Data Clusters, persistent storage is required. This scenario uses the vSphere Cloud Provider which is configured in this file. Add the following configuration entries, modifying the vSphere parameters as needed.

   ```
   kubelet_load_modules: true
   cloud_provider: "vsphere"
   vsphere_vcenter_ip: "100.88.148.181"
   vsphere_vcenter_port: 443
   vsphere_insecure: 1
   vsphere_user: "administrator@vsphere.local"
   vsphere_password: "SomePassword"
   ```

6. 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: 100.88.148.221
         ip: 100.88.148.221
         access_ip: 100.88.148.221
         ansible_ssh_user: root
         ansible_ssh_pass: doug
       node2:
         ansible_host: 100.88.148.222
         ip: 100.88.148.222
   ```

```
      access_ip: 100.88.148.222
      ansible_ssh_user: root
      ansible_ssh_pass: doug
    node3:
      ansible_host: 100.88.148.223
      ip: 100.88.148.223
      access_ip: 100.88.148.223
      ansible_ssh_user: root
      ansible_ssh_pass: doug
    node4:
      ansible_host: 100.88.148.224
      ip: 100.88.148.224
      access_ip: 100.88.148.224
      ansible_ssh_user: root
      ansible_ssh_pass: doug
    node5:
      ansible_host: 100.88.148.225
      ip: 100.88.148.225
      access_ip: 100.88.148.225
      ansible_ssh_user: root
      ansible_ssh_pass: doug
  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: {}
```

7. Run the following command to deploy the Kubernetes cluster.

```
# ansible-playbook -e cloud_provider=vsphere -i
inventory/mycluster/hosts.yaml -b -v cluster.yml
```

3.1.1.6 Deploy SQL Server 2019 CU1 Big Data Cluster

The following steps are used 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
   ```

   The following output 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 datastore name (datastore: "PowerStore PS-16") to the name of the PowerStore datastore used previously. 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:
   provisioner: kubernetes.io/vsphere-volume
   parameters:
       datastore: "PowerStore PS-16"
    fstype: ext4
   ```

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 aks-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 is the storage class that 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. Optionally, set the administrator name and password for the Big Data Cluster. If these values are not set Azdata will prompt for them.

   ```
   # export AZDATA_USERNAME=admin
   # export AZDATA_PASSWORD=SomePassword
   ```

8. 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. Once the deployment is complete, several volumes have been created. See Figure 3 and Figure 4.

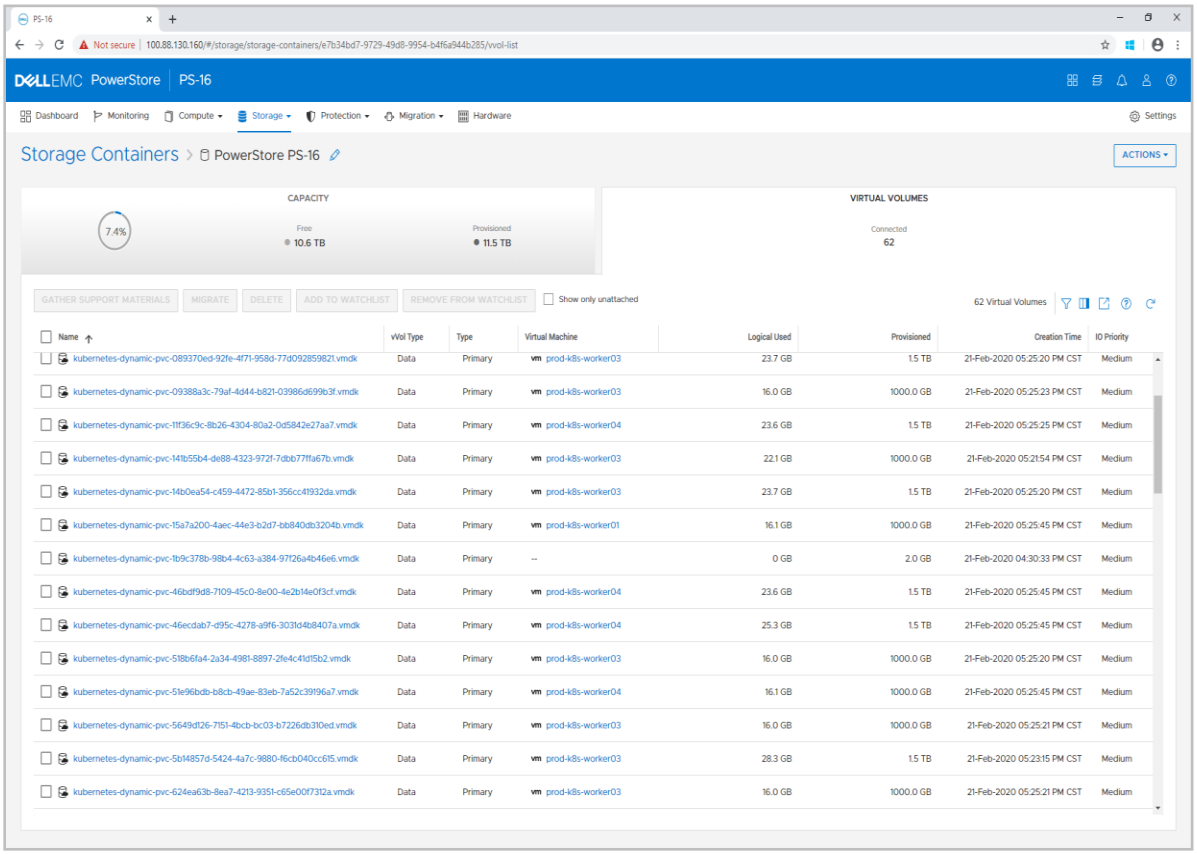

Figure 3 Dynamically created volumes by Big Data Cluster deployment on PowerStore X model system

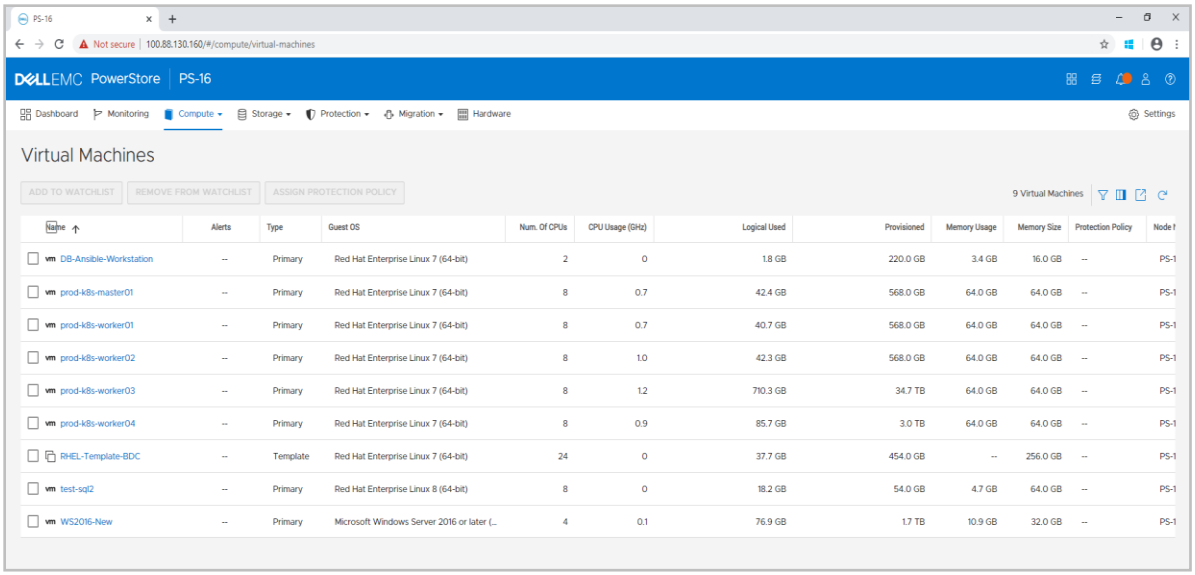

Figure 4 Big Data Cluster nodes on PowerStore X model system

At this point, 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 100.88.148.223:30080.
Cluster control plane is ready.
Data pool is ready.
Storage pool is ready.
Master pool is ready.
Compute pool is ready.
         Cluster deployed successfully.
```

3.1.2 Additional deployments

The power of the IaC deployment model demonstrated is the speed and flexibility of deploying the environment for Big Data Clusters. After the configuration parameters in the files mentioned previously are modified, the entire cluster can be redeployed with the following steps:

1. Delete the Big Data Cluster and its dynamically provisioned storage:

   ```
   # azdata bdc delete -n mssql-cluster
   ```

2. Power off and delete the cluster VMs created in vSphere.
3. Make any required changes to the configuration file, such as changing the number of nodes, memory, CPU, or storage.
4. From the Ansible workstation, execute the following commands:

   ```
   # ansible-playbook -i vms-to-deploy deploy.yml
   # ansible-playbook -e cloud_provider=vsphere -i
   inventory/mycluster/hosts.yaml -b -v cluster.yml
   ```

5. From the Kubernetes master node, run the following command:

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

A Sample configuration files

A.1 answerfile.yml

```
# Infrastructure
# - Defines the vCenter / vSphere environment
deploy_vsphere_host: '100.88.148.181'
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: '100.88.144.1'
guest_dns_server: '100.88.144.11'
guest_domain_name: 'techsol.local'
guest_id: 'rhel7_64Guest'
guest_memory: '65536'
guest_vcpu: '8'
guest_template: 'RHEL-Template-BDC'
```

A.2 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
```

A.3 deploy.yml

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

A.4 sc.yaml

```
    kind: StorageClass
    apiVersion: storage.k8s.io/v1
    metadata:
      name: fast
      annotations:
    provisioner: kubernetes.io/vsphere-volume
    parameters:
        datastore: "PowerStore PS-16"
     fstype: ext4
```

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.

B.1 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