Hewlett Packard Enterprise

Technical white paper

Check if the document is available in the language of your choice.

DEPLOY MICROSOFT SQL SERVER 2019 BIG DATA CLUSTERS ON KUBERNETES AND HPE NIMBLE STORAGE

HPE Container Platform helps bring together Kubernetes and Microsoft SQL Server 2019 Big Data Clusters

CONTENTS

EXECUTIVE SUMMARY

What is Microsoft® SQL Server 2019 Big Data Clusters? It is the SQL Server 2019 database platform optimized for containers running on Kubernetes and bundled with a strong supporting open source cast including Apache Spark, Hadoop Distributed File System (HDFS), Kibana, and Grafana.

Why should you care about Kubernetes (K8s)? Microservices running on K8s are an efficient mechanism for deploying and scaling applications. This can make for a more efficient software developer, who benefits by streamlined Continuous Integration and Continuous Deployment (CI/CD) processes or the deployment team responsible for packaging an application and putting it into production.

Why Hewlett Packard Enterprise for this solution? First, consider data persistence. Having persistent access to the data pool within this solution is key to making it enterprise-class. Hewlett Packard Enterprise delivers this by developing on the standard Kubernetes Container Storage Interface (CSI). HPE CSI Driver for Kubernetes v1.1.1 now supports HPE 3PAR StoreServ storage and HPE Primera as well as HPE Nimble Storage.

Another unique feature of this solution from Hewlett Packard Enterprise is the HPE Container Platform. Whether private, public, or hybrid cloud, the HPE Container Platform provides a multitenant, multicluster management infrastructure for Kubernetes. It is pre-integrated with HPE Nimble Storage to provide persistent volumes, while the Kubernetes Container Storage Interface (CSI) supplies a foundation for future storage arrays.

Target audience: Existing and new Microsoft SQL Server customers who are migrating to Microsoft SQL Server 2019 and are interested in integrating Kubernetes and the tools provided by Big Data Clusters. **Customers who already have container orchestration in place can skip to the relevant parts of the document.**

"78% of respondents are using Kubernetes in production, a huge jump from 58% last year."

– Cloud Native Computing Foundation (CNCF) – 2019 Survey (September/October)

SOLUTION OVERVIEW

This solution, as shown in FIGURE 1, combines a number of technology components to produce a Kubernetes cluster running Microsoft SQL Server 2019 Big Data Clusters. These pieces include:

- Bare-metal servers, in this case the HPE ProLiant DL380 Gen10.
- CentOS Linux—a community-driven, open source Linux® distribution.
- HPE Nimble Storage, providing persistent storage, integrated into Kubernetes.
- Kubernetes—a platform for automating deployment, scaling, and operations of application containers across clusters of hosts.
- HPE Container Platform—a Kubernetes-based software platform for deploying and managing Kubernetes clusters.
- Microsoft SQL Server 2019 Big Data Clusters.

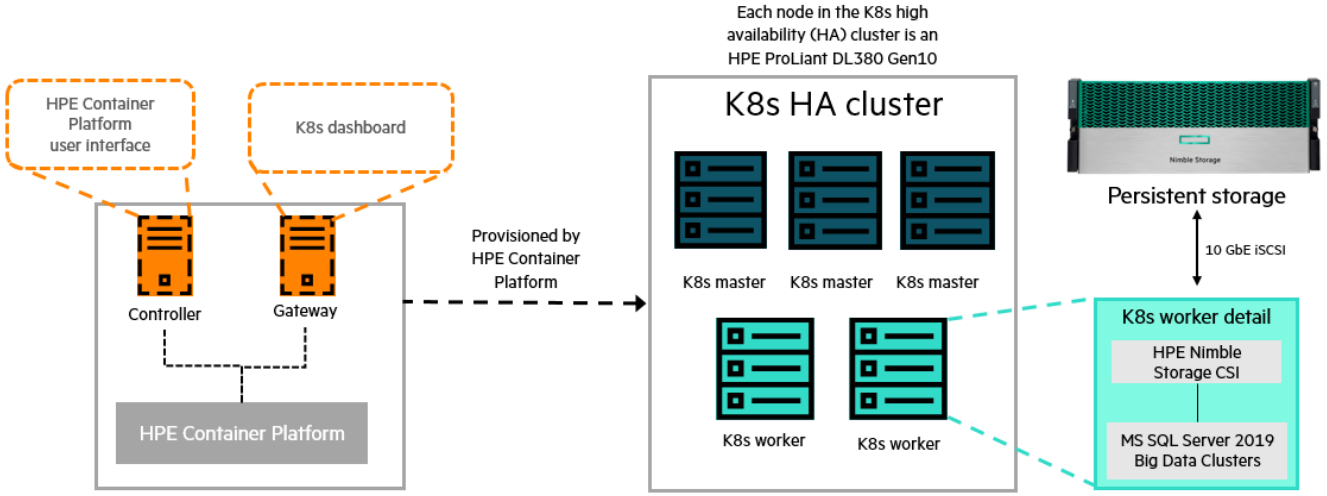

FIGURE 1. A scalable compute and storage architecture foundation for Microsoft SQL Server 2019 Big Data Clusters

SOLUTION COMPONENTS

Hardware

- HPE ProLiant DL380 Gen10 server: a bare-metal, private foundation for HPE Container Platform and Kubernetes
- HPE Nimble Storage AF1000 running NimOS 5.1.4.0: Persistent storage volumes are provisioned, bound, and managed by Kubernetes using the new Container Storage Interface (CSI) standard
- 10 Gbps Ethernet networking between K8s nodes and the HPE Nimble Storage array

Software

- CentOS 7.7 OS: open-source and binary compatible stream of Red Hat® Enterprise Linux® (RHEL)
- HPE Container Platform 5.0
- HPE Nimble Storage CSI/CSP driver, v1.0
- Kubernetes v1.17: designed by Google and maintained by CNCF

Application software

- Microsoft SQL Server 2019 Big Data Clusters CU4

BEST PRACTICES AND CONFIGURATION GUIDANCE FOR THE SOLUTION

This segment of the paper will be broken out into the following areas:

- Image servers and apply OS prerequisites
- Install and configure the HPE Container Platform and deploy Kubernetes
- Prepare persistent storage
- SQL Server 2019 Big Data Clusters customized install

NOTE

If you already have a Kubernetes environment (1.17 or newer), you can skip to the Prepare persistent storage section.

Image servers and apply OS prerequisites

All machines in this environment are imaged and updated with CentOS 7.7. HPE Container Platform also supports Red Hat®.

The following is a summary of recommendations made by the HPE Container Platform documentation.

IMPORTANT

Make sure partitions are adjusted appropriately when deploying the OS to the **Controller** and **Gateway** machines to avoid partition shuffling post-install.

1. Drive and partition layout for Controller, Gateway, and K8s hosts (see the *Storage Recommendations* portion of the HPE Container Platform documentation for more information).
   a. 500 GB disk: The boot disk needs partitioning (see FIGURE 2), **different** from what is partitioned by default during CentOS install. Start with the automatic partitioning. Then, add the **/var** partition and adjust sizes as follows:
      I. 1G <boot>
      II. 200G <root>
      III. 54G <swap>
      IV. 100G <home>
      V. 100G <var>

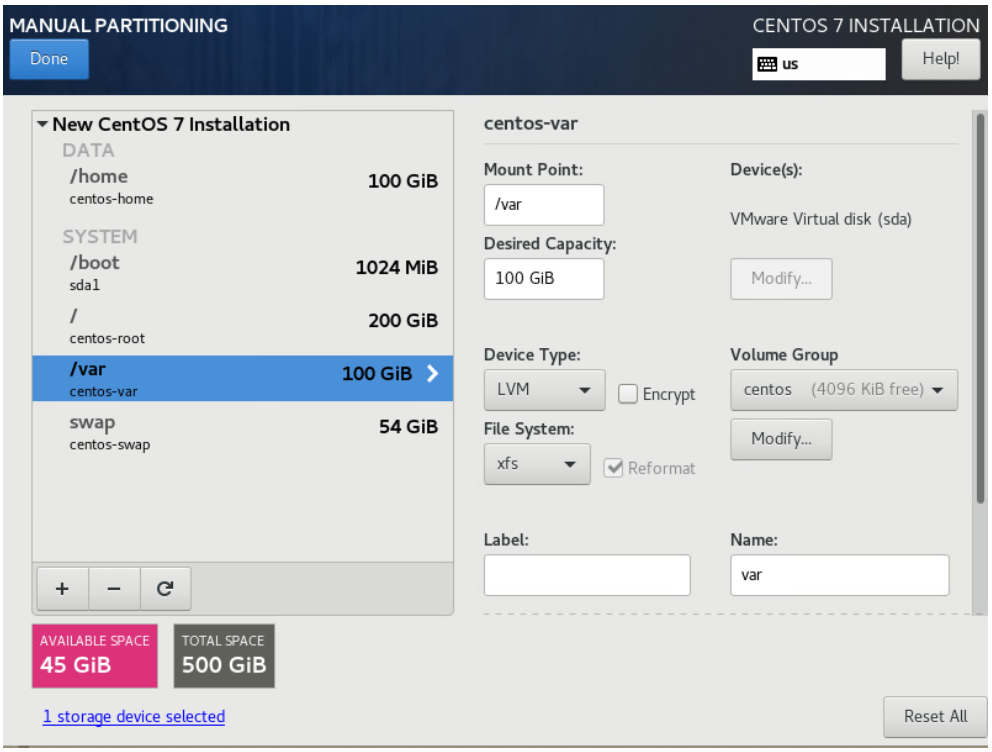

FIGURE 2. Manual partitioning layout

   b. 600 GB: Ephemeral storage (not required for HPE Container Platform Gateway)
   c. 200 GB: Persistent storage (not required for HPE Container Platform Gateway)

2. Appropriate memory and CPU (see the *Sizing Considerations* area of the HPE Container Platform documentation). In particular:
   a. Controller: 4-16 cores and 32-192 GB RAM
   b. Gateway: 8 cores and 32 GB RAM
   c. K8s hosts: 4-16 cores and 32-192 GB RAM
3. HA considerations:
   a. If HPE Container Platform HA is desired (see the *High Availability Requirements* section of the HPE Container Platform documentation), two more hosts are needed for a total of 4 hosts for the HPE Container Platform.
   b. For K8s HA, a minimum of 5 nodes are needed: 3 master nodes and 2 worker nodes.
4. Image each of the machines with CentOS or RHEL 7.7. CentOS was used in this guide.
5. DNS with entries for an HPE Container Platform **Controller** machine, HPE Container Platform **Gateway** machine, and each K8s host.
6. Ensure nameserver and search path are in `/etc/resolv.conf`.
7. Define hostname as fully qualified domain name (FQDN) on all machines.
8. Customize `/etc/profile.d/set_proxy.sh` (see APPENDIX A: Code examples).
   The `no_proxy` section is critical. In particular, IP *and* FQDN must be specified for all hosts in the cluster.
   Place the same file on each machine.
   Source this script or log out of the active session to load the variables.
9. Once the proxy environment variables are in place, use them to create the docker proxy configuration (see APPENDIX A: Code examples).
10. If applicable to your environment, specify a proxy in `/etc/yum.conf`, for example: `proxy=http://URL:PORT`.
11. Install the EPEL repository, as this is used for the Big Data Clusters installation as well as ensuring other supporting packages are present:

```
yum -y install epel-release bind-utils ntp wget
```

12. If desired, update the OS with: `yum -y update`.
13. Configure `/etc/ntpd.conf`; either add your own NTP server(s) or ensure the default/public ones are functioning.
14. Password-less SSH from the **Controller** node and all other machines is helpful. For example:

```
ssh-keygen
ssh-copy-id -i ~/.ssh/id_rsa.pub root@<hostname>
```

15. Set SELinux appropriately.
   a. The HPE Container Platform supports SELinux, but the mode **may not** change after the HPE Container Platform is installed. Reboot for any changes to take effect.
   b. The K8s hosts are currently required to have SELinux set to permissive mode.
   Edit `/etc/selinux/config` and set `SELINUX=permissive`.

Install and configure the HPE Container Platform

See the *Installation Overview* in the HPE Container Platform documentation for full details. The following is an outline of the requirements. Complete these steps before installing the HPE Container Platform components.

1. Obtain the HPE Container Platform installation bundle (see the *Bundles* area of the HPE Container Platform documentation).
2. Obtain a temporary license from the My HPE Software Center website for the HPE Container Platform, if necessary.
3. Complete pre-checks by referencing the *Using the Pre-Check Script* section of the HPE Container Platform documentation. The pre-check script can be executed on the **Controller** node as well as the **Gateway** node.
4. This guide utilized password-less SSH, so the standard install script was executed without arguments.
5. Once the HPE Container Platform UI is up, log in (default credentials are "admin/admin123") and provision the HPE Container Platform Gateway, as presented in FIGURE 3.

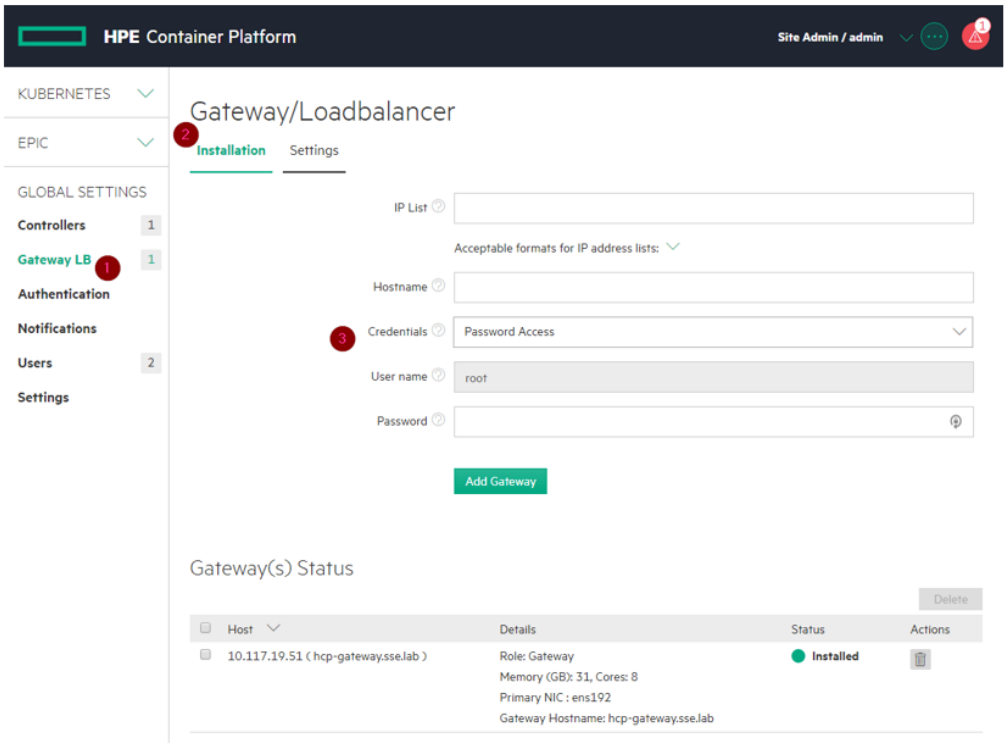

FIGURE 3. Provisioning the HPE Container Platform Gateway

While there is more that can be done with EPIC, this guide and environment will focus specifically on the Kubernetes integration. This is completed in the next step.

Deploy the Kubernetes cluster using the HPE Container Platform

Deploying a highly available K8s cluster is streamlined using the HPE Container Platform.

Add all hosts that will be used for K8s

1. Add hosts to the HPE Container Platform for use as K8s nodes (see the *Kubernetes Host Step 2: Select the Host(s)* and *Kubernetes Host Step 3: Add the Hosts(s)* portions of the HPE Container Platform documentation).
   a. Choose **Hosts** under the **KUBERNETES** section in the HPE Container Platform UI.
   b. Specify an IP range and credentials for import. This should include all hosts intended for K8s master and worker nodes. Click **Submit**.
2. If not done already, apply the HPE Container Platform license:
   a. Under **Settings** in the left-side navigation, choose **License**. Note the **Platform ID**.
   b. Log into https://myenterpriselicense.hpe.com, and find your entitlement for HPE Container Platform. Enter the platform ID in **step 3**.
   c. Download the resulting .dat file from the website.
   d. Back in the HPE Container Platform UI, click **Upload license**. You can now resume the process of adding K8s nodes.
3. Edit storage (Ephemeral and Persistent) on each host as necessary. Confirm the desired local disks were selected (see the *Kubernetes Host Step 4: Select Hard Drives* in the HPE Container Platform documentation).
4. If not already in Lockdown mode, place the HPE Container Platform into Lockdown mode (see the *Kubernetes Host Step 5: Enter Lockdown Mode* in the HPE Container Platform documentation).
5. Select all the hosts, and click the **Install** button (see the *Kubernetes Host Step 6: Add the Host(s) as Workers* in the HPE Container Platform documentation). This step will take several minutes to complete.
6. Exit Lockdown mode. Click the  icon in the top right, and choose **Exit site lockdown**.
7. As exhibited in FIGURE 4, validate the K8s installation (see the *Kubernetes Host Step 7: Validate the Worker Installation* in the HPE Container Platform documentation).

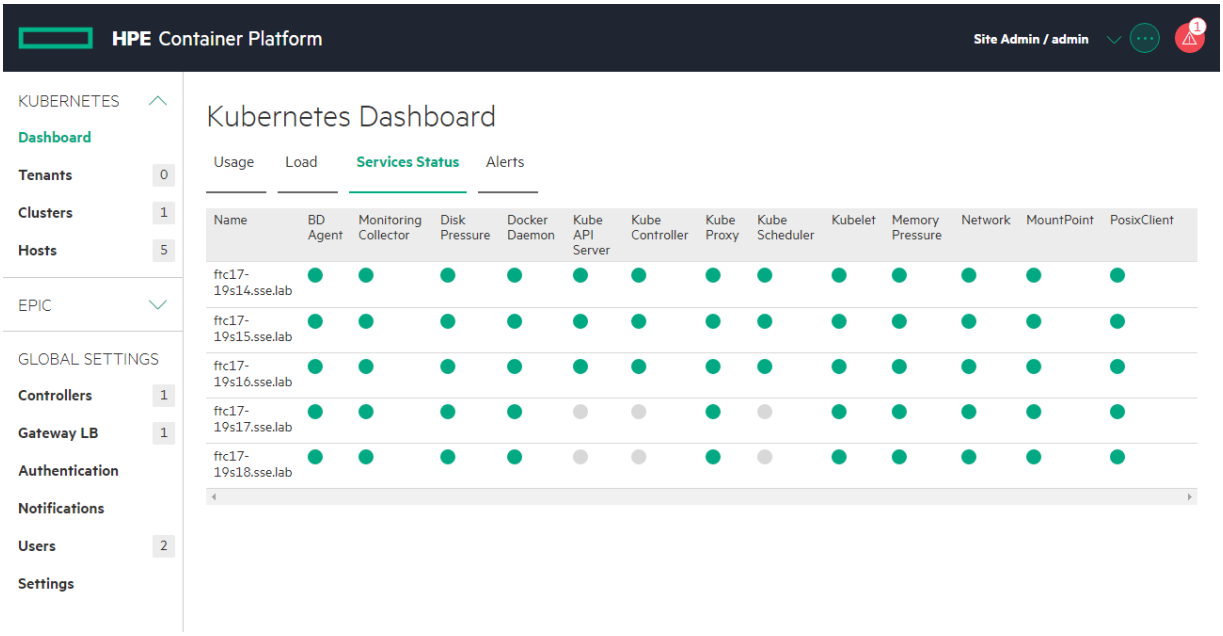

FIGURE 4. Validating installation of the K8s

Create a K8s cluster

Next, create a K8s cluster from the hosts that were added previously (see the *Creating a New Kubernetes Cluster* in the HPE Container Platform documentation). From the **KUBERNETES** menu, choose **Clusters**. Then select the **Create Kubernetes Cluster** button.

1. Step 1: Host Configurations—specify three master (for HA) and at least two worker nodes.
2. Step 2: Cluster configurations—select the K8s version and network. **Ensure you select the box to install the HPE CSI driver for K8s**.
3. Step 3: Summary—review the roles for each machine are as expected (master vs. worker).

Deploy "kubectl" to manage K8s cluster

After K8s is deployed, install `kubectl` on a machine that has network connectivity to the K8s cluster. In this environment, we chose the **hcp-controller** node. See *Kubernetes Host Requirements* in the HPE Container Platform documentation as necessary.

Log into the HPE Container Platform UI, and navigate to your K8s cluster. The fourth button from the right, as displayed in FIGURE 5, allows you to **Download Admin Kubeconfig** for your cluster.

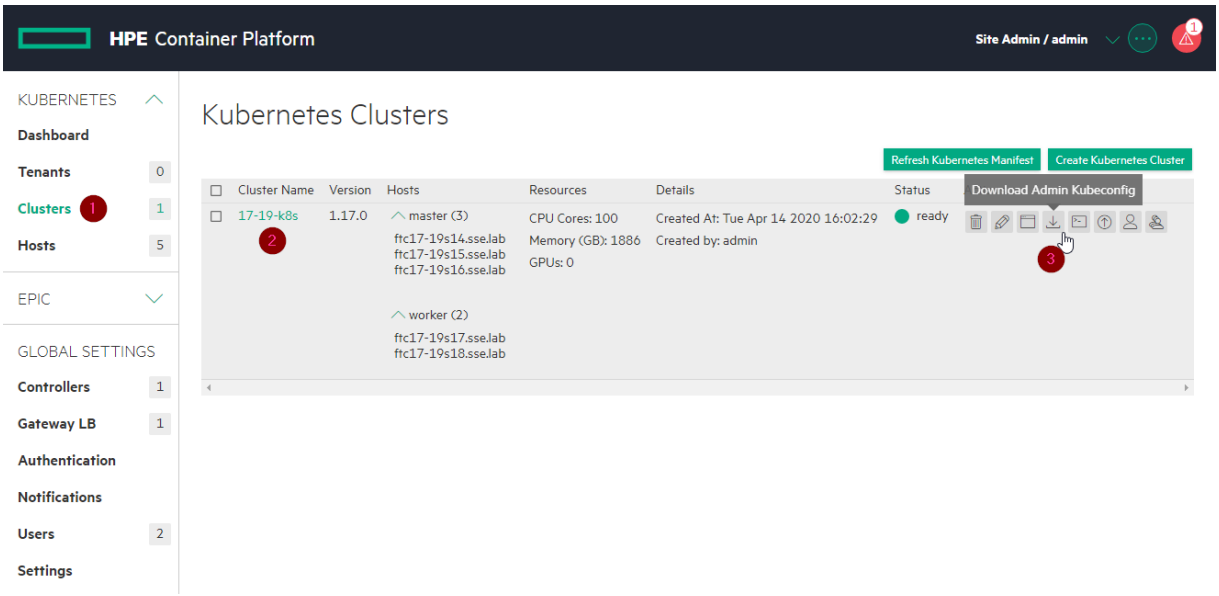

FIGURE 5. Downloading the Kubeconfig for the Kubernetes cluster

Obtain this, and save it to `/root/.kube/config`.

1. Define the K8s yum repository `/etc/yum.repos.d/kubernetes.repo`. See the script in APPENDIX A: Code examples.
2. Install the `kubectl` tool: `yum install -y kubectl --disableexcludes=Kubernetes`
3. Test that `kubectl` can run commands against your K8s cluster:

```
[root@hcp-controller ~]# kubectl cluster-info
Kubernetes master is running at https://hcp-gateway.sse.lab:10000
…
```

4. Enable `kubectl` autocomplete, if desired. See the Kubernetes Kubectl Autocomplete documentation for more information.

The K8s cluster is now running and can be configured with `kubectl`.

Visually manage K8s with the Kubernetes dashboard

The K8s cluster operational state is managed using the Kubernetes dashboard (see the Accessing the Kubernetes Dashboard in the HPE Container Platform documentation). This dashboard is provided by Kubernetes and runs locally on each cluster. Storage classes, persistent volumes, services, and secrets are examples of what can be queried using this UI.

Launch the K8s dashboard via the **Clusters** page from the left-side navigation on the HPE Container Platform UI, as shown in FIGURE 6.

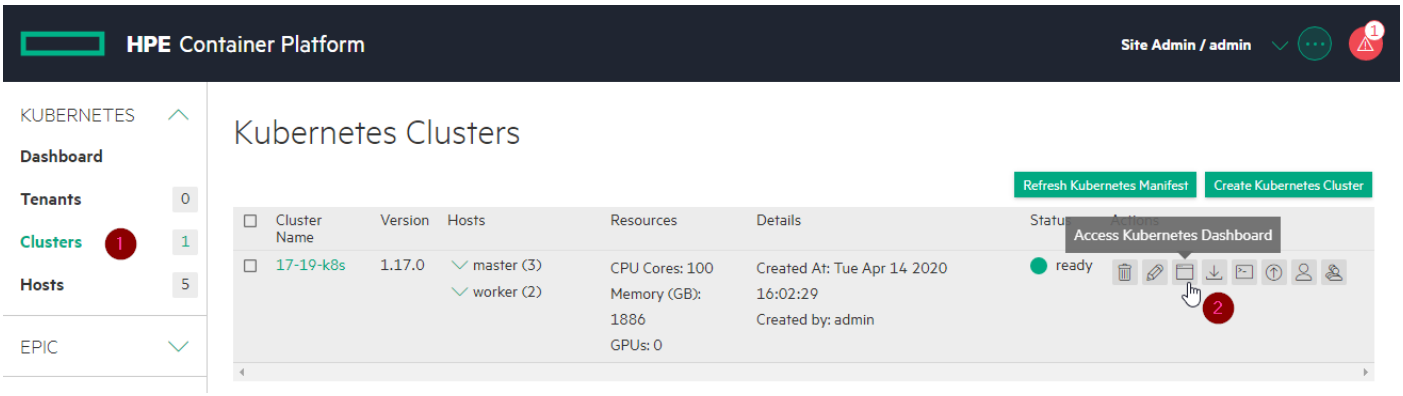

FIGURE 6. Accessing the "Kubernetes Clusters" dashboard

There is a prompt to transfer a token for authentication. This allows for easy login to the K8s dashboard.

The K8s dashboard has a default timeout of 15 minutes. This can be modified. See the *Increase (or disable) K8s dashboard timeout* portion of the APPENDIX B - Troubleshooting section.

Prepare persistent storage

The HPE Nimble Storage CSP must be installed. The previous step of deploying K8s with the HPE Container Platform allowed for this. If your K8s cluster was provisioned in a different manner, install the HPE CSI Driver for Kubernetes (see the *Overview* in the HPE Storage Container Orchestrator Documentation) and corresponding CSP for your array.

All `kubectl` commands are run from the **director** machine or whatever client that has **kubeconfig** access to the K8s cluster.

Define a secret with storage array details (see the *Create a secret with backend details* in the HPE Storage Container Orchestrator Documentation). Provide the array management IP, username, and base64-encoded password. See APPENDIX A: Code examples – `hpe-nimble-secret.yaml` for a code example.

Create the secret with `kubectl`:

```
kubectl create -f hpe-nimble-secret.yaml
```

Define the storage class(es) for data and logs. See APPENDIX A: Code examples for instances. In particular, ensure you adjust the following for your environment:

Array secret: for example, `hpe-nimble-secret`.

`folder`: this is a folder under the default pool in the HPE Nimble Storage array. **If you do not want to use a folder for organization, remove this key/value pair.**

`performance policy`: customize if you want to use different policies on the HPE Nimble Storage array.

Create the K8s storage classes. These will be specified later in the SQL Server Big Data Clusters configuration files.

```
kubectl create -f sc-sql-bdc-ha-data.yaml

kubectl create -f sc-sql-bdc-ha-logs.yaml
```

SQL Server 2019 Big Data Clusters customized install

Start with a client that has `kubectl` installed and is configured with the **kubeconfig** for your destination Kubernetes cluster (see the Deploy "kubectl" to manage K8s cluster section). A CentOS 7.7 machine was used for the examples in this guide.

Next, install `azdata` (see the *Install with yum* in the Microsoft SQL Server 2019 documentation). This CLI package is used to install Microsoft SQL Server 2019 Big Data Clusters.

Install the `azdata-cli` package and associated prerequisites:

```
rpm --import https://packages.microsoft.com/keys/microsoft.asc

curl -o /etc/yum.repos.d/mssql-server.repo https://packages.microsoft.com/config/rhel/7/mssql-server-2019.repo

yum -y install epel-release

yum --disablerepo="*" --enablerepo="epel" -y install lttng-ust

yum -y install azdata-cli
```

Set a few environment variables to simplify the `azdata` commands:

```
export AZDATA_USERNAME=admin

export AZDATA_PASSWORD="Password#1"

export ACCEPT_EULA=yes
```

Test the `azdata` installation:

```
azdata bdc config list -o table
```

Clone the SQL Server Big Data Clusters installation scripts, where `--target` specifies a directory that will be created:

```
azdata bdc config init --source kubeadm-dev-test --target ha-bdc-nimble
```

Move into the configuration directory created (`ha-bdc-nimble`), and edit `control.json`, specifying the `data` and log `className` created in the Prepare persistent storage section as follows:

```
…
"storage": {
            "data": {
                "className": "sc-sql-bdc-ha-data",
                "accessMode": "ReadWriteOnce",
                "size": "15Gi"
            },
            "logs": {
                "className": "sc-sql-bdc-ha-logs",
                "accessMode": "ReadWriteOnce",
                "size": "10Gi"
            }
        },
…
```

Move into the configuration directory created (`bdc-nimble`), and edit `bdc.json`. In particular, around line 53, we set replicas to `3`, added the `MasterSecondary` endpoint, and set `hadr.enabled` to `true`:

```
                "spec": {
                    "type": "Master",
                    "replicas": 3,
                    "endpoints": [
                        {
                            "name": "Master",
                            "serviceType": "NodePort",
                            "port": 31433
                        },
                        {
                            "name": "MasterSecondary",
                            "serviceType": "NodePort",
                            "port": 32433
                        }
                    ],
                    "settings": {
                        "sql": {
                            "hadr.enabled": "true"
                        }
                    }
                }
```

Create the Big Data Cluster, specifying the configuration directory (`bdc-nimble`):

```
azdata bdc create --config-profile ha-bdc-nimble
```

This step can take anywhere from five minutes to potentially 45 minutes. See APPENDIX B - Troubleshooting for tips.

Once the deployment has completed, login to the cluster with `azdata`:

```
# azdata login -n mssql-cluster
```

As demonstrated in FIGURE 7, query the SQL Server Big Data Cluster endpoints (`# azdata bdc endpoint list -o table`):

```
[root@hcp-controller ~]# azdata bdc endpoint list -o table
Description                                        Endpoint                                                  Name                        Protocol
-------------------------------------------------  --------------------------------------------------------  --------------------------  ----------
Gateway to access HDFS files, Spark                https://10.117.19.18:31443                                gateway                     https
Spark Jobs Management and Monitoring Dashboard     https://10.117.19.18:31443/gateway/default/sparkhistory   spark-history               https
Spark Diagnostics and Monitoring Dashboard         https://10.117.19.18:31443/gateway/default/yarn           yarn-ui                     https
Application Proxy                                  https://10.117.19.17:31778                                app-proxy                   https
Management Proxy                                   https://10.117.19.17:31777                                mgmtproxy                   https
Log Search Dashboard                               https://10.117.19.17:31777/kibana                         logsui                      https
Metrics Dashboard                                  https://10.117.19.17:31777/grafana                        metricsui                   https
Cluster Management Service                         https://10.117.19.18:31080                                controller                  https
SQL Server Master Instance Front-End               10.117.19.17,32433                                        sql-server-master           tds
SQL Server Master Readable Secondary Replicas      10.117.19.17,32436                                        sql-server-master-readonly  tds
HDFS File System Proxy                             https://10.117.19.18:31443/gateway/default/webhdfs/v1     webhdfs                     https
Proxy for running Spark statements, jobs, applications  https://10.117.19.18:31443/gateway/default/livy/v1  livy                        https
```

FIGURE 7. SQL Server Big Data Cluster endpoints query

Access the Microsoft SQL Server Master Instance Front End with a tool like Microsoft Azure Data Studio, as exhibited in FIGURE 8 (see the Microsoft Documentation *Download and install Azure Data Studio* for more information). The server is specified in the format `10.117.19.17,32433`, and uses the same username and password defined in the variables at deployment time (for example, `AZDATA_USERNAME` and `AZDATA_PASSWORD`).

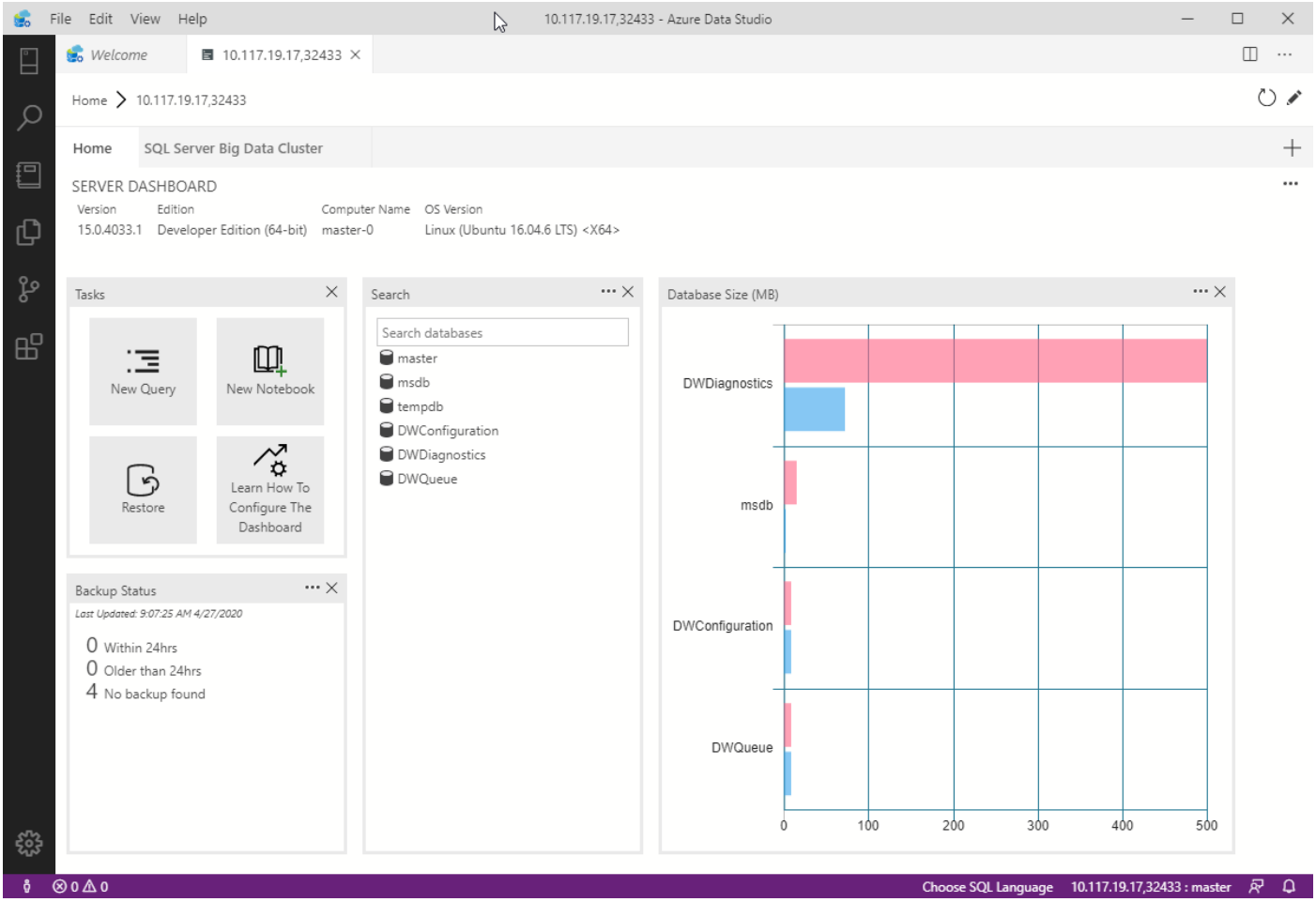

FIGURE 8. "Azure Data Studio" GUI for "SQL Server Big Data Cluster"

From within Azure Data Studio, the SQL Server status can be viewed for **master**, **compute-0**, **data-0**, and **storage-0** (see FIGURE 9).

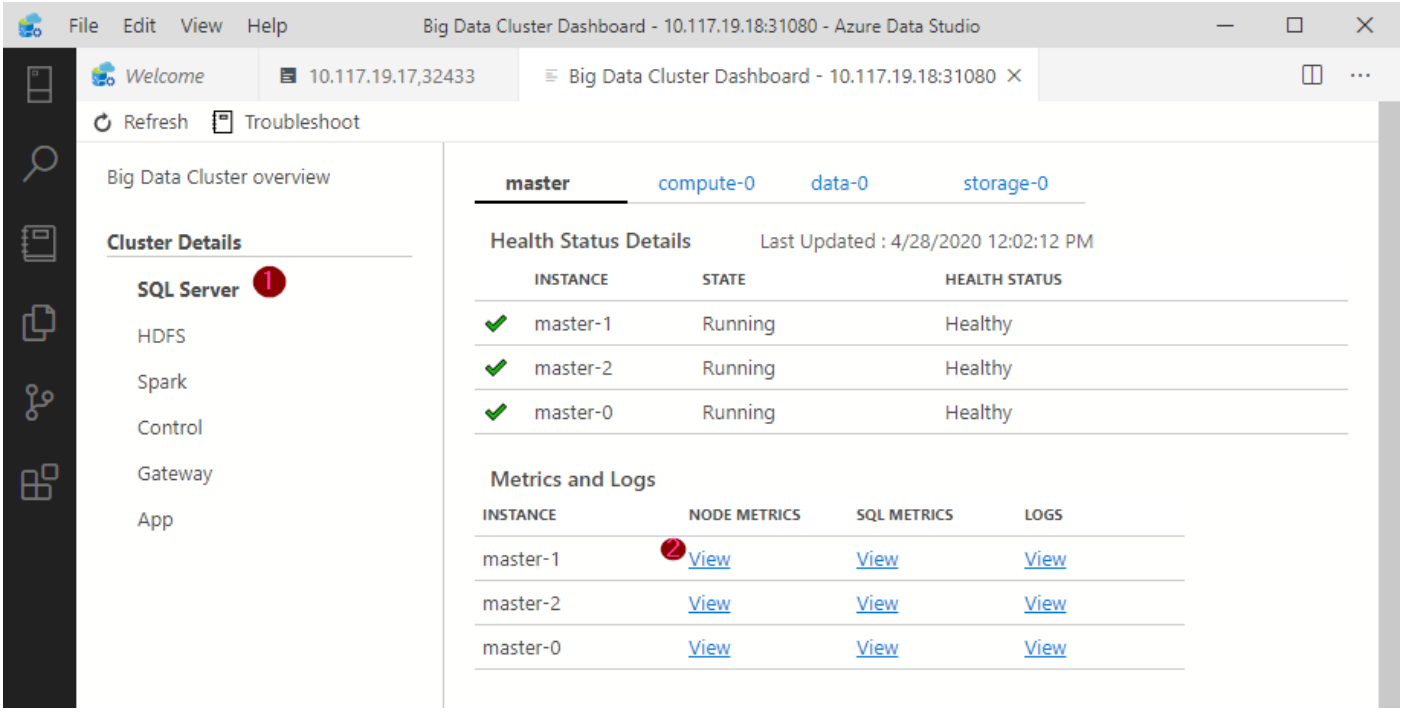

FIGURE 9. SQL Server status in the "Big Data Cluster Dashboard"

A sample Grafana dashboard, as shown in FIGURE 10, is launched from FIGURE 9 by clicking **View** for **INSTANCE master-1** and **NODE METRICS**, denoted as number **2**.

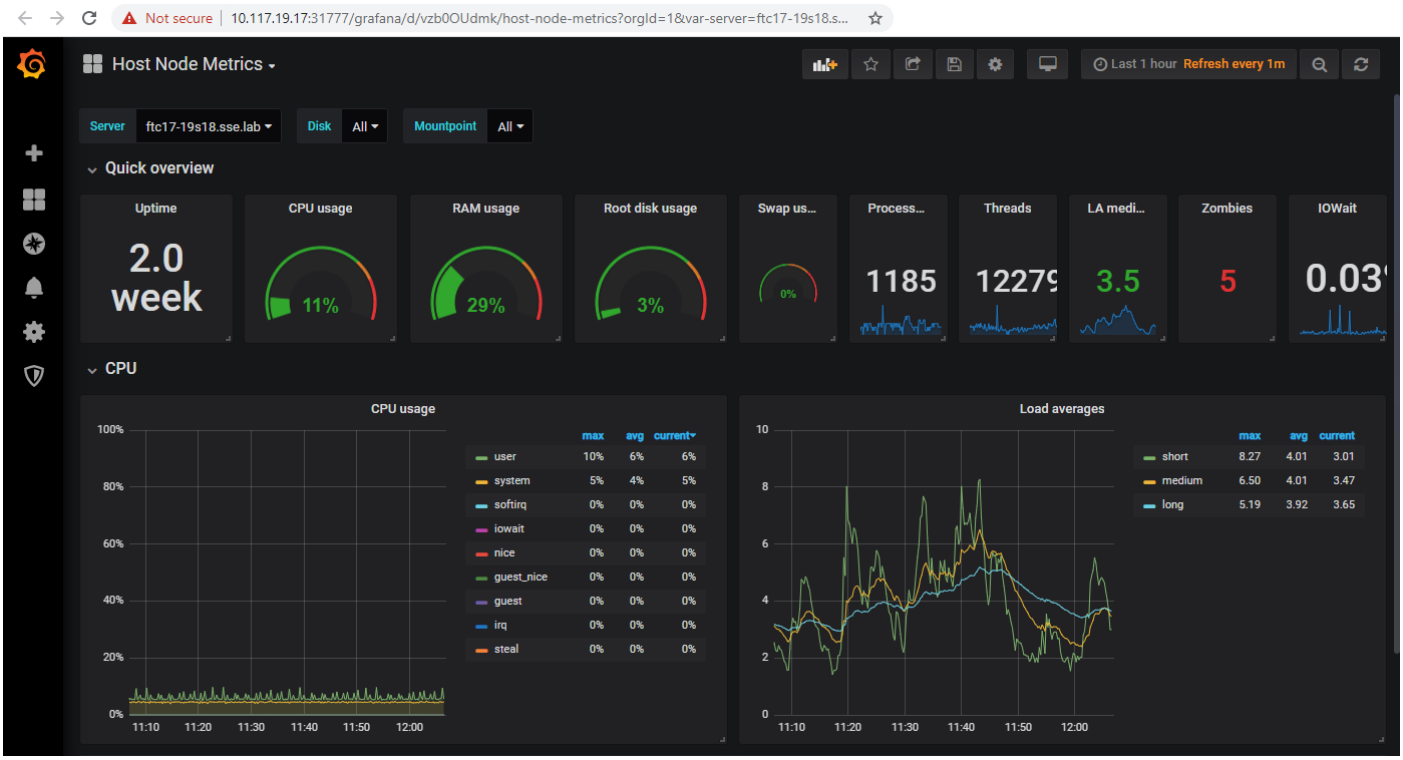

FIGURE 10. Grafana dashboard sample

SUMMARY

While individual product documentation is often well done for a given component, this guide has strived to bring all components together into a complete environment:

- Kubernetes 1.17 as deployed by the HPE Container Platform
- HPE Nimble Storage with support for Kubernetes
- Microsoft SQL Server 2019 Big Data Clusters

Microsoft has assembled a strong package consisting of a containerized and highly scalable SQL Server 2019 platform with the latest in open source tools for analyzing big data.

2020 and beyond will be an exciting period of growth for container orchestration tools and the integration with applications such as Microsoft's SQL Server 2019 Big Data Clusters.

This paper describes solution testing performed in May 2020.

APPENDIX A: CODE EXAMPLES

When using the following examples, please ensure that white space is preserved.

Populate and customize "/etc/profile.d/set_proxy.sh"

```
export http_proxy=<url>
export https_proxy=<url>
export ftp_proxy=<url>
export no_proxy="localhost,127.0.0.1,\
 10.96.0.0/12,10.192.0.0/12,\
 <Individual IP of each HCP and k8s host in the environment comma separated>\
 <Individual FQDN of each HCP and k8s host in the environment comma separated>"
```

Provision "/etc/systemd/system/docker.service.d/docker-proxy.conf"

```
# Create docker service directory if it does not yet exist:
mkdir -p /etc/systemd/system/docker.service.d
# Once set_proxy.sh is properly defined, run this to create the docker-proxy
cat > /etc/systemd/system/docker.service.d/docker-proxy.conf <<EOF
[Service]
Environment="HTTP_PROXY=$http_proxy"
Environment="HTTPS_PROXY=$https_proxy"
Environment="NO_PROXY=$no_proxy"
EOF
```

Provision "/etc/yum.repos.d/kubernetes.repo"

```
#Import k8s yum repository
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-
key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

Populate and customize "hpe-nimble-secret.yaml"

```
apiVersion: v1
kind: Secret
metadata:
  name: hpe-nimble-secret
  namespace: kube-system
stringData:
  serviceName: nimble-csp-svc
  servicePort: "8080"
  backend: <192.168.1.1>
  username:  admin
data:
  # echo -n "admin" | base64
  password: YWRtaW4=
```

Populate and customize “sc-sql-bdc-ha-data.yaml”

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name:  sc-sql-bdc-ha-data
provisioner: csi.hpe.com
parameters:
  description:  "BDC HA volumes"
  accessProtocol: "iscsi"
  csi.storage.k8s.io/fstype: xfs
  csi.storage.k8s.io/provisioner-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/provisioner-secret-namespace: kube-system
  csi.storage.k8s.io/controller-publish-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/controller-publish-secret-namespace: kube-system
  csi.storage.k8s.io/node-stage-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/node-stage-secret-namespace: kube-system
  csi.storage.k8s.io/node-publish-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/node-publish-secret-namespace: kube-system
  csi.storage.k8s.io/controller-expand-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/controller-expand-secret-namespace: kube-system
  # Extras...
  dedupeEnabled: "true"
  performancePolicy: "SQL Server"
  folder: "BDC-SQL-HA-Vols"
  thick: "false"
  destroyOnDelete: "false"
reclaimPolicy: Retain
volumeBindingMode: Immediate
allowVolumeExpansion: True
```

Populate and customize “sc-sql-bdc-ha-logs.yaml”

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-sql-bdc-ha-logs
provisioner: csi.hpe.com
parameters:
  description: "MS SQL BDC HA Logs"
  accessProtocol: "iscsi"
  csi.storage.k8s.io/fstype: xfs
  csi.storage.k8s.io/provisioner-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/provisioner-secret-namespace: kube-system
  csi.storage.k8s.io/controller-publish-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/controller-publish-secret-namespace: kube-system
  csi.storage.k8s.io/node-stage-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/node-stage-secret-namespace: kube-system
  csi.storage.k8s.io/node-publish-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/node-publish-secret-namespace: kube-system
  csi.storage.k8s.io/controller-expand-secret-name: hpe-nimble-secret
  csi.storage.k8s.io/controller-expand-secret-namespace: kube-system
  # Extras...
  dedupeEnabled: "true"
  performancePolicy: "SQL Server Logs"
  folder: "BDC-SQL-HA-Vols"
  thick: "false"
  destroyOnDelete: "false"
reclaimPolicy: Retain
volumeBindingMode: Immediate
# Extras...|
allowVolumeExpansion: True
```

APPENDIX B: TROUBLESHOOTING

Kubernetes troubleshooting

For Kubernetes clusters deployed by the HPE Container Platform, see the following documentation for troubleshooting examples:

- Kubernetes Installation
- Kubernetes Nodes
- Kubernetes Cluster Creation
- Kubernetes Web Interface
- General Kubernetes Application/ Deployment Issues

Microsoft SQL Server Big Data Clusters cleanup

When installing Microsoft SQL Server Big Data Clusters, if the script does not error or time out, Hewlett Packard Enterprise recommends giving it the full 45 minutes before intervening. The default namespace is `mssql-cluster`. If it errors and cleanup is necessary, the following commands may be helpful in cleaning up the K8s namespace:

```
kubectl -n <yournamespace> delete daemonset,statefulset,replicaset,svc,pod,secret --all
kubectl delete namespace mssql-cluster
```

Increase (or disable) K8s dashboard timeout

Edit the K8s dashboard deployment as follows:

```
[root@hcp-controller ~]# kubectl edit deployment -n kubernetes-dashboard kubernetes-dashboard
```

Add the `token-ttl=0` attribute, as in the following example:

```
…
    spec:
      containers:
      - args:
        - --auto-generate-certificates
        - --namespace=kubernetes-dashboard
        - --token-ttl=0
        image: kubernetesui/dashboard:v2.0.0-rc1
        imagePullPolicy: Always
        livenessProbe:
          failureThreshold: 3
…
```

RESOURCES AND ADDITIONAL LINKS

HPE Storage solutions for Microsoft SQL Server 2019 and Big Data Clusters (lightboard video)
https://www.youtube.com/watch?v=9-W-jeArylw

Introduction to Big Data Cluster on SQL Server 2019 | Virtualization, Kubernetes, and Containers
https://www.youtube.com/watch?v=q7mxWcYqBMM

Introduction to Big Data Cluster on SQL Server 2019 | Architecture
https://www.youtube.com/watch?v=GvOo-VV7-p4

Deploy SQL Server Big Data Cluster with high availability
https://docs.microsoft.com/en-us/sql/big-data-cluster/deployment-high-availability?view=sql-server-ver15

HPE Sizing Tool for the Elastic Platform for Analytics (EPA)
https://solutionsizers.ext.hpe.com/EPASizer/Build/HPEContainerPlatform

Kubectl Kubernetes CheatSheet
https://github.com/dennyzhang/cheatsheet-kubernetes-A4/blob/master/cheatsheet-kubernetes-A4.pdf

LEARN MORE AT

hpe.com/storage/microsoft

a00099567ENW, May 2020

Hewlett Packard Enterprise