NetBackup™ Deployment Guide for Azure Kubernetes Services (AKS) Cluster

Release 10.0

VERITAS™

NetBackup Deployment Guide for Azure Kubernetes Services (AKS) Cluster

Last updated: 2022-02-28

Legal Notice

Veritas Technologies LLC
2625 Augustine Drive
Santa Clara, CA 95054

http://www.veritas.com

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:

https://www.veritas.com/support

You can manage your Veritas account information at the following URL:

https://my.veritas.com

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)	CustomerCare@veritas.com
Japan	CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The latest documentation is available on the Veritas website:

https://sort.veritas.com/documents

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:

NB.docs@veritas.com

You can also see documentation information or ask a question on the Veritas community site:

http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:

https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Introduction to NetBackup on AKS

This chapter includes the following topics:

About NetBackup deployment on Azure Kubernetes Services (AKS) cluster

NetBackup provides the product deployment solution on Azure Kubernetes Services cluster (AKS), in the Azure Cloud. The solution facilitates an orchestrated deployment of the NetBackup components on AKS.

You can deploy NetBackup on AKS for scaling the capacity of the NetBackup host to serve a large number of requests concurrently running on the NetBackup primary server at its peak performance capacity.

This guide provides you two distinct methods of deployment. The first and the recommended one is by using the environment operators. In this method, you can deploy the entire NetBackup environment with ease. You can deploy, one primary, and optionally, one media with one or more replicas, and one MSDP Scaleout with four to 16 replicas. The guide describes a very comprehensive method to deploy, configure, and remove the NetBackup components using the environment operators.

You can also go for a discrete deployment of the NetBackup components without using the environment operator. This method is not the recommended method of deployment.

Supported platforms

Currently we support Azure Kubernetes Service.

About the guide

This guide contains the following sections:

- Introduction to NetBackup and MSDP Scaleout —preparatory steps to ensure that your AKS cluster and hardware environment meet the deployment requirements.
- Deploying with environment operators—deploy the entire NetBackup environment primary, media, and MSDP Scaleout servers together in a comprehensive way. This is the most recommended method of deployment.
- Assessing cluster configuration before deployment—check the deployment environment to verify that the environment meets the requirements, before starting the primary server and media server deployments.
- Deploying NetBackup—deploying NetBackup without the environment operator.
- Deploying MSDP Scaleout—deploying MSDP Scaleout without the environment operator.
- Monitoring NetBackup—monitor application health, view logs, expand storage volume and so on.
- Monitoring MSDP Scaleout—monitor status, alerts, events, Azure container insights, and so on.
- Configuring the Load Balancer service—configure the load balancer to access NetBackup from private IPs.
- Performing catalog backup and recovery—how to back up the catalog and recover.
- Configuring MSDP Scaleout—adding MSDP Scaleout engines and data volumes, disaster recovery, scaling and so on.
- Maintaining MSDP Scaleout—running maintenance, logging, reinstalling the operator, and so on.
- Uninstalling MSDP Scaleout from AKS—uninstall and cleanup the cluster and the operator.

- Scenarios for troubleshooting.

The intended audience for this document includes backup, cloud, system administrators, and architects.

Note: NetBackup deployment for AKS offers only English language support and it does not support OpsCenter.

Required terminology

The table describes the important terms for NetBackup deployment on AKS cluster. For more information visit the link to Kubernetes documentation.

Table 1-1 Important terms

Term	Description
Pod	A Pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. For more information on Pods, see Kubernetes Documentation.
StatefulSet	StatefulSet is the workload API object used to manage stateful applications and it represents a set of Pods with unique, persistent identities, and stable hostnames. For more information on StatefulSets, see Kubernetes Documentation.
Job	Kubernetes jobs ensure that one or more pods execute their commands and exit successfully. For more information on Jobs, see Kubernetes Documentation.
ConfigMap	A ConfigMap is an API object used to store non-confidential data in key-value pairs. For more information on ConfigMaps, see Kubernetes Documentation.
Service	A Service enables network access to a set of Pods in Kubernetes. For more information on Service, see Kubernetes Documentation.
Persistent Volume Claim	A PersistentVolumeClaim (PVC) is a request for storage by a user. For more information on Persistent Volumes, see Kubernetes Documentation.
Persistent Volume	A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using storage classes. For more information on Persistent Volumes, see Kubernetes Documentation.

Table 1-1 Important terms *(continued)*

Term	Description
Custom Resource	A Custom Resource (CR) is an extension of the Kubernetes API that is not necessarily available in a default Kubernetes installation. For more information on Custom Resources, see Kubernetes Documentation.
Custom Resource Definition	The CustomResourceDefinition (CRD) API resource lets you define custom resources. For more information on CustomResourceDefinitions, see Kubernetes Documentation.
Secret	A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a Pod specification or in a container image. For more information on Secrets, see Kubernetes Documentation.
ServiceAccount	A service account provides an identity for processes that run in a Pod. For more information on configuring the service accounts for Pods, see Kubernetes Documentation.
ClusterRole	An RBAC Role or ClusterRole contains rules that represent a set of permissions. Permissions are purely additive (there are no "deny" rules). For more information on ClusterRole, see Kubernetes Documentation.
ClusterRoleBinding	A role binding grants the cluster-wide permissions defined in a role to a user or set of users. For more information on ClusterRoleBinding, see Kubernetes Documentation.
Namespace	Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called namespaces. For more information on Namespaces, see Kubernetes Documentation.

User roles and permissions

Note the following for user authentication:

- An Administrator must define the custom user credentials by creating a secret; and then provide the secret name at the time of primary server deployment.
- A custom user is assigned the role of a NetBackup Security Administrator and can access the NetBackup Web UI after deployment.
- A custom user will be persisted during the pods restart or upgrade.

- For the custom user, you can change only the password after the deployment. The changed password will be persisted. If the username is changed after the deployment, an error message will be logged in the Operator pod.
- You can delete the secret after the primary server deployment. In that case, if you want to deploy or scale the media servers, you must create a new secret with the same username which was used in the primary server CR. The password can be the same or different. If you change the password, it is also changed in the primary server pod, and gets persisted.
- Do not create a local user in the pods (using the `kubectl exec` or `useradd` commands) as this user may or may not be persisted.
- The Azure Active Directory user is supported through Single Sign-on (SSO). For the detailed user integration information, refer to the *NetBackup Administrator's Guide Volume I*.
- An **nbitanalyticsadmin** user is available in primary server container. This user is used as **Master Server User ID** while creating data collector policy for data collection on NetBackup IT Analytics portal.
- Service account that is used for this deployment is **netbackup-account** and it is defined in the `operator_deployment.yaml`.
- NetBackup runs most of the primary server services and daemons as non-root user **(nbsvcusr)** and only **root** and **nbsvcusr** are supported as a service account user.
- ClusterRole named **netbackup-role** is set in the NetBackup Operator to define the cluster wide permissions to the resources. This is defined in the `operator_deployment.yaml.`
- Appropriate roles and AKS specific permissions are set to the cluster at the time of cluster creation.
- After successful deployment of the primary and media servers, the operator creates a custom Kubernetes role with name `<resourceNamePrefix>-admin` whereas `resourceNamePrefix` is given in primary server or media server CR specification.
  The following permissions are provided in the respective namespaces:

Resource name	API group	Allowed operations
ConfigMaps	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Nodes	default	■ Get ■ List

This role can be assigned to the NetBackup Administrator to view the pods that were created, and to execute into them. For more information on the access control, see Kubernetes Access Control Documentation.

Note: One role would be created, only if primary and media servers are in same namespace with the same resource name prefix.

- Your AKS cluster must have the RBAC enabled. To view the permissions set for the AKS cluster, use one of the following methods and verify if `enbleRBAC` is set to **true**:
  - Run the following command:
    ```
    az resource show -g <resource group name> -n <cluster name>
    --resource-type
    Microsoft.ContainerService/ManagedClusters --query
    properties.enableRBAC
    ```
  - Run the `az aks list` command.
  - You can check the cluster's resource details at `resources.azure.com` and verify if `enableRBAC` is set to **true**.

Role-based authentication (RBAC)

NetBackup Operator deployment uses a `serviceAccount` and it must have the following permissions:

Table 1-2

Resource Name	API Group	Allowed Operations
ConfigMaps	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Nodes	default	■ Get ■ List
PersistentVolumeClaims	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Pods	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Pods/exec	default	■ Create ■ Get
Secret	default	■ Get ■ List ■ Watch
Services	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch

Table 1-2 *(continued)*

Resource Name	API Group	Allowed Operations
StatefulSet	app	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Jobs	batch	■ Create ■ Delete ■ Get ■ List
Primary servers	netbackup.veritas.com	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
PrimaryServers/status	netbackup.veritas.com	■ Get ■ Patch ■ Update
Media servers	netbackup.veritas.com	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
MediaServers/status	netbackup.veritas.com	■ Get ■ Patch ■ Update
Secrets	netbackup.veritas.com	Watch

Table 1-2 *(continued)*

Resource Name	API Group	Allowed Operations
Secrets/status	netbackup.veritas.com	■ Get ■ Patch ■ Update
Roles	rbac.authorization.k8s.io	■ Create ■ Get ■ List ■ Watch
Storageclasses	storage.k8s.io	■ Get ■ List

About MSDP Scaleout

MSDP Scaleout is based on MSDP. It empowers MSDP with high resilience and scalability capabilities to simplify management and reduce total cost of ownership.

It runs on multiple nodes to represent a single storage pool for NetBackup and other Veritas products to use. You can seamlessly scale out and scale up a MSDP Scaleout on demand. MSDP Scaleout automatically does failure detection and repair in the background.

It is deployed separately in NetBackup environment. The deployment process is with minimal user intervention. The core MSDP services run on each node to expose the storage optimized services, and manage a part of the cluster level data and metadata. Each MSDP Scaleout node is called MSDP engine.

See “Deploying MSDP Scaleout ” on page 61.

About MSDP Scaleout components

Following are the MSDP Scaleout components:

- MDS (MetaData service)
  MDS is an independent and stackable service that provides a single system view of MSDP Scaleout. It's an etcd cluster running inside the MDS pods. These pods run on different AKS nodes. The pod name has a format of **<cr-name>-uss-mds-<1,2...>**.
  The number of pods that get created depends on the number of MSDP Scaleout engines in AKS cluster. These pods are controlled by the MSDP operator.

  - 1 or 2 MSDP Scaleout engines: 1 pod
  - 3 or 4 MSDP Scaleout engines: 3 pods
  - 5 or more MSDP Scaleout engines: 5 pods

- MSDP Scaleout Controller
  Controller is a singleton service and the entry point of MSDP Scaleout that monitors and repairs MSDP Engines. It controls and manages the application-level business of the MSDP Scaleout. The Deployment object name has a format of **<cr-name>-uss-controller**. It is controlled by the MSDP operator.
- MSDP Scaleout Engine
  MSDP Engines provide the ability to write deduplicated data to the storage. The name of a MSDP engine pod is the corresponding FQDN of the static IP that is specified in the CR. Each MSDP engine pod has MSDP services such as spad, spoold, and ocsd running. They are controlled by the MSDP operator.

Limitations in MSDP Scaleout

MSDP Scaleout has the following limitations:

- Does not support cloud immutable (WORM) storage.
- It is not fully compliant with Federal Information Processing Standards (FIPS).
  The internal services MSDP operator, MSDP Controller, and MDS of a MSDP Scaleout are not compliant with FIPS.
  MSDP is FIPS compliant. For more information, see the *NetBackup Deduplication Guide*.
- Does not support SELinux.
- Does not support Instant Access.
- Does not support Universal Shares.
- Supports only NBCA. Does not support ECA.
- Does not support availability zones for AKS cluster.
  For more information about the limitation see Create an Azure Kubernetes Service (AKS) cluster that uses availability zones topic of the Azure documentation.
- Limited AKS node failure tolerance.
  Backup and restore can fail if AKS node fails. If MSDP operator detects the MSDP Scaleout pod failure, it attempts to restart it and perform a repair operation automatically. The repair operation can be delayed if Azure infrastructure or Kubernetes do not allow the pod to be restarted.

An Azure volume cannot be attached to two different nodes at the same time. When the node which Azure volume is attached to fails, MSDP operator cannot run the same pod with the same Azure volume on another node until the failed node is repaired or deleted by AKS.
AKS node auto-repair may take more than 20 minutes to finish. In some cases, it may be necessary to bring the node back up manually.
See Azure Kubernetes Service (AKS) node auto-repair

- IPv6 is not supported.

Chapter 2

Deployment with environment operators

This chapter includes the following topics:

About deployment with the environment operator

This section describes the deployment of the Veritas NetBackup and MSDP Scaleout on Azure Kubernetes Service in Azure cloud. You can start by deploying the two environment operators that together manage the NetBackup environment, the primary server, the media servers, and the MSDP Scaleout storage servers.

Prerequisites

Ensure that the following prerequisites are met before proceeding with the deployment.

- A Kubernetes cluster in Azure with multiple nodes. Using separate node pools is recommended for the NetBackup and MSDP Scaleout deployments.

- Define one or more storage classes in the Kubernetes cluster.
- Access to a container registry that the Kubernetes cluster can access, like an Azure Container Registry.
- Enable AKS Uptime SLA. AKS Uptime SLA is recommended for a better resiliency. For information about AKS Uptime SLA and to enable it, see Azure Kubernetes Service (AKS) Uptime SLA.
- Install Cert-Manager. You can use the following command to install the Cert-Manager:

  ```
  $ kubectl apply -f
  https://github.com/jetstack/cert-manager/releases/download/v1.6.0/cert-manager.yaml
  ```

  For details, see https://cert-manager.io/docs/installation/
- A workstation or VM running Linux with the following:
  - Configure `kubectl` to access the cluster.
  - Install Azure CLI to access Azure resources.
  - Configure docker to be able to push images to the container registry.
  - Free space of approximately 8.5GB on the location where you copy and extract the product installation TAR package file. If using docker locally, there should be approximately 8GB available on the `/var/lib/docker` location so that the images can be loaded to the docker cache, before being pushed to the container registry.

Contents of the TAR file

Download the TAR file from the Veritas download center.

The TAR file contains the following:

Table 2-1 TAR contents

Item	Description
OCI images in the `/images` directory	These docker image files that are loaded and then copied to the container registry to run in Kubernetes. They include NetBackup and MSDP Scaleout application images and the operator images.
MSDP kubectl plug-in at `/bin/kubectl-msdp`	Used to deploy and manage the MSDP Scaleout operator tasks.
Configuration(.yaml) files at `/operator` directory	You can edit these to suit your configuration requirements before installation.

Table 2-1 TAR contents *(continued)*

Item	Description
Sample product (.yaml) files at `/samples` directory	You can use these as templates to define your NetBackup environment.
`README.md`	Readme file.
`deploy.sh`	Deployment script.

Known limitations

Here are some known limitations.

- Changes to the CorePattern which specifies the path used for storing core dump files in case of a crash are not supported. CorePattern can only be set during initial deployment.
- Changes to MSDP Scaleout credential autoDelete, which allows automatic deletion of credential after use, is not supported. The autoDelete value can only be set during initial deployment.

Deploying using the deploy.sh file

You can use the `deploy.sh` script to deploy the complete NetBackup environment, along with configurations and MSDP Scaleout storage. You can perform the deployment from a Linux workstation or VM where you have downloaded the TAR file.

During the deployment process, the `deploy.sh` script prompts you for several required inputs. It is recommended that you gather the following information before you start with the deployment.

Table 2-2 Parameters required during deployment

Parameter	Description
Container registry name	The cluster must have access to this registry and the docker must be configured and authorized on the VM to push the images to the registry. For example , if FQDN of Azure Container Registry is `example.azurecr.io` then login using short name, like: `az acr login -n example`

Table 2-2 Parameters required during deployment *(continued)*

Parameter	Description
Namespaces	Namespaces where the operators and the applications are deployed. The Primary, Media, and MSDP Scaleout application namespace must be different from the one used by the operators.
kubectl-msdp plug-in location	The plug-in's directory must be set in the path environment variable of the shell.
License keys	NetBackup license keys.
Storage classes	Storage classes for data, catalog, and the log volumes.
Host names and IP addresses	A FQDN/IP pair for each of the following: - 1 Primary - 1 or more media server replicas - 4-16 MSDP Scaleout server replicas **Note:** You must use separate host names and IP addresses for the Primary, Media, and MSDP Scaleout servers, respectively.
Node selector key/values	This defines the node pools to be used for the deployment of primary, media, and MSDP Scaleout components. Recommendation is to use separate node pools for them (with appropriate node selectors) and to use appropriate node sizes for them according to your requirements.
Details of the secrets for MSDP Scaleout storage, Primary server, and KMS DB	The deployment script creates the secrets, taking inputs from you.
KMS key group name	Key secret.

To deploy using the deploy.sh script.

1. Execute the following commands, one by one:

   ```
   $ REVISION=<Build version number>

   $ tar -xvf VRTSk8s-netbackup-10.0-${REVISION}.tar

   $ cd VRTSk8s-netbackup-10.0-${REVISION}

   $ ./deploy.sh
   ```

2. When prompted, specify the following:
   - Option *1*: To install the NetBackup and MSDP Scaleout operators. The script validates the prerequisites and performs the installation of the operators.
   - Option *2*: To install NetBackup primary, media and MSDP Scaleout applications (collectively called as the NetBackup Environment). To perform this step, you must have already installed the operators.
   - Option *3*: To quit and close the prompt.

Deploying the operators manually

To perform these steps, log on to the Linux workstation or VM where you have extracted the TAR file.

To deploy the operators

1. Install the MSDP kubectl plug-in at some location which is set in the path environment variable of your shell. For example, copy the file `kubectl-msdp` to`/usr/local/bin/`.

2. Run the following commands to load each of the product images to the local docker instance.

   ```
   $ docker load -i netbackup-main-10.0.tar.gz

   $ docker load -i netbackup-operator-10.0.tar.gz

   $ docker load -i pdcluster-16.0.tar.gz

   $ docker load -i pdde-16.0.tar.gz

   $ docker load -i pdk8soptr-16.0.tar.gz
   ```

   Run the command `docker image ls` to confirm that the product images are loaded properly to the docker cache.

3. Run the following commands to re-tag the images to associate them with your container registry.

```
$ REGISTRY=<example.azurecr.io> (Replace with your own container
registry name)

$ docker tag netbackup/main:10.0 ${REGISTRY}/netbackup/main:10.0

$ docker tag netbackup/operator:10.0
${REGISTRY}/netbackup/operator:10.0

$ docker tag uss-engine:16.0 ${REGISTRY}/uss-engine:16.0

$ docker tag uss-controller:16.0 ${REGISTRY}/uss-controller:16.0

$ docker tag uss-mds:16.0 ${REGISTRY}/uss-mds:16.0

$ docker tag msdp-operator:16.0 ${REGISTRY}/msdp-operator:16.0
```

4. Run the following commands to push the images to the container registry.

```
$ docker push ${REGISTRY}/netbackup/main:10.0

$ docker push ${REGISTRY}/netbackup/operator:10.0

$ docker push ${REGISTRY}/uss-engine:16.0

$ docker push ${REGISTRY}/uss-controller:16.0

$ docker push ${REGISTRY}/uss-mds:16.0

$ docker push ${REGISTRY}/msdp-operator:16.0
```

5. Create a namespace for deploying the NetBackup and MSDP Scaleout operators. These instructions use the default `netbackup-operator-system` namespace but a custom namespace is also supported, run:

```
$ kubectl create namespace netbackup-operator-system
```

6. Install the MSDP Scaleout operator in the created namespace, using this command. To run this command you must define a full image name in step 3, define a storage class for storing logs from the MSDP operator, and define node selector labels (optional) for scheduling the MSDP operator pod on specific nodes. See “Prerequisites” on page 19.

```
$ kubectl msdp init --image ${REGISTRY}/msdp-operator:16.0
--storageclass x --namespace netbackup-operator-system -l
key1=value1
```

7 To verify that the MSDP Scaleout operator is running, run:

```
$ kubectl get all --namespace netbackup-operator-system
```

Here, we are using the namespace created in step 5.

The **msdp-operator** pod should show status as *Running*.

8 In this step, configure the namespace, image name, and node selector to use for the NetBackup operator image by editing the provided configuration yaml files.

- (Optional) Perform this step only when using a custom namespace. Edit the file `operator/kustomization.yaml` and change `namespace` to your custom namespace. For example: *namespace: my-custom-namespace*
- Edit the file `operator/kustomization.yaml` and change `newName` and `newTag`. For example:

```
images:
     - name: netbackupoperator
       newName: example.com/netbackup/operator
       newTag: '10.0'
```

- Edit the file `operator/patches/operator_patch.yaml` to add or remove node selectors that control what nodes Kubernetes may schedule the operator to run on. For example:

```
nodeSelector:
        nbu_node: 'true'
```

  If you have no restrictions on pod scheduling, then remove the `nodeSelector` section entirely.

9 To install the NetBackup operator, run the following command from the installer's root directory:

```
$ kubectl apply -k operator
```

10 To verify if the NetBackup operator is running, run:

```
$ kubectl get all --namespace netbackup-operator-system
```

Verify that pod/netbackup-operator STATUS is showing as *Running*.

Deploying NetBackup and MSDP Scaleout manually

After the operators are deployed, you can deploy the NetBackup and MSDP Scaleout environment.

To deploy NetBackup primary, media, and MSDP Scaleout components:

1 Create a Kubernetes namespace where your new NetBackup environment will run. Run the command:

```
kubectl create namespace nb-example
```

Where, *nb-example* is the name of the namespace. The Primary, Media, and MSDP Scaleout application namespace must be different from the one used by the operators. It is recommended to use two namespaces. One for the operators, and a second one for the applications.

2 Create a secret to hold the primary server credentials. Those credentials are configured in the NetBackup primary server, and other resources in the NetBackup environment use them to communicate with and configure the primary server. The secret must include fields for `username` and `password`. If you are creating the secret by YAML, the type should be opaque or basic-auth. For example:

```
apiVersion: v1
       kind: Secret
       metadata:
         name: primary-credentials
         namespace: nb-example
       type: kubernetes.io/basic-auth
       stringData:
         username: nbuser
         password: p@ssw0rd
```

You can also use this command to create a secret.

```
$ kubectl create secret generic primary-credentials --namespace
nb-example --from-literal=username='nbuser'
--from-literal=password='p@ssw0rd'
```

3 Create a KMS DB secret to hold Host Master Key ID (`HMKID`), Host Master Key passphrase (`HMKpassphrase`), Key Protection Key ID (`KPKID`), and Key Protection Key passphrase (`KPKpassphrase`) for NetBackup Key Management Service. If creating the secret by YAML, the type should be _opaque_. For example:

```
apiVersion: v1
      kind: Secret
      metadata:
        name: example-key-secret
        namespace: nb-example
      type: Opaque
      stringData:
        HMKID: HMKID
        HMKpassphrase: HMKpassphrase
        KPKID: KPKID
        KPKpassphrase: KPKpassphrase
```

You can also create a secret using kubectl from the command line:

```
$ kubectl create secret generic example-key-secret --namespace
nb-namespace --from-literal=HMKID="HMKID"
--from-literal=HMKpassphrase="HMKpassphrase"
--from-literal=KPKID="KPKID"
--from-literal=KPKpassphrase="KPKpassphrase"
```

For more details on NetBackup deduplication engine credential rules, see: https://www.veritas.com/content/support/en_US/article.100048511

4 Create a secret to hold the MSDP Scaleout credentials for the storage server. The secret must include fields for `username` and `password` and must be located in the same namespace as the Environment resource. If creating the secret by YAML, the type should be _opaque_ or _basic-auth_. For example:

```
apiVersion: v1
       kind: Secret
       metadata:
         name: msdp-secret1
         namespace: nb-example
       type: kubernetes.io/basic-auth
       stringData:
         username: nbuser
         password: p@ssw0rd
```

You can also create a secret using kubectl from the command line:

```
$ kubectl create secret generic msdp-secret1 --namespace
nb-example --from-literal=username='nbuser'
--from-literal=password='p@ssw0rd'
```

Note: You can use the same secret for the primary server credentials (from step 2) and the MSDP Scaleout credentials, so the following step is optional. However, to use the primary server secret in an MSDP Scaleout, you must set the `credential.autoDelete` property to *false*. The sample file includes an example of setting the property. The default value is *true*, in which case the secret may be deleted before all parts of the environment have finished using it.

5 (Optional) Create a secret to hold the KMS key details. Specify KMS Key only if the KMS Key Group does not already exist and you need to create.

Note: When reusing storage from previous deployment, the KMS Key Group and KMS Key may already exist. In this case, provide KMS Key Group only.

If creating the secret by YAML, the type should be _opaque_. For example:

```
apiVersion: v1
       kind: Secret
       metadata:
         name: example-key-secret
         namespace: nb-example
       type: Opaque
       stringData:
         username: nbuser
         passphrase: 'test passphrase'
```

You can also create a secret using kubectl from the command line:

```
$ kubectl create secret generic example-key-secret --namespace
nb-example --from-literal=username="nbuser"
--from-literal=passphrase="test passphrase"
```

You may need this key for future data recovery. After you have successfully deployed and saved the key details. It is recommended that you delete this secret and the corresponding key info secret.

6 Configure the `samples/environment.yaml` file according to your requirements. This file defines a primary server, media servers, and scale out MSDP Scaleout storage servers. See "Configuring the `environment.yaml` file" on page 30. for details.

7 Apply the environment yaml file, using the same application namespace created in step 1.

```
$ kubectl apply --namespace nb-example --filename environment.yaml
```

Use this command to verify the new environment resource in your cluster:

```
$ kubectl get --namespace nb-example environments
```

The output should look like:

```
NAME                 AGE
environment-sample   2m
```

After a few minutes, NetBackup finishes starting up on the primary server, and then the media servers and MSDP Scaleout storage servers you configured in the environment resource start appearing. Run:

```
$ kubectl get --namespace nb-example
all,environments,primaryservers,mediaservers,msdpscaleouts
```

The output should show:

- All pod status as Ready and Running

  ```
  NAME                         READY   STATUS
  pod/dedupe1-uss-controller- 1/1     Running
  pod/dedupe1-uss-mds-1        1/1     Running
  ```

- For `msdpscaleout` SIZE = READY, for example: 4=4.

  ```
  NAME                                     SIZE   READY
  msdpscaleout.msdp.veritas.com/dedupe1      4      4
  ```

- `environment.netbackup` should show STATUS as Success

  ```
  NAME                                                   STATUS
  environment.netbackup.veritas.com/environment-sample   Success
  ```

8 To start using your newly deployed environment sign-in to NetBackup web UI. Open a web browser and navigate to `https://<primaryserver>/webui/login` URL.

The primary server is the host name or IP address of the NetBackup primary server.

You can retrieve the primary server's hostname by using the command:

```
$ kubectl describe primaryserver.netbackup.veritas.com/<primary
server CR name>--namespace <namespace_name>
```

Configuring the `environment.yaml` file

The `environment.yaml` file lets you configure the primary server, media servers, and scale out MSDP Scaleout storage servers. The file contains four sections, the first section contains parameters that are applicable to all the servers, rest of the sections are one each for the primary, media, MSDP Scaleout servers.

The following configurations apply to all the components:

Table 2-3 Common environment parameters

Parameter	Description
name:*environment-sample*	Specify the name of the environment in your cluster.
namespace: *example-ns*	Specify the namespace where all the NetBackup resources are managed. If not specified here, then it will be the current namespace when you run the command `kubectl apply -f` on this file.
containerRegistry: *example.azurecr.io*	Specify a container registry that the cluster has access. NetBackup images are pushed to this registry.
tag: '*10.0*'	This tag is used for all images in the environment. Specifying a \`tag\` value on a sub-resource affects the images for that sub-resource only. For example, if you apply an EEB that affects only primary servers, you might set the \`primary.tag\` to the custom tag of that EEB. The primary server runs with that image, but the media servers and MSDP scaleouts continue to run images tagged \`10.0\`. Beware that the values that look like numbers are treated as numbers in YAML even though this field needs to be a string; quote this to avoid misinterpretation.
licenseKeys:	List the license keys that are shared among all the sub-resources. Licenses specified in a sub-resource are appended to this list and applied only to the sub-resource.
paused: *false*	Specify whether the NetBackup operator attempts to reconcile the differences between this YAML specification and the current Kubernetes cluster state. Only set it to true during maintenance.
configCheckMode: *default*	This controls whether certain configuration restrictions are checked or enforced during setup. Other allowed values are skip and dryrun.
corePattern: */corefiles/core.%e.%p.%t*	Specify the path to use for storing core files in case of a crash.
loadBalancerAnnotations: service. beta.kubernetes.io/ azure-load- balancer-internal-subnet: *example-subnet*	Specify the annotations to be added for the network load balancer

The following configurations apply to the primary server. The values specified in the following table can override the values specified in the table above.

Table 2-4 Environment parameters for the primary server

Paragraph	Description
paused: *false*	Specifies whether the NetBackup operator attempts to reconcile the differences between this YAML specification and the current Kubernetes cluster state. Set it to *true* only during maintenance. This applies only to the environment object. To pause reconciliation of the managed primary server, for example, you must set `spec.primary.paused`. Setting `spec.paused:true` ceases updates to the managed resources, including updates to their \`paused\` status. Entries in the media servers and MSDP scaleouts lists also support the \`paused\` field. The default value is *false*.
primary	Specifies attributes specific to the primary server resources. Every environment has exactly one primary server, so this section cannot be left blank.
resourceNamePrefix: *primary-name*	Set resourceNamePrefix to control the name of the primary server. The default value is the same as the environment's name.
tag: *10.0-special*	To use a different image tag specifically for the primary server, uncomment this value and provide the desired tag. This overrides the tag specified in the common section.
nodeSelector: labelKey: *kubernetes.io/os* labelValue: *linux*	Specify a key and value that identifies nodes where the primary server pod runs.
networkLoadBalancer: annotations: - service.beta. kubernetes.io / azure-load- balancer-internal- subnet: *example- subnet* ipList: *- 4.3.2.1:primary.example.com*	Uncomment the annotations to specify additional primary server-specific annotations. These values are merged with the values given in the loadBalancerAnnotations above. Any duplicate values given here override the corresponding values above. Next, specify the hostname and IP address of the primary server.

Table 2-4 Environment parameters for the primary server *(continued)*

Paragraph	Description
credSecretName: *primary-credential-secret*	This determines the credentials for the primary server. Media servers use these credentials to register themselves with the primary server.
itAnalyticsPublicKey: *ssh-rsaxxx*	If using NetBackup IT Analytics, uncomment this and provide the SSH public key. IT Analytics uses this to access the primary server.
kmsDBSecret: *kms-secret*	Secret name which contains the Host Master Key ID (HMKID), Host Master Key passphrase (HMKpassphrase), Key Protection Key ID (KPKID) and Key Protection Key passphrase (KPKpassphrase) for NetBackup Key Management Service. The secret should be 'Opaque', and can be created either using a YAML or the following example command: `kubectl create secret generic kms-secret --namespace nb-namespace --from-literal=HMKID="HMK@ID" —from-literal=HMKpassphrase="HMK@passphrase" --from-literal=KPKID="KPK@ID" —from-literal=KPKpassphrase="KPK@passphrase"`
licenseKeys:	To specify additional license keys that are applied only to the primary server, uncomment this and provide the license key(s). In this example, the primary server would have the "X" license key defined in the previous section, followed by this "Y" key.
catalog: capacity: *100Gi* storageClassName: *standard*	This storage applies to the primary server for the NetBackup catalog and log volumes. The primary server catalog volume must be at least 100 Gi.
log: capacity: *30Gi* storageClassName: *standard*	Log volume must be at least 30Gi.

The following section describes the media server configurations. If you do not have a media server either remove this section from the configuration file entirely, or define it as an empty list.

Table 2-5 Media server related parameters

parameters	Description
mediaServers: - resourceNamePrefix: media1	This specifies media server configurations. This is given as a list of media servers, but most environments will have just one, with multiple replicas. It's also possible to have zero media servers; in that case, either remove the media servers section entirely, or define it as an empty list: mediaServers: []
replicas: *1*	Specifies the number of replicas of this media server. Minimum number of supported replicas is 1.
tag: *10.0-special*	To use a different image tag specifically for the media servers, uncomment this value and provide the desired tag. This overrides the tag specified above in the common table.
nodeSelector: labelKey: *kubernetes.io/os* labelValue: *linux*	Specify a key and value that identifies nodes where media-server pods will run.
data: capacity: *50Gi* storageClassName: *managed-premium-nbux*	This storage applies to the media server data volumes. The minimum data size for a media server is 50 Gi.
log capacity: *30Gi* storageClassName: *managed-premium-nbux*	This storage applies to the media server log volumes. Log volumes must be at least 30Gi.

Table 2-5 Media server related parameters *(continued)*

parameters	Description
networkLoadBalancer: annotations: - service.beta.kubernetes.io/ azure-load-balancer -internal-subnet: *example-subnet* ipList: ipAddr: *4.3.2.2* fqdn: *media1-1.example.com* ipAddr: *4.3.2.3* fqdn: *media1-2.example.com*	Uncomment annotations to specify additional media-server specific annotations. These values are merged with the values given in the loadBalancerAnnotations. The duplicate values given here, override the corresponding values in the loadBalancerAnnotations. The number of entries in the IP list should match the replica count specified above.
licenseKeys:	Uncomment annotations to specify any license key that is specifically applicable to the media server(s).

The following section describes MSDP-related parameters. You may also deploy without any MSDP scaleouts. In that case, remove the msdpScaleouts section entirely from the configuration file.

Table 2-6 MSDP Scaleout related parameters

Parameter	Description
msdpScaleouts: - resourceNamePrefix: dedupe1	This specifies MSDP Scaleout configurations. This is given as a list, but it would be rare to need more than one scaleout deployment in a single environment. Use the \`replicas\` property below to scale out. It's also possible to have zero MSDP scaleouts; in that case, either remove the msdpScaleouts section entirely, or define it to an empty list: msdpScaleouts: []
tag: '*16.0*'	This tag overrides the one defined in the table 1-3. It is necessary because the MSDP Scaleout images are shipped with tags different from the NetBackup primary and media images.

Table 2-6 MSDP Scaleout related parameters *(continued)*

Parameter	Description
replicas: *4*	This is the scaleout size of this MSDP Scaleout component. It is a required value, and it must be between 4 and 16 inclusive. **Note:** Scale-down of the MSDP Scaleout replicas after deployment is not supported.
serviceIPFQDNs: ipAddr: 1.2.3.4 fqdn: dedupe1-1.example.com ipAddr: 1.2.3.5 fqdn: dedupe1-2.example.com ipAddr: 1.2.3.6 fqdn: dedupe1-3.example.com ipAddr: 1.2.3.7 fqdn: dedupe1-4.example.com	These are the IP addresses and host names of the MSDP Scaleout servers. The number of the entries should match the number of the replicas specified above.
kms: keyGroup: *example-key-group*	Specifies the initial key group and key secret to be used for KMS encryption. When reusing storage from a previous deployment, the key group and key secret may already exist. In this case, provide the keyGroup only.
keySecret: *example-key-secret*	Specify keySecret only if the key group does not already exist and needs to be created. The secret type should be Opaque, and you can create the secret either using a YAML or the following command: `kubectl create secret generic example-key-secret --namespace nb-namespace --from-literal=username="devuser" --from-literal=passphrase="test passphrase"`

Table 2-6 MSDP Scaleout related parameters *(continued)*

Parameter	Description
loadBalancerAnnotations: service.beta.kubernetes .io/azure-load-balancer-internal: *true*	For MSDP scaleouts, the default value for the azure-load-balancer-internal annotation is \`false\`, which may cause the MSDP Scaleout services in this Environment to be accessible publicly. To make sure that they use private IP addresses, specify \`true\` here or in the loadBalancerAnnotations above in Table 1-3.
credential: secretName: *msdp-secret1*	This defines the credentials for the MSDP Scaleout server. It refers to a secret in the same namespace as this environment resource. Secret can be either of type 'Basic-auth' or 'Opaque'. You can create secrets using a YAML or by using the following command:`kubectl create secret generic <msdp-secret1> --namespace <nb-namespace> --from-literal=username=<"devuser"> --from-literal=password=<"Y@123abCdEf">`
autoDelete: *false*	Optional parameter. Default value is true. When set to true, the MSDP Scaleout operator deletes the MSDP secret after using it. In such case, the MSDP and primary secrets must be distinct. To use the same secret for both MSDP scaleouts and the primary server, set autoDelete to false.
catalog: capacity: *1Gi* storageClassName: *standard*	This storage applies to MSDP Scaleout to store the catalog and metadata. The catalog size may only be increased for capacity expansion. Expanding the existing catalog volumes cause short downtime of the engines. Recommended size is 1/100 of backend data capacity.

Table 2-6 MSDP Scaleout related parameters *(continued)*

Parameter	Description
dataVolumes: capacity: *5Gi* storageClassName: *standard*	This specifies the data storage for this MSDP Scaleout resource. You may increase the size of a volume or add more volumes to the end of the list, but do not remove or re-order volumes. Maximum 16 volumes are allowed. Appending new data volumes or expanding existing ones will cause short downtime of the Engines. Recommended volume size is 5Gi-32Ti.
log: capacity: *20Gi* storageClassName: *standard*	Specifies log volume size used to provision Persistent Volume Claim for Controller and MDS Pods. In most cases, 5-10 Gi capacity should be big enough for one MDS or Controller Pod to use.
nodeSelector: labelKey: *kubernetes.io/os* labelValue: *linux*	Specify a key and value that identifies nodes where MSDP Scaleout pods will run.

Edit restricted parameters post deployment

Do not change these parameters post initial deployment. Changing these parameters may result in an inconsistent deployment.

Table 2-7 Edit restricted parameters post deployment

Parameter	Description
resourceNamePrefix	Specifies the prefix name for the primary, media, and MSDP Scaleout server resources.

Table 2-7 Edit restricted parameters post deployment *(continued)*

Parameter	Description

ipAddr, fqdn and
 	
loadBalancerAnnotations
	The values against `ipAddr`, `fqdn` and `loadBalancerAnnotations` against following fields should not be changed post initial deployment. This is applicable for primary, media, and MSDP Scaleout servers. For example:
	
	`- The loadBalancerAnnotations for loadBalancerAnnotations: service.beta.kubernetes.io/azure-load-balancer -internal-subnet: example-subnet service.beta.kubernetes.io/azure-load-balancer -internal: 'true'`
	
	`- The IP and FQDNs values defined for Primary, Media and MSDPScaleout`
	`  ipList:`
	`    - ipAddr: 4.3.2.1`
	`      fqdn: primary.example.com`
	
	`  ipList:`
	`    - ipAddr: 4.3.2.2`
	`      fqdn: media1-1.example.com`
	`    - ipAddr: 4.3.2.3`
	`      fqdn: media1-2.example.com`
	
	`  serviceIPFQDNs:`
	`    - ipAddr: 1.2.3.4`
	`      fqdn: dedupe1-1.example.com`
	`    - ipAddr: 1.2.3.5`
	`      fqdn: dedupe1-2.example.com`
	`    - ipAddr: 1.2.3.6`
	`      fqdn: dedupe1-3.example.com`
	`    - ipAddr: 1.2.3.7`
	`      fqdn: dedupe1-4.example.com`

Uninstalling NetBackup environment and the operators

You can uninstall the NetBackup primary, media, and MSDP Scaleout environment and the operators as required. You need to uninstall the NetBackup environment before you uninstall the operators.

Note: Replace the environment custom resource names as per your configuration in the steps below.

To uninstall the NetBackup environment

1 To remove the environment components from the application namespace, run:

```
$ kubectl delete
environment.netbackup.veritas.com/environment-sample --namespace
<namespce_name>
```

2 Wait for all the pods, services and resources to be terminated. To confirm, run

```
$ kubectl get --namespace <namespce_name>
all,environments,primaryservers,mediaservers,msdpscaleouts
```

You should get a message that no resources were found in the *nb-example* namespace.

3 To identify and delete any outstanding persistent volume claims, run the following:

```
$ kubectl get pvc --namespace <namespce_name>
```

```
$ kubectl delete pvc <pvc-name>
```

4 To locate and delete any persistent volumes created by the deployment, run:

```
$ kubectl get pv
```

```
$ kubectl delete pv <pv-name> --grace-period=0 --force
```

Note: Certain storage drivers may cause physical volumes to get stuck in the terminating state. To resolve this issue, remove the finalizer, using the command: `$ kubectl patch pv <pv-name> -p '{"metadata":{"finalizers":null}}'`

---

5 To delete the application namespace, run:

```
$ kubectl delete ns <namespace name>
```

To uninstall the operators

1. To uninstall the NetBackup operator run the following command from the installation directory.

   ```
   $ kubectl delete -k operator
   ```

2. To uninstall the MSDP Scaleout operator and remove the operator's namespace, run.

   ```
   $ kubectl msdp delete --namespace <namespace name>
   ```

Note: Do not remove the MSDP Scaleout operator first as it may corrupt the NetBackup operator.

Applying security patches

This section describes how to apply security patches for operator and application images.

In the instructions below, we assume that the operators were deployed to the *netbackup-operator-system* namespace (the default namespace suggested by the deployment script), and that an environment resource named *nb-env* was deployed to a namespace named *nb-example*.

Although it is not necessary to manually shut down NetBackup primary server or media servers, it's still a good idea to quiesce scheduling so that no jobs get interrupted while pods are taken down and restarted.

Prepare the images

To prepare the images to apply patches

1. Unpack the tar file on a system where docker is able to push to the container registry, and `kubectl` can access the cluster.

2. Decide on a unique tag value to use for MSDP Scaleout images. For example, 16.0-update1. Set the **DD_TAG** environment variable accordingly and run `deploy.sh`:

   ```
   DD_TAG=16.0-update1 ./deploy.sh
   ```

3. In the menu that appears, select option **1** to install the operators.

4. Enter the fully qualified domain name of the container registry. For example: *exampleacr.azurecr.io*. When the script prompts to load images, answer *yes*.

5 When the script prompts to tag and push images, wait. Open another terminal window and re-tag the MSDP Scaleout images as:

```
docker tag msdp-operator:16.0 msdp-operator:16.0-update1
```

```
docker tag uss-controller:16.0 uss-controller:16.0-update1
```

```
docker tag uss-engine:16.0 uss-engine:16.0-update1
```

```
docker tag uss-mds:16.0 uss-mds:16.0-update1
```

6 Return to the deploy script and when prompted, enter *yes* to tag and push the images. Wait for the images to be pushed, and then the script will pause to ask another question. The remaining questions are not required, so press Ctrl+c to exit the deploy script.

Update the NetBackup operator

1 Get the image ID of the existing NetBackup operator container and record it for later. Run:

```
kubectl get pod -n netbackup-operator-system -l
nb-control-plane=nb-controller-manager -o jsonpath --template
"{.items[*].status.containerStatuses[?(@.name=='netbackup-operator')].imageID}{'\n'}"
```

The command prints the name of the image and includes the SHA-256 hash identifying the image. For example:

exampleacr.azurecr.io/netbackup/operator @sha256:59d4d46d82024a1ab6353 33774c8e19eb5691f3fe988d86ae16a0c5fb636e30c

2 To restart the NetBackup operator, run:

```
pod=$(kubectl get pod -n netbackup-operator-system -l
nb-control-plane=nb-controller-manager -o jsonpath --template
'{.items[*].metadata.name}')
```

```
kubectl delete pod -n netbackup-operator-system $pod
```

3 Re-run the `kubectl` command from earlier to get the image ID of the NetBackup operator. Confirm that it's different from what it was before the update.

Update the MSDP Scaleout operator

1 Get the image ID of the existing MSDP Scaleout operator container and save it for later use. Run:

```
kubectl get pods -n netbackup-operator-system -l
control-plane=controller-manager -o jsonpath --template
"{.items[*].status.containerStatuses[?(@.name=='manager')].imageID}{'\n'}"
```

2 Re-initialize the MSDP Scaleout operator using the new image.

```
kubectl msdp init -n netbackup-operator-system --image
exampleacr.azurecr.io/msdp-operator:16.0-update1
```

3 Re-run the `kubectl` command from earlier to get the image ID of the MSDP Scaleout operator. Confirm that it's different from what it was before the update.

Update the primary server or media servers

1 Look at the list of pods in the application namespace and identify the pod or pods to update. The primary-server pod's name typically end with "primary-0" and media-server pods end with "media-0", "media-1", etc. Hereafter, pod will be referred to as $pod. Run:

```
kubectl get pods -n nb-example
```

2 Get the image ID of the existing NetBackup container and record it for later. Run:

```
kubectl get pods -n nb-example $pod -o jsonpath --template
"{.status.containerStatuses[*].imageID}{'\n'}"
```

3 Look at the list of StatefulSets in the application namespace and identify the one that corresponds to the pod or pods to be updated. The name is typically the same as the pod, but without the number at the end. For example, a pod named nb-primary-0 is associated with statefulset nb-primary. Hereafter the statefulset will be referred to as $set. Run:

```
kubectl get statefulsets -n nb-example
```

4 Restart the statefulset. Run:

```
kubectl rollout restart -n nb-example statefulset $set
```

The pod or pods associated with the statefulset are terminated and be re-created. It may take several minutes to reach the "Running" state.

5 Once the pods are running, re-run the kubectl command from step 2 to get the image ID of the new NetBackup container. Confirm that it's different from what it was before the update.

Update the MSDP Scaleout containers

1 Look at the list of pods in the application namespace and identify the pods to update. The controller pod have "uss-controller" in its name, the MDS pods have "uss-mds" in their names, and the engine pods are be named like their fully qualified domain names. Run:

```
kubectl get pods -n nb-example
```

2 Get the image IDs of the existing MSDP Scaleout containers and record them for later. All the MDS pods use the same image, and all the engine pods use the same image, so it's only necessary to get three image IDs, one for each type of pod.

```
kubectl get pods -n nb-example $engine $controller $mds -o
jsonpath --template "{range
.items[*]}{.status.containerStatuses[*].imageID}{'\n'}{end}"
```

3 Edit the Environment resource and change the *spec.msdpScaleouts[*].tag* values to the new tag used earlier in these instructions.

```
kubectl edit environment -n nb-example nb-env

...
spec:
  ...
  msdpScaleouts:
  - ...
    tag: "16.0-update1"
```

4 Save the file and close the editor. The MSDP Scaleout pods are terminated and re-created. It may take several minutes for all the pods to reach the "Running" state.

5 Run `kubectl get pods`, to check the list of pods and note the new name of the uss-controller pod. Then, once the pods are all ready, re-run the kubectl command above to get the image IDs of the new MSDP Scaleout containers. Confirm that they're different from what they were before the update.

Assessing cluster configuration before deployment

This chapter includes the following topics:

- How does the Config-Checker utility work
- Config-Checker execution and status details

How does the Config-Checker utility work

The Config-Checker utility performs checks on the deployment environment to verify that the environment meets the requirements, before starting the primary server and media server deployments.

How does the Config-Checker works:

- RetainReclaimPolicy check:
  This check verifies that the storage classes used for PVC creation in the CR have reclaim policy as **Retain.** The check fails if any of the storage classes do not have the **Retain** reclaim policy.
  Persistent Volumes Reclaiming
- MinimumVolumeSize check:
  This check verifies that the PVC storage capacity meets the minimum required volume size for each volume in the CR. The check fails if any of the volume capacity sizes does not meet the requirements.
  Following are the minimum volume size requirements:
  - Primary server:

- Catalog volume size: 100Gi
- Log volume size: 30Gi

- Media server:
  - Data volume size: 50Gi
  - Log volume size: 30Gi

- Disk check:
  This check verifies that the storage type used in defining the storage class is not Azure files, for the volumes in both Primary and Media servers. That is, catalog and log volumes in case of Primary server and, data and log volumes in case of Media server.

Config-Checker execution and status details

Note the following points.

- Config-Checker is executed as a separate job in Kubernetes cluster for both the primary server and media server CRs respectively. Each job creates a pod in the cluster. Config-checker creates the pod in the operator namespace.

  Note: Config-checker pod gets deleted after 4 hours.

- Execution summary of the Config-Checker can be retrieved from the Config-Checker pod logs using the `kubectl logs <configchecker-pod-name> -n <operator-namespace>` command.
  This summary can also be retrieved from the operator pod logs using the `kubectl logs <operator-pod-name> -n <operator-namespace>` command.
- Following are the Config-Checker modes that can be specified in the Primary and Media CR:
  - Default: This mode executes the Config-Checker. If the execution is successful, the Primary and Media CRs deployment is started.
  - Dryrun: This mode only executes the Config-Checker to verify the configuration requirements but does not start the CR deployment.
  - Skip: This mode skips the Config-Checker execution of Config-Checker and directly start the deployment of the respective CR.
- Status of the Config-Checker can be retrieved from the primary server and media server CRs by using the `kubectl describe <PrimaryServer/MediaServer> <CR name> -n <namespace>` command.

For example, `kubectl describe primaryservers environment-sample -n test`

- Following are the Config-Checker statuses:
  - Success: Indicates that all the mandatory config checks have successfully passed.
  - Failed: Indicates that some of the config checks have failed.
  - Running: Indicates that the Config-Checker execution is in progress.
  - Skip: Indicates that the Config-Checker is not executed because the `configcheckmode` specified in the CR is skipped.
- If the Config-Checker execution status is Failed, you can check the Config-Checker job logs using `kubectl logs <configchecker-pod-name> -n <operator-namespace>`. Review the error codes and error messages pertaining to the failure and update the respective CR with the correct configuration details to resolve the errors.
  For more information about the error codes, refer to NetBackup™ Status Codes Reference Guide.
- If Config-Checker ran in **dryrun** mode and if user wants to run Config-Checker again with same values in Primary or Media server YAML as provided earlier, then user needs to delete respective CR of Primary or Media server. And then apply it again.
  - If it is primary server CR, delete primary server CR using the `kubectl delete -f <environment.yaml>` command.
    Or
    If it is media server CR, edit the Environment CR by removing the media server section in the `environment.yaml` file. Before removing the **mediaServer** section, you must save the content and note the location of the content. After removing section apply environment CR using `kubectl apply -f <environment.yaml>` command.
  - Apply the CR again. Add the required data which was deleted earlier at correct location, save it and apply the yaml using `kubectl apply -f <environment.yaml>` command.

Deploying NetBackup

This chapter includes the following topics:

Preparing the environment for NetBackup installation on AKS

Refer to this section to prepare your host system and Azure Kubernetes cluster for NetBackup installation.

AKS-specific requirements

Use the following checklist to prepare the AKS for installation.

- Your Azure Kubernetes cluster must be created with appropriate network and configuration settings.
  Supported Kubernetes cluster version is 1.21.x and later.

- While creating the cluster, assign appropriate roles and permissions.
  Concepts - Access and identity in Azure Kubernetes Services (AKS) - Azure Kubernetes Service | Microsoft Docs
- Use an existing Azure container registry or create a new one. Your Kubernetes cluster must be able to access this registry to pull the images from the container registry. For more information on the Azure container registry, see Azure Container Registry documentation.
- A dedicated node pool for NetBackup must be created with manual scaling or Autoscaling enabled in Azure Kubernetes cluster. The autoscaling feature allows your node pool to scale dynamically by provisioning and de-provisioning the nodes as required automatically.
  The following table lists the node configuration for the primary and media servers.

Node type		D16ds v4
Disk type		P30
vCPU		16
RAM		64 GiB
Total disk size per node (TiB)		1 TB
Number of disks/node		1
Cluster storage size	Small (4 nodes)	4 TB
	Medium (8 nodes)	8 TB
	Large (16 nodes)	16 TB

- All the nodes in the node pool must be running the Linux operating system.
- If you want to use static public IPs, private IPs and fully qualified domain names for the load balancer service, the public IP addresses, private IP addresses and FQDNs must be created in AKS before deployment.
- If you want to bind the load balancer service IPs to a specific subnet, the subnet must be created in AKS and its name must be updated in the **annotations** key in the **networkLoadBalancer** section of the custom resource (CR).
  For more information on the network configuration for a load balancer service, refer to the *How-to-Guide* section of the Azure documentation.
  For more information on managing the load balancer service, see About the Load Balancer service.

Host-specific requirements

Use the following checklist to address the prerequisites on the system that you want to use as a NetBackup host that connects to the AKS cluster.

- Linux operating system: For a complete list of compatible Linux operating systems, refer to the Software Compatibility List (SCL) at:
  NetBackup Compatibility List for all Versions
  https://sort.veritas.com/netbackup
- Install Docker on the host to install NetBackup container images through tar, and start the container service.
  https://docs.docker.com/engine/install/
- Prepare the host to manage the AKS cluster.
  - Install Azure CLI.
    https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-linux/
  - Install Kubernetes CLI
    https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/
  - Log in to the Azure environment to access the Kubernetes cluster by running this command on Azure CLI:
    ```
    # az login -identity
    # az account set --subscription <subscriptionID>
    # az aks get-credentials --resource-group
    <resource_group_name> --name <cluster_name>
    ```
  - Log in to the container registry:
    ```
    # az acr login -n <container-registry-name>
    ```

Recommendations of NetBackup deployment on AKS

Note the following recommendations:

- Use Azure Premium storage for catalog volume in primary server CR and data volume in media server CR.
- Use Azure Standard storage for log volume in primary server CR and media server CR.
- Do not delete the disk linked to PV used in primary server and media server CR deployment. This may lead to data loss.
- Ensure that in one cluster, only one NetBackup operator instance is running.

- Do not edit any Kubernetes resource created as part of primary server and media server custom resource. Update is supported through custom resource update only.
- Detailed primary server custom resource deployment and media server custom resource deployment logs are retrieved from NetBackup operator pod logs using the `kubectl logs <netbackup-operator-pod-name> -c netbackup-operator -n <netbackup operator-namespace>` command .
- Deploy primary server custom resource and media server custom resource in same namespace.
- Ensure that you follow the symbolic link and edit the actual persisted version of the file, if you want to edit a file having a symbolic link in the primary server or media server.

Limitations of NetBackup deployment on AKS

Note the following limitations:

- A storage class that has the storage type as `Azure file` is not supported. When the Config-Checker runs the validation for checking the storage type, the Config-Checker job fails if it detects the storage type as `Azure file`. But if the Config-Checker is skipped then this validation is not run, and there can be issues in the deployment. There is no workaround available for this limitation. You must clean up the PVCs and CRs and reapply the CRs.
- Media server scale down is not supported. Certain workloads that require media server affinity for the clients would not work.
- External Certificate Authority (ECA) is not supported.
- In case of load balancer service updating the CR with dynamic IP address to static IP address and vice versa is not allowed.
- Media server pods as NetBackup storage targets are not supported. For example, NetBackup storage targets like AdvancedDisk and so on are not supported on the media server pods.

About primary server CR and media server CR

Primary server custom resource is used to deploy the NetBackup primary server and media server custom resource is used to deploy the NetBackup media server.

- After the operator is installed, update the custom resource YAMLs to deploy the primary server and media server CRs located in the `samples` folder. CR yamls are located in `samples` folder where package is untarred.

- The primary server CRD and media server CRD are located in `operator_deployment.yaml` in the operator folder where the package is extracted.
- **ResourcePrefixName** used in the primary server and media server CRs must not be same. In the primary server CR the **ResourcePrefixName** should not contain the word **media** and in the media server CR the **ResourcePrefixName** should not contain the word **primary**.

  Note: After deployment, you cannot change the **ResourcePrefixName** in primary server and media server CR.

- Before the CRs can be deployed, the utility called `Config-Checker` is executed that performs checks on the environment to ensure that it meets the basic deployment requirements. The config-check is done according to the **configCheckMode** and **paused** values provided in the custom resource YAML.
  For more information, refer to See "How does the Config-Checker utility work" on page 45.
- You can deploy the primary server and media server CRs in same namespace.
- Use the storage class that has the storage type as 'Azure disk' for the catalog and log volumes in the primary server CR, and data and log volumes in the media server CR.
- During fresh installation of the NetBackup servers, the value for **keep logs up to** under **log retention configuration** is set based on the log storage capacity provided in the respective primary server or media server CR inputs. You may change this value if required.
  - The NetBackup deployment sets the value as per the formula.
    Size of logs PVC/PV * 0.8 = Keep logs up value By default, the default value is set to 24GB.
    For example: If the user configures the storage size in the CR as 40GB (instead of the default 30GB) then the default value for that option become 32GB automatically based on the formula.
- Deployment details of primary server and media server can be observed from the operator pod logs using the following command:

  ```
  kubectl logs <operator-pod-name> -c netbackup-operator -n
  <operator-namespace>
  ```

After Installing primary server CR

Note the following points:

- The primary server CR will create a pod, a statefulset, a load balancer service, a configmap, a persistent volume claim for log volume, and a persistent volume claim for catalog volume.
- Initially pod will be in not ready state (0/1) when installation is going on in the background. Check the pod logs for installation progress usingthe following command:

  ```
  kubectl logs <primary-pod-name> -n <namespace>
  ```

  Primary server can be considered as successfully installed and running when the primary server pod's state is **ready (1/1)** and the Statefulset is **ready (1/1)**.
- You can access the NetBackup webUI using the **primary server hostname** that was specified in the primary server CR status in **Primary server details** section.
  For example, if the primary server hostname is **nbu-primary**, then you can access the webUI at *https://nbu-primary/webui/login*.

After Installing the media server CR

Note the following points:

- The media server CR will create a pod, a statefulset, a configmap, a loadbalancer service, a persistent volume claim for data volume, and a persistent volume claim for log volume.
- Initially pod will be in not ready state (0/1) when installation is going in the background. Check the pod logs for installation progress using the following command:

  ```
  kubectl logs <media-pod-name> -n <namespace>
  ```

  Media server can be considered as successfully installed and running when the media server pod's state is **ready (1/1)**, and the Statefulset is **ready (1/1)**, for each replica count.
- Details of media server name for each replica can be obtained from media server CR status by running the following command:

  ```
  kubectl describe <MediaServer_cr_name> -n <namespace>
  ```

Monitoring the status of the CRs

You can view the status and other details of the primary server and media server CRs using the following commands:

- `kubectl get <PrimaryServer/MediaServer> -n <namespace>` or
- `kubectl describe <PrimaryServer/MediaServer> <CR name> -n <namespace>`

Following table describes the primary server CR and media server CR status fields:

Table 4-1

Section	Field / Value	Description
Primary Server Details Only one hostname and IP address for the respective primary server.	Host Name	Name of the primary server that should be used to access the web UI.
	IP	IP address to access the primary server.
Media Server Details Number of hostname and IP address is equal to the replica count mentioned CR spec.	Host Name	Name of the media server.
	IP	IP address to access the media server.
Attributes	Resource Name	Statefulset name of the respective server.
	Primary/Media server name	Name of the primary server or media server deployed.
	Config checker status	Indicates the status of the config checker as passed, failed, or skipped. For more information on the Config-Checker, seeHow does the Config-Checker utility work.
	SHA Fingerprint	Represents the SHA key fingerprint of the NetBackup primary server. **Note:** SHAFingerprint represents the SHA256 CA certificate fingerprint of the primary server.

Table 4-1 *(continued)*

Section	Field / Value	Description
Error Details	Code	A code assigned to an error encountered during the CR deployment or during the config-check operation. For more information on the error, refer to the *NetBackup Status Code Reference Guide*.
	Message	Message that describes the respective error code.
State	Success /Paused /Failed /Running	Current state of the custom resource, from one of the following: ■ Success: Indicates that the deployment of Primary/Media Custom Resource (CR) is successful. However, this does not mean that the installation of the NetBackup primary/media servers is successful. The primary or media server StatefulSets and/or the pods might or might not be in a **ready** state, irrespective of that the Primary/Media CR state will show as Success. ■ Paused: Indicates that the reconciler is in **paused** state and deployment of a CR is paused. ■ Failed: Indicates that the deployment of a CR failed with errors. However this does not mean failed installation of the NetBackup Server. Errors can be analyzed from the Operator logs or CR **describe**. ■ Running: Indicates that the CR deployment is in progress and the resources are getting created.

Updating the CRs

After the successful deployment of the primary server and media server CRs, you can update the values of only selected specs by editing the respective environment custom resource.

Note: Updating the Kubernetes resources (pod, configmap, services, statefulset etc) created for the CRs is not recommended.

Following tables describe the specs that can be edited for each CR.

Table 4-2 Primary server CR

Spec	Description
paused	Specify *True* or *False* as a value, to temporarily stop the respective CR controller. *True*: Stop the controller. *False*: Resume the controller.
configCheckMode	Specify *default*, *dryrun* or *skip* as a value. See "Config-Checker execution and status details" on page 46.

Table 4-3 Media server CR

Spec	Description
paused	Specify *True* or *False* as a value, to temporarily stop the respective CR controller. *True*: Stop the controller. *False*: Resume the controller.
replicas	Represents the replica count of the media server. Media server count can be scaled up by incrementing the replica count. Reducing the replica count is not supported.
configCheckMode	Specify *default*, *dryrun* or *skip* as a value. See "Config-Checker execution and status details" on page 46.

If you edit any other fields, the deployment can go into an inconsistent state.

Deleting the CRs

If you must delete any of the CRs for a valid reason such as for the troubleshooting purpose, or because any of the specs provided were incorrect; you can reinstall the deleted CR after resolving the issues.

Note: Once installed, deleting a CR is not recommended as it will stop the deployment and NetBackup will not work.

Notes:

- Deleting a CR will delete all its child resources like pod, statefulset, services, configmaps, config checker job, config checker pod.
- Deleting operator with `kubectl delete -k <operator_folder_path>` will delete the CRs and its resources except the PVC.
- Persistent volume claim (PVC) will not be deleted upon deleting a CR so that the data is retained in the volumes. Then if you create a new CR with the same name as the deleted one, the existing PVC with that same name will be automatically linked to the newly created pods.
- Do not delete "/mnt/nbdata" and "/mnt/nblogs" folders manually from primary server and media pods. The NetBackup deployment will go into an inconsistent state and will also result in data loss.

Configuring NetBackup IT Analytics for NetBackup deployment

NetBackup IT Analytics can be configured to use with NetBackup primary server in this Kubernetes environment. NetBackup IT Analytics can be configured at the time of primary server deployment or user can update the primary server CR to configure NetBackup IT Analytics.

To configure NetBackup IT Analytics for NetBackup deployment

1. Using the `ssh-keygen` command, generates public key and private key on NetBackup IT Analytics data collector.

   NetBackup IT Analytics data collector uses passwordless ssh login.

2. Update the primary server CR, copy public key generated in previous steps to “itAnalyticsPublicKey” section in spec.

   - Apply the primary server CR changes using `kubectl apply -f environment.yaml -n <namespace>`.

On successfully deployment of primary server CR, describe the primary server CR using `kubectl describe PrimaryServer <primary-server-name> -n <namespace>`

- In status section, verify **It Analytics Configured** is set to **true**.
  For more information, refer to the NetBackup™ Web UI Administrator's Guide.

3 Create and copy NetBackup API key from NetBackup web UI.

4 On NetBackup IT Analytics portal:

- Navigate to **Admin > Collector Administration > Select respective data collector > Add policy > Veritas NetBackup > Add**.
- Add required options, specify the NetBackup API in the **API Key** field, and then click **OK**.
- Select newly added primary server from **NetBackup Master Servers** and provide **nbitanalyticsadmin** as **Master Server User ID**.
- Provide **privateKey=<path-of-private-key>|password=<passphrase>** as **Master Server Password** and **Repeat Password** whereas **<path-of-private-key>** is the private key created using ssh-keygen in earlier steps and **<passphrase>** is the passphrase used while creating private key via ssh-keygen.
- Provide appropriate data to data collector policy fields and select collection method as **SSH or WMI protocol to NetBackup Master Server**.

Configuring the primary server with NetBackup IT Analytics tools is supported only once from primary server CR.

For more information about IT Analytics data collector policy, see Add a Veritas NetBackup Data Collector policy and for more information about adding NetBackup Primary Servers within the Data Collector policy, see Add/Edit NetBackup Master Servers within the Data Collector policy.

To change the already configured public key

1 Execute the following command in the primary server pod:

```
kubectl exec -it -n <namespace> <primaryServer-pod-name> --
/bin/bash
```

2 Copy the new public keys in the `/home/nbitanalyticsadmin/.ssh/authorized_keys` and `/mnt/nbdata/.ssh/nbitanalyticsadmin_keys` files.

3 Restart the **sshd** service using the `systemctl restart sshd` command.

Managing NetBackup deployment using VxUpdate

VxUpdate package is not shipped with the NetBackup deployment package. You must download and add it in NetBackup primary server.

To manage NetBackup deployment using VxUpdate

1. Download the required VxUpdate package on the docker-host used to interact with AKS cluster.
2. Copy the VxUpdate package to primary server pod using the `kubectl cp <path-vxupdate.sja> <primaryServerNamespace>/<primaryServer-pod-name>:<path-on-primary-pod>` command.
3. After VxUpdate package is available on primary server Pod, add it to NetBackup repository using any one of the following:
   - Select the VxUpdate package from the NetBackup web UI.
   - Execute the following command in the primary server pod:
     ```
     kubectl exec -it -n <primaryserver-namespace>
     <primaryServer-pod-name> -- bash
     ```
     And run the following command:
     ```
     nbrepo -a <vxupdate-package-path-on-PrimaryPod>
     ```

   After adding the VxUpdate package to `nbrepo`, this package is persisted even after pod restarts.

Migrating the node pool for primary or media servers

You can migrate the node pool for primary or media servers.

To migrate the node pool for primary or media servers

1. Edit environment CR object using the `<environmentCR_name> -n <namespace>` command.
   - Remove **keySecret** name and **passphrase** field from **msdpScaleout** section in CR object.

- Set **paused** field to **true** in primary or media section and save.

2 Edit primary or media statefulset using the `kubectl edit statefulset.apps/<sts-name> -n netbackup-environment` command , set replica count to **0** and save.

The pods are deleted.

3 Run the `kubectl describe statefulset.apps/<sts-name> -n <namespace>` command:

Wait till the pod status displays **0 Running**. Depending on replica count and server load, time taken to delete the pods may change.

During the node pool migration for the primary server, media server or MSDP Scaleout pods may go into a not ready state (0/1). After the primary server pod goes into a running state, the media server or msdp pods will be in ready state.

4 Edit the primary or media statefulset using the following command and update node selector with label key and value for new node pool and save

```
kubectl edit statefulset.apps/<sts-name> -n <namespace>
```

5 Edit environment CR object, update node selector with the same node selector value and set **paused** to **false**.

Pods are created on new node pool with the same persistent storage volumes. All the pods will be in ready and running state.

Chapter 5

Deploying MSDP Scaleout

This chapter includes the following topics:

Deploying MSDP Scaleout

You must deploy MSDP Scaleout solution in your Azure Kubernetes Cluster environment. AKS must be created with appropriate network and configuration settings.

You can have multiple MSDP Scaleout deployments in the same AKS cluster. Ensure that each MSDP Scaleout deployment runs in a dedicated namespace on a dedicated node pool.

Before you deploy the solution, ensure that your environment meets the requirements.

See "Prerequisites" on page 62.

Table 5-1 MSDP Scaleout deployment steps

Step	Task	Description
Step 1	Install the docker images and binaries.	See "Installing the docker images and binaries" on page 64.
Step 2	Initialize MSDP operator.	See "Initializing the MSDP operator" on page 65.
Step 3	Configuring MSDP Scaleout.	See "Configuring MSDP Scaleout" on page 66.
Step 4	Use MSDP Scaleout as a single storage pool in NetBackup.	See "Using MSDP Scaleout as a single storage pool in NetBackup" on page 68.

See "Cleaning up MSDP Scaleout" on page 115.

See "Cleaning up the MSDP Scaleout operator" on page 116.

Prerequisites

A working Azure Kubernetes cluster (AKS cluster)

- Azure Kubernetes cluster
  - Your Azure Kubernetes cluster must be created with appropriate network and configuration settings.
    Supported Azure Kubernetes cluster version is 1.21.x and later.
  - Availability zone for AKS cluster must be disabled.
  - At least one storage class is backed with Azure disk CSI storage driver "disk.csi.azure.com", and allows volume expansion. It must be in LRS category with Premium SSD. For example, the built-in storage class "managed-csi-premium". It is recommended that the storage class has "Retain" reclaim.
  - Cert-Manager must be installed.
  - A Kubernetes Secret that contains the MSDP credentials is required.
    See " Secret" on page 142.
  - Enable AKS Uptime SLA.
    AKS Uptime SLA is recommended for a better resiliency.
    For information about AKS Uptime SLA and to enable it, see Azure Kubernetes Service (AKS) Uptime SLA.

- Azure container registry (ACR)
  Use existing ACR or create a new one. Your Kubernetes cluster must be able to access this registry to pull the images from.
- Node Pool
  You must have a dedicated node pool for MSDP Scaleout created. Azure availability zone must be disabled.
  The Azure autoscaling allows your node pool to scale dynamically as required. If Azure autoscaling is not enabled, ensure the node number is not less than MSDP Scaleout size.
  It is recommended that you set the minimum node number to 1 or more to bypass some limitations in AKS.
- Client machine to access AKS cluster
  - A separate computer that can access and manage your AKS cluster and ACR.
  - It must have Linux operating system.
  - It must have Docker daemon, the Kubernetes command-line tool (kubectl), and Azure CLI installed.
    The Docker storage size must be more than 6 GB. The version of kubectl must be v1.19.x or later. The version of Azure CLI must meet the AKS cluster requirements.
  - If AKS is a private cluster, see Create a private Azure Kubernetes Service cluster.
- Static Internal IPs
  If the internal IPs are used, reserve the internal IPs (avoid the IPs that are reserved by other systems) for MSDP Scaleout and add DNS records for all of them in your DNS configuration.
  The Azure static public IPs can be used but is not recommended.
  If Azure static public IPs are used, create them in the node resource group for the AKS cluster. A DNS name must be assigned to each static public IP. The IPs must be in the same location of the AKS cluster.

Existing NetBackup environment

MSDP Scaleout connects to the existing NetBackup environment with the required network ports 1556 and 443. The NetBackup primary server should be 10.0 or later. The NetBackup environment can be anywhere, locally or remotely. It may or may not be in AKS cluster. It may or may not be in the same AKS cluster.

If the NetBackup servers are on Azure cloud, besides the NetBackup configuration requirements, the following settings are recommended. They are not MSDP-specific requirements, they just help your NetBackup environment run smoothly on Azure cloud.

- Add the following in `/usr/openv/netbackup/bp.conf`

  ```
  HOST_HAS_NAT_ENDPOINTS = YES
  ```

- Tune sysctl parameters as follows:

  ```
  net.ipv4.tcp_keepalive_time=120

  net.ipv4.ip_local_port_range = 14000 65535

  net.core.somaxconn = 1024
  ```

  Tune the max open files to 1048576 if you run concurrent jobs.

Installing the docker images and binaries

The MSDP package `VRTSpddek.tar.gz` for Kubernetes includes the following:

- A docker image for MSDP operator
- 3 docker images for MSDP Scaleout: uss-controller, uss-mds, and uss-engine
- A kubectl plugin: kubectl-msdp

To install the docker images and binaries

1. Download `VRTSpddek.tar.gz` from the Veritas site.
2. Load the docker images to your docker storage.

   ```
   tar -zxvf VRTSpddek.tar.gz

   ls VRTSpddek-*/images/*.tar.gz|xargs -i docker load -i {}
   ```

3. Copy MSDP kubectl plugin to a directory from where you access AKS host. This directory can be configured in the PATH environment variable so that kubectl can load kubectl-msdp as a plugin automatically.

   For example,

   ```
   cp ./VRTSpddek-*/bin/kubectl-msdp /usr/local/bin/
   ```

4. Push the docker images to the ACR.

   ```
   docker login <your-acr-url>
   for image in msdp-operator uss-mds uss-controller uss-engine; do \
       docker image tag $image:<version> <your-acr-url>/$image:<version>; \
       docker push <your-acr-url>/$image:<version>; \
   done
   ```

Initializing the MSDP operator

Run the following command to initialize MSDP operator.

```
kubectl msdp init -i <acr-url>/msdp-operator:<version> -s
<storage-class-name> [-l agentpool=<nodepool-name>]
```

You can use the following `init` command options.

Table 5-2 init command options

Option	Description
`-i`	MSDP operator images on your ACR.
`-s`	The storage class name.
`-l`	Node selector of the MSDP operator. By default, each node pool has a unique label with key-value pair *agentpool=<nodepool-name>*. If you have assigned a different and cluster-wise unique label for the node pool, you can use that instead of *agentpool*.
`-c`	Core pattern of the operator pod. Default value: "/core/core.%e.%p.%t"
`-d`	Enable debug-level logging in MSDP operator.

Table 5-2 init command options *(continued)*

Option	Description
`-a`	The maximum number of days to retain the old log files. Range: 1-365 Default value: 28
`-u`	The maximum number of old log files to retain. Range: 1-20 Default value: 20
`-n`	Namespace scope for this request. Default value: msdp-operator-system
`-o`	Generate MSDP operator CRD YAML.
`-h`	Help for the `init` command.

This command installs Custom Resource Definitions (CRD) **msdpscaleouts.msdp.veritas.com** and deploys MSDP operator in the Kubernetes environment. MSDP operator runs with Deployment Kubernetes workload type with single replica size in the default namespace **msdp-operator-system**.

MSDP operator also exposes the following services:

- Webhook service
  The webhook service is consumed by Kubernetes api-server to mutate and validate the user inputs and changes of the MSDP CR for the MSDP Scaleout configuration.
- Metrics service
  The metric service is consumed by Kubernetes/AKS for Azure Container Insight integration.

You can deploy only one MSDP operator instance in an AKS cluster.

Run the following command to check the MSDP operator status.

```
kubectl -n msdp-operator-system get pods -o wide
```

Configuring MSDP Scaleout

After you push the docker images to ACR and initialize MSDP operator, configure MSDP Scaleout.

To configure MSDP Scaleout

1. Create a dedicated namespace for MSDP Scaleout to run.

   ```
   kubectl create ns <sample-namespace>
   ```

2. Create an MSDP Scaleout Secret. The Secret is used in CR.

   ```
   kubectl apply -f <secret-yaml-file>
   ```

   See " Secret" on page 142.

3. Display the custom resource (CR) template.

   ```
   kubectl msdp show -c
   ```

4. Save the CR template.

   ```
   kubectl msdp show -c -f <file path>
   ```

5. Edit the CR file in the text editor.

6. Apply the CR file to the AKS cluster.

   Caution: Add `MSDP_SERVER = <first Engine FQDN>` in `/usr/openv/netbackup/bp.conf` file on the NetBackup primary server before applying the CR YAML.

   ```
   kubectl apply -f <sample-cr-yaml>
   ```

7 Monitor the configuration progress.

```
kubectl get all -n <namespace> -o wide
```

In the **STATUS** column, if the readiness state for the controller, MDS and engine pods are all **Running**, it means that the configuration has completed successfully.

In the **READINESS GATES** column for engines, 1/1 indicates that the engine configuration has completed successfully.

8 If you specified `spec.autoRegisterOST.enabled: true` in the CR, when the MSDP engines are configured, the MSDP operator automatically registers the storage server, a default disk pool, and a default storage unit in the NetBackup primary server.

A field **ostAutoRegisterStatus** in the Status section indicates the registration status. If **ostAutoRegisterStatus.registered** is **True**, it means that the registration has completed successfully.

You can run the following command to check the status:

```
kubectl get msdpscaleouts.msdp.veritas.com -n <sample-namespace>
```

You can find the storage server, the default disk pool, and storage unit on the Web UI of the NetBackup primary server.

Using MSDP Scaleout as a single storage pool in NetBackup

If you did not enable automatic registration of the storage server (autoRegisterOST) in the CR, you can configure it manually using the NetBackup Web UI.

See “ MSDP Scaleout CR” on page 143.

To use MSDP Scaleout as a single storage pool in NetBackup

1. Follow the OpenStorage wizard with storage type **PureDisk** to create the storage server using the first Engine FQDN.

   MSDP storage server credentials are defined in the Secret resource.

   For more information, see *Create a Cloud storage, OpenStorage, or AdvancedDisk storage server* topic of the *NetBackup Web UI Administrator's Guide*.

2. Follow the MSDP wizard to create the disk pool.

   For more information, see *Create a disk pool* topic of the *NetBackup Web UI Administrator's Guide*.

3. Follow the MSDP wizard to the storage unit.

   For more information, see *Create a disk pool* topic of the *NetBackup Web UI Administrator's Guide*.

You can use MSDP Scaleout like the legacy single-node MSDP.

Configuring the MSDP cloud in MSDP Scaleout

After you configure the local LSU, you can also configure MSDP cloud in MSDP Scaleout.

For more information about MSDP cloud support, see the *NetBackup Deduplication Guide*.

Chapter 6

Monitoring NetBackup

This chapter includes the following topics:

Monitoring the application health

Kubernetes Liveness and Readiness probes are used to monitor and control the health of the NetBackup primary server and media server pods. The probes collectively also called as health probes, keep checking the availability and readiness of the pods, and take designated actions in case of any issues. The kubelet uses liveness probes to know when to restart a container, and readiness probes to know when a container is ready. For more information, refer to the Kubernetes documentation.

Configure Liveness, Readiness and Startup Probes | Kubernetes

The health probes monitor the following for the NetBackup deployment:

- Mount directories are present for the data/catalog at `/mnt/nbdata` and the log volume at `/mnt/nblogs`.
- `bp.conf` is present at `/usr/openv/netbackup`
- NetBackup services are running as expected.

Following table describes the actions and time intervals configured for the probes:

Table 6-1

Action	Description	Probe name	Primary server (seconds)	Media server (seconds)
Initial delay	This is the delay that tells kubelet to wait for a given number of seconds before performing the first probe.	Readiness Probe	120	60
		Liveness Probe	300	90
Periodic execution time	This action specifies that kubelet should perform a probe every given number of seconds.	Readiness Probe	30	30
		Liveness Probe	90	90
Threshold for failure retries	This action specifies that kubelet should retry the probe for given number of times in case a probe fails, and then restart a container.	Readiness Probe	1	1
		Liveness Probe	5	5

Heath probes are run using the `nbu-health` command. If you want to manually run the `nbu-health` command, the following options are available:

- **Disable**
  This option disables the health check that will mark pod as not ready (0/1).
- **Enable**
  This option enables the already disabled health check in the pod. This marks the pod in ready state(1/1) again if all the NetBackup health checks are passed.
- **Deactivate**
  This option deactivates the health probe functionality in pod. Pod remains in ready state(1/1). This will avoid pod restarts due to health probes like liveness, readiness probe failure. This is the temporary step and not recommended to use in usual case.
-

- **Activate**
  This option activates the health probe functionality that has been deactivated earlier using the **deactivate** option.

You can manually disable or enable the probes if required. For example, if for any reason you need to exec into the pod and restart the NetBackup services, the health probes should be disabled before restarting the services, and then they should be enabled again after successfully restarting the NetBackup services. If you do not disable the health probes during this process, the pod may restart due to the failed health probes.

Note: It is recommended to disable the health probes only temporarily for troubleshooting purposes. When the probes are disabled, the web UI is not accessible in case of the primary server pod, and the media server pods cannot be scaled up. Then the health probes must be enabled again to successfully run NetBackup.

To disable or enable the health probes

1. Execute the following command in the Primary or media server pod as required:

   ```
   kubectl exec -it -n <namespace> <primary/media-server-pod-name>
   -- /bin/bash
   ```

2. To disable the probes, run the `/opt/veritas/vxapp-manage/nbu-health disable` command. Then the pod goes into the **not ready** (0/1) state.

3. To enable the probes, run the `"/opt/veritas/vxapp-manage/nbu-health enable"` command. Then the pod will be back into the **ready** (1/1) state.

   You can check pod events in case of probe failures to get more details using the `kubectl describe <primary/media-pod-name> -n <namesapce>` command.

Telemetry reporting

Telemetry reporting entries for the NetBackup deployment on AKS are indicated with the **AKS based deployments** text.

- By default, the telemetry data is saved at the `/var/veritas/nbtelemetry/` location. The default location will not persisted during the pod restarts.
- If you want to save telemetry data to persisted location, execute the `kubectl exec -it -n <namespace> <primary/media_server_pod_name> - /bin/bash` command in the pod using the and execute telemetry command using

`/usr/openv/netbackup/bin/nbtelemetry` with `--dataset-path=DESIRED_PATH` option.

About NetBackup operator logs

Note the following about the NetBackup operator logs.

- NetBackup operator logs can be checked using the operator pod logs using the `kubectl logs <Netbackup-operator-pod-name> -c netbackup-operator -n <netbackup-opertaor-namespace>` command.
- NetBackup operator provides different log levels that can be changed before deployment of NetBackup operator.
  The following log levels are provided:
  - -1 - Debug
  - 0 - Info
  - 1 - Warn
  - 2 - Error

  By default, the log level is 0.
  It is recommended to use 0, 1, or 2 log level depending on your requirement.
  Before you deploy NetBackup operator, you can change the log levels using `operator_patch.yaml`.
  After deployment if user changes operator log level, to reflect it, user has to perform the following steps:
  - Apply the operator changes using the `kubectl apply -k <operator-folder>` command.
  - Restart the operator pod. Delete the pod using the `kubectl delete pod/<netbackup-opertaor-pod-name> -n <namespace>` command.
    Kubernetes will recreate the NetBackup operator pod again after deletion.
- Config-Checker jobs that run before deployment of primary server and media server creates the pod. The logs for config checker executions can be checked using the `kubectl logs <configchecker-pod-name> -n <netbackup-operator-namespace>` command.
- Installation logs of NetBackup primary server and media server can be retrieved using any of the following methods:
  - Run the `kubectl logs <primaryServer/MediaServer-Pod-Name> -n <primaryServer/mediaServer namespace>` command.

- Execute the following command in the primary server/media server pod and check the `/mnt/nblogs/setup-server.log` file:

  ```
  kubectl exec -it -n <PrimaryServer/MediaServer-namespace>
  <primaryServer/MediaServer-Pod-Name> -- bash
  ```

Expanding storage volumes

You can update storage capacity of already created persistent volume claim for primary server and media server. Expanding storage volume for particular replica of respective CR object is not supported. In case of media server user needs to update volumes for all the replicas of particular media server object.

To expand storage volumes

1. Edit the environment custom resource using the `kubectl edit Environment <environmentCR_name> -n <namespace>` command.
2. To pause the reconciler of the particular custom resource, change the **Paused** value to **true** in the primaryServer or mediaServer section and save the changes. In case of multiple media server objects change Paused value to true for respective media server object only.
3. Edit StatefulSet of primary server or particular media server object using the`kubectl edit <statfulset name> -n <namespace>` command, change replica count to 0 and wait for all pods to terminate for the particular CR object.
4. Update all the persistent volume claim which expects capacity resize with the `kubectl edit pvc <pvcName> -n <namespace>` command. In case of particular media server object, resize respective PVC with expected storage capacity for all its replicas.
5. Update the respective custom resource section using the `kubectl edit Environment <environmentCR_name> -n <namespace>` command with updated storage capacity for respective volume and change **Paused = false**. Save updated custom resource.

   To update the storage details for respective volume, add storage section with specific volume and its capacity in respective primaryServer or mediaServer section in environment CR.

   Earlier terminated pod and StatefulSet must get recreated and running successfully. Pod should get linked to respective persistent volume claim and data must have been persisted.

6 Run the `kubectl get pvc -n <namespace>` command and check for **capacity** column in result to check the persistent volume claim storage capacity is expanded.

7 (Optional) Update the log retention configuration for NetBackup depending on the updated storage capacity.

For more information, refer to the `NetBackup™ Administrator's Guide, Volume I`

Allocating static PV for Primary and Media pods

When you want to use a disk with specific performance parameters, you can statically create the PV and PVC. You must allocate static PV and PVC before deploying the NetBackup server for the first time.

To allocate static PV for Primary and Media pods

1 Create storage class in cluster as per recommendations.

See "How does the Config-Checker utility work" on page 45. for storage class recommendation.

This newly created storage class name is used while creating PV and PVC's and should be mention for Catalog, Log, Data volume in the environment CR in primaryServer and mediaServer section at the time of deployment.

Refer to the following link to create storage class:

https://docs.microsoft.com/en-us/azure/aks/azure-disk-csi#create-a-custom-storage-class

For example,

```
kind: StorageClass

apiVersion: storage.k8s.io/v1

metadata:

  name: managed-premium-retain

provisioner: disk.csi.azure.com

reclaimPolicy: Retain

allowVolumeExpansion: true

volumeBindingMode: Immediate

parameters:

  storageaccounttype: Premium_LRS

  kind: Managed
```

2 Calculate number of disks required.

The following persistent volumes are required by Primary and Media pods:

- Catalog and Log volume disk for primary server.
- Data and Log volume disk per replica of media server.

Use the following format to form PVC names.

For primary server:

- catalog-<resourceNamePrefix_of_primary>-primary-0
- logs-<resourceNamePrefix_of_primary>-primary-0

For media server

- data-<resourceNamePrefix_of_media>-media-<media server replica number. Count starts from 0>
- logs-<resourceNamePrefix_of_media>-media-<media server replica number. Count starts from 0>

Example 1	If user wants to deploy a primary server and media server with replica count 3. Names of the Primary and Media PVC assuming resourceNamePrefix_of_primary is **testprimary** and resourceNamePrefix_of_media is **testmedia**.	For this scenario, you must create total 8 disks, 8 PV and 8 PVCs. 2 disks, 2 PV and 2 PVCs for primary server. 6 disks, 6 PV and 6 PVCs for media server. Following will be the names for primary server volumes For Catalog: ■ catalog-testprimary-primary-0 For logs: ■ logs-testprimary-primary-0 Following will be the names for media server volumes For data: ■ data-testmedia-media-0 ■ data-testmedia-media-1 ■ data-testmedia-media-10 ■ data-testmedia-media-2 For log: ■ logs-testmedia-media-0 ■ logs-testmedia-media-1 ■ logs-testmedia-media-2

Example 2	If user wants to deploy a primary server and media server with replica count 5 Names of the Primary and Media PVC assuming resourceNamePrefix_of_primary is **testprimary** and resourceNamePrefix_of_media is **testmedia**.	For this scenario, you must create 12 disks, 12 PV and 12 PVCs 2 disks, 2 PV and 2 PVCs for primary server. 10 disks, 10 PV and 10 PVCs for media server. Following will be the names for primary server volumes For Catalog: ■ catalog-testprimary-primary-0 For logs: ■ logs-testprimary-primary-0 Following will be the names for media server volumes For data: ■ data-testmedia-media-0 ■ data-testmedia-media-1 ■ data-testmedia-media-2 ■ data-testmedia-media-3 ■ data-testmedia-media-4 For log: ■ logs-testmedia-media-0 ■ logs-testmedia-media-1 ■ logs-testmedia-media-2 ■ logs-testmedia-media-3 ■ logs-testmedia-media-4

3 Create required number of Azure disks and save the ID of newly created disk.

Azure Disk - Static

4 Create PVs for each disk and link the PVCs to respective PVs.

To create the PVs, specify the created storage class and diskURI (ID of the disk received in step 3). The PV must be created using the **claimRef** field and provide PVC name for its corresponding namespace.

For example, if you are creating PV for catalog volume, storage required is 128GB, diskName is **primary_catalog_pv** and namespace is **test**. PVC named **catalog-testprimary-primary-0** is linked to this PV when PVC is created in the namespace test.

```
apiVersion: v1

               kind: PersistentVolume

               metadata:

                 name: catalog

               spec:

                 capacity:

                              storage: 128Gi

                 accessModes:

                              - ReadWriteOnce

                 azureDisk:

                              kind: Managed

                              diskName: primary_catalog_pv

                              diskURI:  /subscriptions/
3247febe-4e28-467d-a65c-10ca69bcd74b/
resourcegroups/MC_NBU-k8s-network_xxxxxx_eastus/providers
/Microsoft.Compute/disks/deepak_s_pv

                 claimRef:

                              apiVersion: v1

                              kind: PersistentVolumeClaim
                              name: catalog-testprimary-primary-0

                              namespace: test
```

5. Create PVC with correct PVC name (step 2), storage class and storage.

   For example,

```
apiVersion: v1

               kind: PersistentVolumeClaim

               metadata:

                 name: catalog-testprimary-primary-0

                 namespace: test

               spec:

                 storageClassName: "managed-premium-retain"

                 accessModes:

                              - ReadWriteOnce

                 resources:

                              requests:

                                storage: 128Gi
```

6. Deploy the Operator.
7. Use previously created storage class names for the volumes in primary section and mediaServers section in environment CR spec and deploy environment CR.

Chapter 7

Monitoring MSDP Scaleout

This chapter includes the following topics:

About MSDP Scaleout status and events

The MSDP Scaleout CR status includes the readiness state, the storage space utilization (via PersistentVolumeClaim) of each Controller, MDS, and Engine pod.

In the initial configuration of MSDP Scaleout, the readiness state of each pod changes from "false" to "true" in the first few minutes. When the state of all the pods changes to "true", it indicates MSDP Scaleout is ready for use.

You can check the storage space utilization routinely to plan MSDP Scaleout autoscaling before the storage space runs out.

To check the MSDP Scaleout status and events

1. Check the status and the events under the namespace for MSDP Scaleout.

   ```
   kubectl -n <sample-namespace> describe msdpscaleout
   <sample-cr-name>
   ```

2. Check the MSDP Scaleout events.

   ```
   kubectl -n <sample-namespace> get events
   [--sort-by='{.lastTimestamp}']
   ```

3. Check the storage space utilization.

   ```
   kubectl -n <sample-namespace> get msdpscaleout <sample-cr-name>
   -o json
   ```

Example of the of the status format:

```
kubectl -n sample-cr-namespace get msdpscaleout sample-cr -o json
```

```
{
  "controllers": [
    {
      "apiVersions": [
        "1.0"
      ],
      "name": "msdp-aks-demo-uss-controller",
      "nodeName": "aks-nodepool1-25250377-vmss000002",
      "productVersion": "15.1-0159",
      "pvc": [
        {
          "pvcName": "msdp-aks-demo-uss-controller-log",
          "stats": {
            "availableBytes": "10125.98Mi",
            "capacityBytes": "10230.00Mi",
            "percentageUsed": "1.02%",
            "usedBytes": "104.02Mi"
          }
        }
      ],
      "ready": "True"
    }
  ],
  "engines": [
    {
      "ip": "x.x.x.x",
      "name": "msdppods1.westus2.cloudapp.azure.com",
```

```
  "nodeName": "aks-nodepool1-25250377-vmss000003",
  "pvc": [
    {
      "pvcName": "msdppods1.westus2.cloudapp.azure.com-catalog",
      "stats": {
        "availableBytes": "20293.80Mi",
        "capacityBytes": "20470.00Mi",
        "percentageUsed": "0.86%",
        "usedBytes": "176.20Mi"
      }
    },
    {
      "pvcName": "msdppods1.westus2.cloudapp.azure.com-data-0",
      "stats": {
        "availableBytes": "30457.65Mi",
        "capacityBytes": "30705.00Mi",
        "percentageUsed": "0.81%",
        "usedBytes": "247.35Mi"
      }
    }
  ],
  "ready": "True"
},
......
```

Monitoring with Azure Container insights

You can use Azure Container insights to collect Prometheus metrics to monitor pods in MSDP Scaleout.

To configure Azure Container insights

1. Enable Azure Container insights.

   See Azure documentation.

2. Download the template ConfigMap YAML file and save it as `container-azm-ms-agentconfig.yaml`.

3 Add the YAML file with the following sample configuration:

```
prometheus-data-collection-settings: |-

[prometheus_data_collection_settings.cluster]
interval = "1m"

fieldpass = ["msdpoperator_reconcile_total",
             "msdpoperator_reconcile_failed",
             "msdpoperator_operator_run",
             "msdpoperator_diskFreeLess5GBEngines_total",
             "msdpoperator_diskFreeMiBytesInEngine",
             "msdpoperator_diskFreeLess10GBClusters_total",
             "msdpoperator_totalDiskFreePercentInCluster",
             "msdpoperator_diskFreePercentInEngine",
             "msdpoperator_pvcFreePercentInCluster",
             "msdpoperator_unhealthyEngines_total",
             "msdpoperator_createdPods_total"]

monitor_kubernetes_pods = true

# Add the namespace of MSDP operator in the follow list.
It's "msdp-operator-system" by default.

monitor_kubernetes_pods_namespaces =
["msdp-operator-system"]
```

Table 7-1 lists the Prometheus metrics that MSDP Scaleout supports.

4 Apply the ConfigMap.

```
kubectl apply -f container-azm-ms-agentconfig.yaml
```

The configuration change takes a few minutes and all omsagent pods in the cluster restart.

The default namespace of prometheus metrics is **prometheus**.

5 Add alert rules for the integrated metrics.

Add related log query, add new alert rule for the selected query, and alert group/action for it.

For example,

If the free space size of the MSDP Scaleout engines is lower than 1 GB in past 5 minutes, alert the users.

Log query:

```
InsightsMetrics

| where Name == "msdpoperator_diskFreeMiBytesInEngine"
| where Namespace == "prometheus"
| where TimeGenerated > ago(5m)
| where Val <= 1000000
| where Val > 0
```

If multiple MSDP Scaleouts are deployed in the same AKS cluster, use the filter to search the results. For example, search the MSDP engines with the free space size lower than 1GB in the namespace **sample-cr-namespace**

Log query:

```
InsightsMetrics
| where Name == "msdpoperator_diskFreeMiBytesInEngine"
| where Namespace == "prometheus"
| where TimeGenerated > ago(5m)
| where Val <= 1000000
| where Val > 0
| extend Tags = parse_json(Tags)
| where Tags.msdpscalout_ns == "sample-cr-namespace"
```

MSDP Scaleout supports the following Prometheus metrics:

Table 7-1 Supported Prometheus metrics list in MSDP Scaleout

Metrics	Type	Filters	Description
msdpoperator_reconcile_total	Counter	N/A	The total of the reconcile loops msdp-operator run.
msdpoperator_reconcile_failed	Counter	N/A	The total of the reconcile loops msdp-operator failed to run.
msdpoperator_operator_run	Counter	N/A	The total of the running operator.
msdpoperator_diskFreeLess 5GBEngines_total	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The checked number of the engines which have free spaces lower than 5GB.
msdpoperator_diskFreeMi BytesInEngine	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The free space of current engine in MiBytes.
msdpoperator_diskFreeLess 10GBClusters_total	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The checked number of the msdpscaleout apps which have free spaces lower than 10GB.
msdpoperator_totalDiskFree PercentInCluster	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The percent of the msdpscaleout apps that have free spaces. For example, 0.95 means 95%
msdpoperator_diskFree PercentInEngine	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The percent of the current engines, which have free spaces.
msdpoperator_pvcFree PercentInCluster	Gauge	InsightsMetrics.Tags.msdpscalout_ns, InsightsMetrics.Tags.component	The percent of the used PVC, which have free spaces.
msdpoperator_unhealthy Engines_total	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The total of unhealthy engines.
msdpoperator_createdPods _total	Gauge	InsightsMetrics.Tags.msdpscalout_ns, InsightsMetrics.Tags.component	The total of created msdpscaleout pods.

The Kubernetes resources for MSDP Scaleout and MSDP operator

Do not change or delete the Kubernetes resources that MSDP deployment has created.

- Run the following command to find all the namespaced resources:

```
kubectl api-resources --verbs=list --namespaced=true -o name |
xargs -n 1 -i bash -c 'if ! echo {} |grep -q events; then kubectl
get --show-kind --show-labels --ignore-not-found -n <cr or operator
namespace> {}; fi'
```

- Run the following command to find commonly used namespace resources:

  ```
  kubectl get pod,svc,deploy,rs,pvc -n <cr or operator namespace>
  -o wide
  ```

- Run the following command to find the Kubernetes cluster level resources that belong to the CR:

  ```
  kubectl api-resources --verbs=list --namespaced=false -o name |
  xargs -n 1 -i bash -c 'kubectl get --show-kind --show-labels
  --ignore-not-found {} |grep [msdp-operator|<cr-name>]'
  ```

Managing the Load Balancer service

This chapter includes the following topics:

About the Load Balancer service

Key features of the Load Balancer service:

- Load balancer services are created in primary server and media server deployment that lets you access the NetBackup application from public domains.
- In primary server or media server CR spec, networkLoadBalancer section is used for handling the IP address and DNS name allocation for load balancer services. This section combines to sub fields **type**, **annotations**, and **ipList** whereas these fields are optional. If **ipList** is provided in CR spec, IP address count must match the replica count in case of media server CR whereas in case of primary server CR, only one IP address needs to be mentioned.
- In CR yaml, networkLoadBalancer is an optional field. If not defined in CR yaml, by default value of type is **Private** and services are added with annotations `service.beta.kubernetes.io/azure-load-balancer-internal: "true"`. In this case, by default internal load balancer is selected for deployment.
- If networkLoadBalancer section is not defined, by default internal load balancer with dynamic IP address allocation are done. In this case, DNS names for the services can be obtained from **HostName** in CR status using the `kubectl describe <CR name> -n <namespace>` command.

- Whenever, **HostName** in CR status is not in FQDN format, you must add entry of hostname and its corresponding IP address in `/etc/host` to access the primary server and its corresponding IP address in hosts file of computer accessing the primary server. Hosts file is present at the following location:
    - For Linux: `/etc/hosts`
    - For Windows: `c:\Windows\System32\Drivers\etc\hosts`
- In case of media server, FQDN per media server replica is generated using **resourcePrefix** mentioned in media server CR and listed under status attributes **media server-name** of the media server CR.

- In this deployment, it is recommended to use internal load balancer using type as **Private** with static IP allocation and DNS name allocation.
  For details about internal load balancer, see Microsoft documentation.
  However, if type is **public**, then external load balancer is used and for more details to create and use public loadbalancer, see Microsoft documentation.
- The networkLoadBalancer section can be used to provide static IP address and DNS name allocation to the loadbalancer services. For more information to create and use static loadbalancer, see Microsoft documentation.
  Static IP addresses and FQDN if used must be created before being used. Refer below sections for different allowed scenarios.
    - Case 1: Internal load balancer with static IP address allocation
        - Example: In Primary section in environment CR

          ```
          networkLoadBalancer:
                ipList:
                  - ipAddr: 10.123.45.123
          ```

          In media server section in environment CR

          ```
          networkLoadBalancer:
             ipList:
              - ipAddr: 10.123.45.124
              - ipAddr: 10.123.45.125
          ```

          In this case, number of IP addresses for primary server should be one, and for media server, the number of IP addresses should match with the replica count mentioned in CR spec. Dynamically created FQDN mentioned in CR status attribute is used directly as DNS name for primary/media server services.
        - Example: In primary CR

In this case, the number of IP addresses for primary server should be one, and for media server, it should match with the replica count mentioned in CR spec. IP address and DNS name mentioned in CR spec is used as DNS name for primary/media server services.

```
networkLoadBalancer:
      ipList:
        - ipAddr: 10.123.45.123
          fqdn: abc.eastus.cloudapp.azure.com
```

In media server section in environment CR

```
networkLoadBalancer:
      ipList:
        - fqdn: xyz.eastus.cloudapp.azure.com
          ipAddr: 10.123.45.124
        - fqdn: pqr.eastus.cloudapp.azure.com
          ipAddr: 10.123.45.125
```

- Case 2: Internal load balancer and dynamic IP address allocation
  - Example: In primary/media CR
    In this case, IP address and DNS name are allocated dynamically and internal load balancer is used. User needs to add entry of Hostname (FQDN) mentioned in CR status attribute and IP address allocated to load balancer service in `/etc/hosts` location on Linux machine. While `c:\Windows\System32\Drivers\etc\hosts` location on Windows computer to access primary server webUI.
- Case 3: Internal load balancer for different subnet with dynamic IP
  In this case, IP addresses for load balancer service are allocated dynamically. The subnet mentioned in annotations is bound to internal load balancer service.
  - Example: In primary CR

```
networkLoadBalancer:
annotations:
- service.beta.kubernetes.io/
azure-load-balancer-internal-subnet: "apps-subnet"
```

Media CR

```
networkLoadBalancer:
annotations:
```

```
- service.beta.kubernetes.io/
azure-load-balancer-internal-subnet: "apps-subnet"
```

- Case 4: Internal load balancer for different subnet with static IP
  In this case, load balancer service gets assigned with the static IP addresses mentioned in the `ipList`, DNS name is generated dynamically, and gets bound to the subnet given in the annotations.
  - Example: In primary section in environment CR,

    ```
    networkLoadBalancer:
      annotations:
      - service.beta.kubernetes.io/azure-load-balancer-
    internal-subnet: apps-subnet
      ipList:
      - ipAddr: 10.123.45.123
    ```

    Media server section in environment CR

    ```
    networkLoadBalancer:
      annotations:
      - service.beta.kubernetes.io/azure-load-balancer-
    internal-subnet: apps-subnet
      ipList:
      - ipAddr: 10.123.45.125
      - ipAddr: 10.123.45.124
    ```

- Case 5: Pre-allocation of static IP address and FQDN from resource group
  In this case, it is required to provide the network resource group in annotations. This resource group is the resource group of load balancer public IPs that are in the same resource group as the cluster infrastructure (node resource group). This static FQDN and IP address must be valid in case of pod failure or upgrades scenarios as well.
  In case user wants to use public load balancer, add **type: Public** in networkLoadBalancer section in primary and media server section in environment CR.
  - Example: In primary CR,

    ```
    networkLoadBalancer:
      type: Public
      annotations:
      - service.beta.kubernetes.io/azure-load-balancer-
    resource-group:<name of network resource-group>
      ipList:
    ```

```
  - fqdn: primary.eastus.cloudapp.azure.com
    ipAddr: 40.123.45.123
```

Media server section in environment CR -

```
networkLoadBalancer:
  annotations:
  - service.beta.kubernetes.io/azure-load-balancer-
resource-group: ""<name of network resource-group>""
  ipList:
  - fqdn: media-1.eastus.cloudapp.azure.com
    ipAddr: 40.123.45.123
  - fqdn: media-2.eastus.cloudapp.azure.com
    ipAddr: 40.123.45.124
```

Preferred annotations

Table 8-1 Preferred annotations

Annotations	Value	Description
service.beta.kubernetes.io/ azure-load-balancer- internal	true or false	Specify whether the load balancer should be internal. Added by default when type is selected as **Private** in load balancer service annotations.
service.beta.kubernetes.io/ azure-load-balancer- internal-subnet	Name of the subnet	Specify which subnet the internal load balancer should be bound to.
service.beta.kubernetes.io/ azure-load-balancer -resource-group	Name of the resource group	Specify the resource group of load balancer public IPs that are not in the same resource group as the cluster infrastructure (node resource group).

Default ports used in the Load Balancer service

- Primary server:
  - 1556
    Used as bidirectional port. Primary server to/from media servers and primary server to/from client require this TCP port for communication.
  - 8443
    Used to inbound to java nbwmc on the primary server.

- 443

  Used to inbound to vnet proxy tunnel on the primary server. Also, this is used Nutanix workload, communication from primary server to the deduplication media server.
- 13781

  The MQBroker is listening on TCP port 13781. NetBackup client hosts - typically located behind a NAT gateway - be able to connect to the message queue broker (MQBroker) on the primary server.
- 13782

  Used by primary server for bpcd process.
- Port 22

  Used by NetBackup IT Analytics data collector for data collection.

- Media server:
  - 1556

    Used as bidirectional port. Primary server to/from media servers and primary server to/from client require this TCP port for communication.
  - 13782

    Used by media server for bpcd process.

Notes for Load Balancer service

Note the following points:

- After deployment of primary server or media server, updating the DNS name, IP address and subnet through CR is not allowed.
- If mistakenly user has added wrong values:
  - User wants to update IP address and subnet, you must delete the CR and update the CR yaml and reapply it.
  - User wants to update the DNS name, you must delete the respective CR and delete the respective PVC and PV as well.

    Note: Be caution while performing this step, this may lead to data loss.
- Before using the DNS and its respective IP address in CR yaml, you can verify the IP address and its DNS resolution using nslookup.
- In case of media server scaleout, ensure that the number of IP addresses mentioned in IPList in networkLoadBalancer section matches the replica count.

- If nslookup is done for loadbalancer IP inside the container, it returns the DNS in the form of **<svc name>.<namespace_name>.svc.cluster.local**. This is Kubernetes behavior. Outside the pod, the loadbalancer service IP address is resolved to the configured DNS. The `nbbptestconnection` command inside the pods can provide a mismatch in DNS names, which can be ignored.

Opening the ports from the Load Balancer service

In this deployment, most of the required ports are already opened from the NetBackup primary and media server load balancer services by default.

- If you want to use a specific workload and that needs specific ports, you must add those ports in the port specification of the load balancer service.
- In case of media server, you must add custom ports in the load balancer service of all the replicas. In case of scaling up the media server, user needs to explicitly add newly added custom ports in respective newly created load balancer services.
- In case custom ports are added in the load balancer service and the same load balancer service is deleted or created again, you must add respective custom ports again in the load balancer service specification.

For all three scenarios, perform the steps given in this section.

To open the ports from the Load Balancer service

1. Run the `kubectl get service -n <namespace>` command.

   This command lists all the services available in given namespace.

2. Edit the required primary or media load balancer service using `kubectl edit service <service-name> -n <namespace>` command.

   For example:

   - For primary server load balancer service:
     - Service name starts with **resourcePrefixName** of primary server like **<resourcePrefixName>-primary**. Edit the service with the `kubectl edit service <resourcePrefixName>-primary -n <namespace>` command.
   - For media server load balancer service:
     - Each replica of media server has its own load balancer service with name **<resourcePrefixName>-media-<ordinal number>**. For example, replica 2 of media server has a load balancer service with name **<resourcePrefixName>-media-1**.

- You must modify service for specific replica with the `kubectl edit service <resourcePrefixName>-media-<replica-ordinal number> -n <namespace>` command.

3. Add entry for new port in ports array in specification field of the service. For example, if user want to add 111 port, then add the following entry in ports array in specification field.

```
name: custom-111

    port: 111

    protocol: TCP

    targetPort: 111
```

4. Save the changes.

   The service is updated and the new port is listed in ports list of the respective service when you run the `kubectl get service -n <namespace>` command.

Performing catalog backup and recovery

This chapter includes the following topics:

Backing up a catalog

You can back up a catalog.

To back up a catalog

1. Execute the following command in the primary server pod:

   ```
   kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash
   ```

2. Create a directory **DRPackages** at persisted location using `mkdir /mnt/nblogs/DRPackages`.

3. Change ownership of **DRPackages** folder to service user using `chown nbsvcusr:nbsvcusr /mnt/nblogs/DRPackages`.

4. Set the passphrase to be used at time of catalog recovery.

   - Open NetBackup Administrator Console (Java UI).
   - Navigate to **Security Management > Global Security Setting > Disaster Recovery**.
   - In **Encryption for Disaster Recovery** section, add the passphrase, confirm passphrase, and save it.

5 Add respective external media server entry in host properties through **NetBackup Management > Host properties > Master Server > Add Additional server**.

Note: It is recommended to use an external media server for catalog backup and recovery.

---

6 Restart the NetBackup services in primary and external media server.

- Execute the following command in the primary server pod:
  `kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash`
- Deactivate NetBackup health probes using the `/opt/veritas/vxapp-manage/nbu-health deactivate` command.
- Run the `/usr/openv/netbackup/bin/bp.kill_all` command. After stopping all services restart the services using the `/usr/openv/netbackup/bin/bp.start_all` command.
- Activate NetBackup health probes using the `/opt/veritas/vxapp-manage/nbu-health activate` command.
- Run the `/usr/openv/netbackup/bin/bp.kill_all` command. After stopping all services restart the services using the `/usr/openv/netbackup/bin/bp.start_all` command on the external media server.

7 Configure storage unit on earlier added external media server.

For more information, refer to the *NetBackup™ Administrator's Guide,Volume I*

Note: It is recommended to use AdvancedDisk or BasicDisk storage unit.

---

8 Configure NetBackup catalog backup policy.

Add package path as `/mnt/nblogs/DRPackages` while configuring the catalog backup policy.

9 Run the catalog backup job.

Restoring a catalog

You can restore a catalog.

To restore a catalog

1. Copy DRPackages files (packages) located at `/mnt/nblogs/DRPackages/` from the pod to the host machine from where Azure cluster is accessed.

   Run the `kubectl cp <primary-pod-namespace>/<primary-pod-name>:/mnt/nblogs/DRPackages <Path_where_to_copy_on_host_machine>` command.

2. Preserve the data of `/mnt/nbdata` and `/mnt/nblogs` on host machine by creating tar and copying it using the `kubectl cp <primary-pod-namespace>/<primary-pod-name>:<tar_file_name> <path_on_host_machine_where_to_preserve_the_data>` command.

3. Change CR spec from **paused** to **true** in primary, mediaServers, and msdpScaleouts sections in `environment.yaml` and re-apply yaml using the `kubectl apply -f environment.yaml -n <namespace>` command.

4. Change replica count to 0 in primary server's statefulset using the `kubectl edit statefulset <primary-server-statefulset-name> -n <namespace>` command.

5. Get names of PV attached to primary server PVC (catalog,log) using the `kubectl get pvc -n <namespace> -o wide` command.

6. Delete primary server PVC (catalog, log) using the `kubectl delete pvc <pvc-name> -n <namespace>` command.

7. Delete the PV linked to primary server PVC using the `kubectl delete pv <pv-name>` command.

8. Change CR spec paused to false in primary server section in `environment.yaml` and reapply yaml with the `kubectl apply -f environment.yaml -n <namespace>` command.

9. After the primary server pod is in ready state, change CR spec from **paused** to **true** in primary server section in `environment.yaml` and reapply yaml with the `kubectl apply -f environment.yaml -n <namespace>` command.

10. Execute the `kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash` command in the primary server pod.

    - Increase the debug logs level on primary server.
    - Create a directory `DRPackages` at persisted location using `mkdir /mnt/nblogs/DRPackages`.
    - Change ownership of the `DRPackages` folder to service user using the `chown nbsvcusr:nbsvcusr /mnt/nblogs/DRPackages` command.

11 Copy earlier copied DR files to primary pod at `/mnt/nblogs/DRPackages` using the `kubectl cp <Path_of_DRPackages_on_host_machine> <primary-pod-namespace>/<primary-pod-name>:/mnt/nblogs/DRPackages` command.

12 Execute the following steps in the primary server pod.

- Change ownership of files in `/mnt/nblogs/DRPackages` using the `chown nbsvcusr:nbsvcusr <file-name>` command.
- Deactivate NetBackup health probes using the `/opt/veritas/vxapp-manage/nbu-health deactivate` command.
- Stop the NetBackup services using `/usr/openv/netbackup/bin/bp.kill_all`.
- Execute the `nbhostidentity -import -infile /mnt/nblogs/DRPackages/<filename>.drpkg` command.
- Restart all the NetBackup services using `/usr/openv/netbackup/bin/bp.start_all`.

13 Verify security settings are back.

14 Add respective media server entry in host properties using NetBackup Administration Console.

- Navigate to **NetBackup Management > Host properties > Master Server > Add Additional server** and add media server.

15 Restart the NetBackup services in primary server pod and external media server.

- Execute the following command in the primary server pod:

  ```
  kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash
  ```

- Run the `/usr/openv/netbackup/bin/bp.kill_all` command. After stopping all services restart the same using the `/usr/openv/netbackup/bin/bp.start_all` command.
- Run the `/usr/openv/netbackup/bin/bp.kill_all` command. After stopping all services restart the services using the `/usr/openv/netbackup/bin/bp.start_all` command on the external media server.

16 Configure a storage unit on external media server that is used during catalog backup.

17 Perform catalog recovery from NetBackup Administration Console.

For more information, refer to the NetBackup Troubleshooting Guide.

18 Execute the `kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash` command in the primary server pod.

- Stop the NetBackup services using the `/usr/openv/netbackup/bin/bp.kill_all` command.
- Start NetBackup services using the `/usr/openv/netbackup/bin/bp.start_all` command.
- Activate NetBackup health probes using the `/opt/veritas/vxapp-manage/nbu-health activate` command.

19 Change CR spec from **paused** to **false** in primary, mediaServers, and msdpScaleouts sections in `environment.yaml` and re-apply yaml using the `kubectl apply -f environment.yaml -n <namespace>` command.

20 To configure NetBackup IT Analytics refer to the following topic.

See "Configuring NetBackup IT Analytics for NetBackup deployment" on page 57.

Managing MSDP Scaleout

This chapter includes the following topics:

Adding MSDP engines

You can add new MSDP engines by updating the CR. You can add maximum 16 MSDP engines.

Prerequisites:

- Allocate new static IP/FQDN pairs in the same node resource group.
- The node number must not be less than the MSDP Scaleout size that you plan to change.
- CR YAML file of MSDP Scaleout

To add the MSDP engines by updating the CR YAML file

1. Open the CR YAML file to edit.
2. Append the new IP/FQDN pairs in the **spec.serviceIPFQDNs** field.

3. Update the **spec.size** field to increase the cluster size accordingly.
4. Apply new CR YAML to update the CR in the Kubernetes environment.

   ```
   kubectl apply -f <your-cr-yaml>
   ```

To add the MSDP engines using the kubectl command directly

- Run the following command to append the IP/FQDN pairs in the **spec.serviceIPFQDNs** field and increase the cluster size in **spec.size** field.

  ```
  kubectl -n <sample-namespace> edit msdpscaleout <your-cr-name>
  [-o json | yaml]
  ```

The MSDP Scaleout services are not interrupted when MSDP engines are added.

Adding data volumes

You can add the data volumes by updating the CR.

To add the data volumes by updating the CR YAML file

1. Open the CR YAML file to edit.
2. Append the new data volume specifications in the **spec.dataVolumes** field.
3. Apply new CR YAML to update the CR in the Kubernetes environment.

   ```
   kubectl apply -f <your-cr-yaml>
   ```

To add the MSDP engine using the kubectl command directly

- Run the following command to append new data volume specifications in the **spec.dataVolumes** field.

  ```
  kubectl -n <sample-namespace> edit msdpscaleout <your-cr-name>
  [-o json | yaml]
  ```

In the MSDP engine pod, the first data volume is mounted on `/msdp/data/dp1/pdvol`. Nth data volume is mounted on `/msdp/data/dp1/${N-1}pdvol`. For example, 2nd data volume is mounted on `/msdp/data/dp1/1pdvol`.

Each MSDP engine can support up to 16 data volumes.

It is recommended that you use the same data volume size if you add multiple volumes.

Note: Due to some Kubernetes restrictions, MSDP operator restarts the engine pods for attaching the existing and new volumes, which can cause the short downtime of the services.

Expanding existing data or catalog volumes

You can expand the existing data or catalog volumes by updating the CR.

To expand the data or catalog volumes by updating the CR YAML file

1. Open the CR YAML file to edit.
2. Increase the requested storage size in the **spec.dataVolumes** field or in the **spec.catalogVolume** field.
3. Apply new CR YAML to update the CR in the Kubernetes environment.

   ```
   kubectl apply -f <your-cr-yaml>
   ```

To expand the data or catalog volumes using the kubectl command directly

- ◆ Run the following command to increase the requested storage size in the **spec.dataVolumes** field or in the **spec.catalogVolume** field..

  ```
  kubectl -n <sample-namespace> edit msdpscaleout <your-cr-name>
  [-o json | yaml]
  ```

Sometimes Azure disk CSI driver may not respond the volume expansion request promptly. In this case, the operator retries the request by adding 1 byte to the requested volume size to trigger the volume expansion again. If it is successful, the actual volume capacity could be slightly larger than the requested size.

Due to the limitation of Azure disk CSI storage driver, the engine pods need to be restarted for resizing the existing volumes. This can cause the short downtime of the services.

MSDP Scaleout does not support the following:

- Cannot shrink the volume size.
- Cannot change the existing data volumes other than for storage expansion.
- Cannot expand the log volume size. You can do it manually. See "Manual storage expansion" on page 103.
- Cannot expand the data volume size for MDS pods. You can do it manually. See "Manual storage expansion" on page 103.

Manual storage expansion

You also can manually expand storage size by expanding PVC size.

To expand the data or catalog volumes

1. Open the CR YAML file to edit.
2. Configure `spec.paused: true`.

3 Apply new CR YAML to stop MSDP operator from reconciling and repairing the pods automatically.

```
kubectl apply -f <your-cr-yaml>
```

4 Patch the corresponding PVCs.

```
kubectl patch pvc <pvc-name> --type merge --patch '{"spec":
{"resources": {"requests": {"storage": "<requested-size>"}}}}'
-n <sample-namespace>
```

5 Specify `spec.paused: false` in the CR.

6 Apply new CR YAML to recover MSDP operator to continue to reconcile and repair the pods automatically.

```
kubectl apply -f <your-cr-yaml>
```

Note: If you add new MSDP Engines later, the new Engines will respect the CR specification only. Your manual changes would not be respected by the new Engines.

MSDP Scaleout scaling recommendations

Following are the scaling recommendations for the MSDP Scaleout:

- Allocate the data volumes of the similar sizes for MSDP to have better load balancing performance.
- Each data volume size is more than 4 TB.
- Have multiple data volumes for each engine to gain better throughput.
- Split a bigger backup policy to smaller ones
  In most cases, one backup job goes to one MSDP engine at the same time even if multistream is enabled for the backup policy. If the current MSDP engine, which is taking a backup job hits the high space watermark, the following backup data would be sent to a second MSDP engine. If the backup data is too big for up to 2 MSDP engines to persist, the backup job fails. When more MSDP engines are added, the backup jobs may not be evenly balanced on each MSDP engine at the first a few hours or days. If the situation keeps longer beyond your expectation, consider to re-plan the backup policies, by splitting a bigger backup policy to two smaller ones, to help MSDP Scaleout to balance the new backup jobs more faster.
- After scaling up, the memory and CPU of the existing node pool may not meet the performance requirements anymore. In this case, you can add more memory and CPU by upgrading to the higher instance type to improve the existing node pool performance or create another node pool with higher instance type and

update the node-selector for the CR accordingly. If you create another node pool, the new node-selector does not take effect until you manually delete the pods and deployments from the old node pool, or delete the old node pool directly to have the pods re-scheduled to the new node pool.

- Ensure that each AKS node supports mounting the number of data volumes plus 5 of the data disks.

  For example, if you have 16 data volumes for each engine, then each your AKS node should support mounting at least 21 data disks. The additional 5 data disks are for the potential MDS pod, Controller pod or MSDP operator pod to run on the same node with MSDP engine.

MSDP Cloud backup and disaster recovery

For information about MSDP cloud backup and disaster recovery, see MSDP Cloud section of the *NetBackup Deduplication Guide*

About the reserved storage space

About 1 TB storage space is reserved by default on each MSDP engine for each cloud LSU.

The 1 TB storage space is selected from one of the data volumes of every engine. It requires each engine at least has one data volume, which has more than 1 TB available storage space, when a cloud LSU is to be configured. Otherwise, the configuration of the cloud LSU fails.

Cloud LSU disaster recovery

Scenario 1: MSDP Scaleout and its data is lost and the NetBackup primary server remains unchanged and works well

1 Redeploy MSDP Scaleout on a AKS cluster by using the same CR parameters and NetBackup re-issue token.

2 When MSDP Scaleout is up and running, re-use the cloud LSU on NetBackup primary server.

```
/usr/openv/netbackup/bin/admincmd/nbdevconfig -setconfig
-storage_server <STORAGESERVERNAME> -stype PureDisk -configlist
<configuration file>
```

Credentials, bucket name, and sub bucket name must be the same as the recovered Cloud LSU configuration in the previous MSDP Scaleout deployment.

Configuration file template:

```
V7.5 "operation" "reuse-lsu-cloud" string
V7.5 "lsuName" "LSUNAME" string
V7.5 "lsuCloudUser" "XXX" string
V7.5 "lsuCloudPassword" "XXX" string
V7.5 "lsuCloudAlias" "<STORAGESERVERNAME_LSUNAME>" string
V7.5 "lsuCloudBucketName" "XXX" string
V7.5 "lsuCloudBucketSubName" "XXX" string
V7.5 "lsuKmsServerName" "XXX" string
```

If the LSU cloud alias does not exist, you can use the following command to add it.

```
/usr/openv/netbackup/bin/admincmd/csconfig cldinstance -as -in
<instance-name> -sts <storage-server-name> -lsu_name <lsu-name>
```

3. On the first MSDP Engine of MSDP Scaleout, run the following command for each cloud LSU:

   ```
   sudo -E -u msdpsvc /usr/openv/pdde/pdcr/bin/cacontrol --catalog
   clouddr <LSUNAME>
   ```

4. Restart the MSDP services in the MSDP Scaleout.

   Option 1: Manually delete all the MSDP engine pods.

   ```
   kubectl delete pod <sample-engine-pod> -n <sample-cr-namespace>
   ```

   Option 2: Stop MSDP services in each MSDP engine pod. MSDP service starts automatically.

   ```
   kubectl exec <sample-engine-pod> -n <sample-cr-namespace> -c
   uss-engine -- /usr/openv/pdde/pdconfigure/pdde stop
   ```

Scenario 2: MSDP Scaleout and its data is lost and the NetBackup primary server was destroyed and is re-installed

1 Redeploy MSDP Scaleout on a AKS cluster by using the same CR parameters and new NetBackup token.

2 When MSDP Scaleout is up and running, reuse the cloud LSU on NetBackup primary server.

```
/usr/openv/netbackup/bin/admincmd/nbdevconfig -setconfig
-storage_server <STORAGESERVERNAME> -stype PureDisk -configlist
<configuration file>
```

Credentials, bucket name, and sub bucket name must be the same as the recovered Cloud LSU configuration in previous MSDP Scaleout deployment.

Configuration file template:

```
V7.5 "operation" "reuse-lsu-cloud" string
V7.5 "lsuName" "LSUNAME" string
V7.5 "lsuCloudUser" "XXX" string
V7.5 "lsuCloudPassword" "XXX" string
V7.5 "lsuCloudAlias" "<STORAGESERVERNAME_LSUNAME>" string
V7.5 "lsuCloudBucketName" "XXX" string
V7.5 "lsuCloudBucketSubName" "XXX" string
V7.5 "lsuKmsServerName" "XXX" string
```

If KMS is enabled, setup KMS server and import the KMS keys.

If the LSU cloud alias does not exist, you can use the following command to add it.

```
/usr/openv/netbackup/bin/admincmd/csconfig cldinstance -as -in
<instance-name> -sts <storage-server-name> -lsu_name <lsu-name>
```

3 On the first MSDP Engine of MSDP Scaleout, run the following command for each cloud LSU:

```
sudo -E -u msdpsvc /usr/openv/pdde/pdcr/bin/cacontrol --catalog
clouddr <LSUNAME>
```

4 Restart the MSDP services in the MSDP Scaleout.

Option 1: Manually delete all the MSDP engine pods.

```
kubectl delete <sample-engine-pod> -n <sample-cr-namespace>
```

Option 2: Stop MSDP services in each MSDP engine pod.

```
kubectl exec <sample-engine-pod> -n <sample-cr-namespace> -c
uss-engine -- /usr/openv/pdde/pdconfigure/pdde stop
```

5. Create disk pool for the cloud LSU on NetBackup server.
6. Do two-phase image importing.

   See the *NetBackup Administrator's Guide, Volume I*

For information about other DR scenarios, see *NetBackup Deduplication Guide*.

MSDP multi-domain support

An MSDP storage server is configured in a NetBackup media server. The NetBackup media servers and clients in the NetBackup domain can use this storage server. By default, the NetBackup media servers and clients cannot directly use an MSDP storage server from another NetBackup domain. For example, NetBackup media servers or clients cannot backup data to an MSDP storage server from another NetBackup domain.

To use an MSDP storage server from another NetBackup domain, the MSDP storage server must have multiple MSDP users. Then NetBackup media servers or clients can use the MSDP storage server from another NetBackup domain by using a different MSDP user. Multiple NetBackup domains can use the same MSDP storage server but each NetBackup domain must use a different MSDP user to access that MSDP storage server.

For more information, See *NetBackup Deduplication Guide*.

When you add a new MSDP user, the command `spauser` must be executed in the first MSDP engine of MSDP Scaleout, not on any of the NetBackup servers.

Ensure that you run MSDP commands with non-root user **msdpsvc** after logging into an engine pod.

For example, `sudo -E -u msdpsvc /usr/openv/pdde/pdcr/bin/spauser`

Configuring Auto Image Replication

The backups that are generated in one NetBackup domain can be replicated to storage in one or more target NetBackup domains. This process is referred to as Auto Image Replication (A.I.R.).

You can configure Auto Image Replication in NetBackup, which is using MSDP Scaleout storage servers.

To configure Auto Image Replication

1. Logon to the NetBackup Web UI of both replication source and target domain.
2. Add each other NetBackup's primary server as trusted primary server.

   For more information, see the *NetBackup Web UI Administrator's Guide*.
3. In the replication source domain, get the MSDP_SERVER name from the NetBackup Web UI.

   Navigate to **Storage > Storage configuration > Storage servers**.
4. Add MSDP_SERVER in the primary server of replication target domain. Login to the target primary server and run the following command:

   ```
   echo "MSDP_SERVER = <Source MSDP server name>" >>
   /usr/openv/netbackup/bp.conf
   ```
5. Get the token from the target domain NetBackup Web UI.

   Navigate to **Security > Token**. In the **Create token** window, enter the token name and other required details. Click **Create**.

   For more information, see the *NetBackup Web UI Administrator's Guide*.
6. Add replication targets for the disk pool in replication source domain.

   In the **Disk pools** tab, click on the disk pool link.

   Click **Add** to add the replication target.
7. In the **Add replication targets** window:

   - Select the replication target primary server.
   - Provide the target domain token.
   - Select the target volume.
   - Provide the target storage credentials.

   Click **Add**.

About MSDP Scaleout logging and troubleshooting

- AKS troubleshooting
  See AKS troubleshooting page of *Azure documentation*.
- Logs and core dumps files in MSDP Scaleout
  MSDP Operator, Controller, and MDS pod logs are stored in */log* location of the pods.

- Collect the logs and inspection information
  You can collect the logs and inspection information for MSDP Scaleout for troubleshooting purpose.
  See "Collecting the logs and the inspection information" on page 111.

Collecting the logs and the inspection information

You can collect the logs and inspection information for MSDP Scaleout for troubleshooting purpose.

Run the command `kubectl msdp collect-logs`

For example, `kubectl msdp collect-logs -o <output path> [-n <MSDP operator namespace>] [-c <MSDP applications namespace(s)>]`

Table 10-1 collect-logs command options

Option	Description
`-c`	Comma-separated namespaces of MSDP applications. **Note:** If not specified, it collects MSDP applications of all namespaces.
`-f`	Output format of logs/core files/MSDP history files. Available options: targz: Copy logs/core files/MSDP history files from containers and compress them by tar/gzip. raw: Copy logs/core files/MSDP history files from containers as same format in the containers. Default value: targz
`-n`	Namespace of MSDP operator. Default value: msdp-operator-system
`-o`	Output path of the log file.

About MSDP Scaleout maintenance

This chapter includes the following topics:

Pausing the MSDP Scaleout operator for maintenance

For maintenance purpose, if you want the operator to stop reconciling the resources of one CR but do not affect the resources of the other CRs, you can pause the MSDP Scaleout operator.

To pause the MSDP Scaleout operator

1. Specify `spec.paused: true` in the CR.
2. Run `kubectl apply -f <sample CR YAML>`.

Do not forcibly delete the deployment resource of MSDP Scaleout operator.

Logging in to the pods

You can log in to the pods for the maintenance purpose.

To log in to the pod, run the `kubectl` executable file.

Run MSDP commands with non-root user **msdpsvc** after logging in to an engine pod.

For example, `sudo -E -u msdpsvc <command>`

The MSDP Scaleout services in an engine pods are running with non-root user **msdpsvc**. If you run the MSDP Scaleout services or commands with the root user, MSDP Scaleout may stop working due to file permissions issues.

Reinstalling MSDP Scaleout operator

When you undeploy MSDP Scaleout operator, the MSDP Scaleout CRD is removed from the AKS cluster. It also deletes all the existing MSDP Scaleout on the AKS cluster. The PVC for the operator logs is also deleted. However, the MSDP Scaleout critical data and metadata is not deleted.

To reinstall MSDP Scaleout operator

1. Run the following command to delete the MSDP Scaleout operator:

   ```
   kubectl msdp delete [-k] [-n <sample-operator-namespace>]
   ```

2. Run the following command to redeploy the operator.

   ```
   kubectl msdp init -i <your-acr-url>/msdp-operator:<version> -s
   <storage-class-name> -l agentpool=<nodepool-name> [-n
   <sample-operator-namespace>]
   ```

3. If the reclaim policy of the storage class is **Retain**, run the following command to restart the existing MSDP Scaleout. MSDP Scaleout starts with the existing data/metadata.

   ```
   kubectl apply -f <your-cr-yaml>
   ```

Migrating the MSDP Scaleout to another node pool

You can migrate an existing MSDP Scaleout on another node pool in case of the Kubernetes infrastructure issues.

To migrate the MSDP Scaleout to another node pool

1. Ensure that no job running related to MSDP Scaleout that is going to migrate.
2. Update the node selector value **spec.nodeSelector** to the new node in the CR YAML file.

3. Apply new CR YAML to update the CR in the Kubernetes environment.

   ```
   kubectl apply -f <your-cr-yaml>
   ```

   Note: All affected pods or other Kubernetes workload objects must be restarted for the change to take effect.

4. Delete all the pods and deployments in the namespace of MSDP Scaleout manually.

   ```
   kubectl get pod,deploy -n <sample-namespace>
   ```

   ```
   kubectl delete -n <sample-namespace> <pod1> <pod2> ... <podn>
   <deployment>
   ```

   After you run the commands, the pods are re-scheduled onto the new node pool automatically.

5. Run the following command to change MSDP Scaleout operator to the new node pool:

   ```
   kubectl msdp init -i <your-acr-url>/msdp-operator:<version> -s
   <storage-class-name> -l agentpool=<new-nodepool-name>
   ```

6. If MSDP Scaleout operator pod is not restarted, manually delete the pods in the namespace of MSDP Scaleout operator. After that, the Pods of MSDP Scaleout operator would be re-scheduled onto the new node pool.

   ```
   kubectl get pod -n msdp-operator-system
   ```

   ```
   kubectl delete -n msdp-operator-system pod <MSDP operator pod>
   ```

   After you run the commands, the MSDP Scaleout operator pods are re-scheduled onto the new node pool.

Uninstalling MSDP Scaleout from AKS

This chapter includes the following topics:

Cleaning up MSDP Scaleout

When you uninstall the MSDP Scaleout deployment from AKS, the MSDP engines, MSDP MDS servers, and the data is deleted from the cluster. The data is lost and cannot be recovered.

To clean up MSDP Scaleout from AKS

1 Delete the MSDP Scaleout CR.

```
kubectl delete -f <sample-cr-yaml>
```

When an MSDP Scaleout CR is deleted, the critical MSDP data and metadata is not deleted. You must delete it manually. If you delete the CR without cleaning up the data and metadata, you can re-apply the same CR YAML file to restart MSDP Scaleout again by reusing the existing data.

2 If your storage class is with the **Retain** policy, you must write down the PVs that are associated with the CR PVCs for deletion in the Kubernetes cluster level.

```
kubectl get
pod,svc,deploy,rs,ds,pvc,secrets,certificates,issuers,cm,sa,role,rolebinding
-n <sample-namespace> -o wide
```

```
kubectl get clusterroles,clusterrolebindings,pv -o wide
--show-labels|grep <sample-cr-name>
```

3 Delete all resources under the namespace where MSDP CR is deployed.

```
kubectl delete namespace <namespace>
```

4 If your storage class is with the **Retain** policy, you must delete the Azure disks on Azure portal or using the Azure CLI.

```
az disk delete -g $RESOURCE_GROUP --name $AZURE_DISK --yes
```

See "Deploying MSDP Scaleout " on page 61.

See "Reinstalling MSDP Scaleout operator" on page 113.

Cleaning up the MSDP Scaleout operator

You can delete the MSDP Scaleout operator to remove all related resources about MSDP Scaleout operator. The MSDP Scaleout operator and logs are deleted.

To clean up MSDP Scaleout operator from AKS

1 If your storage class is with **Retain** policy, write down the PVs that are associated with the Operator PVCs for deletion in the Kubernetes cluster level.

```
kubectl get
pod,svc,deploy,rs,ds,pvc,secrets,certificates,issuers,cm,sa,role,rolebinding
-n <sample-operator-namespace> -o wide
```

```
kubectl get clusterroles,clusterrolebindings,pv -o wide
--show-labels
```

2 Delete the MSDP Scaleout operator.

```
kubectl msdp delete [-n <sample-operator-namespace>].
```

- `-k`: Delete all resources of MSDP Scaleout operator except the namespace.
- `-n`: Namespace scope for this request.
  Default value: msdp-operator-system

3 If your storage class is with the **Retain** policy, you must delete the Azure disks on Azure portal or using the Azure CLI.

```
az disk delete -g $RESOURCE_GROUP --name $AZURE_DISK --yes
```

See "Deploying MSDP Scaleout " on page 61.

See "Reinstalling MSDP Scaleout operator" on page 113.

Chapter 13

Troubleshooting

This chapter includes the following topics:

- Resolve an issue related to inconsistency in file ownership
- Resolve an issue related to KMS database
- Resolve an issue related to pulling an image from the container registry
- Resolving an issue related to recovery of data
- Check primary server status
- Pod status field shows as pending
- Ensure that the container is running the patched image
- Getting EEB information from an image, a running container, or persistent data

View the list of operator resources

To view all the operator resources, execute the following command on Kubernetes cluster:

```
$ kubectl get all -n netbackup-operator-system
```

The output should be something like this:

```
            NAME                                  READY    STATUS    RESTARTS
pod/msdp-operator-controller-manager-65d8fd7c4d-whqpm        2/2      Running
pod/netbackup-operator-controller-manager-55d6bf59c8-vltmp   2/2      Running

NAME                         TYPE     CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/msdp-operator-       ClusterIP 10.96.144.99 <none>   8443/TCP   3h6m
controller-manager-
metrics-service

service/msdp-operator-       ClusterIP 10.96.74.75  <none>   443/TCP    3h6m
webhook-service

service/netbackup-           ClusterIP 10.96.104.94 <none>   8443/TCP   93m
operator-controller
-manager-metrics-service

service/netbackup-           ClusterIP 10.96.210.26 <none>   443/TCP    93m
operator-webhook-service

NAME                            READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/msdp-           1/1      1             1            3h6m
```

```
operator-controller-manager

deployment.apps/netbackup   1/1     1            1           93m
-operator-controller-manager

NAME                         DESIRED   CURRENT   READY   AGE
replicaset.apps/msdp-        1         1         1       3h6m
operator-controller-
manager-65d8fd7c4d
replicaset.apps/netbackup-   1         1         1       93m
operator-controller-manager-
55d6bf59c8
```

Verify that both pods display **Running** in the Status column and both deployments display **2/2** in the **Ready** column.

View the list of product resources

To view the list of product resources run the following command:

```
$ kubectl get --namespace <namespace>
all,environments,primaryservers,mediaservers,msdpscaleouts
```

The output should look like the following:

```
NAME                           READY   STATUS    RESTARTS   AGE
pod/dedupe1-uss-controller
-79d554f8cc-598pr              1/1     Running   0          68m
pod/dedupe1-uss-mds-1          1/1     Running   0          75m
pod/dedupe1-uss-mds-2          1/1     Running   0          74m
pod/dedupe1-uss-mds-3          1/1     Running   0          71m
pod/media1-media-0             1/1     Running   0          53m
pod/environment-sample
-primary-0                     1/1     Running   0          86m
pod/x10-240-0-12.veritas
.internal                      1/1     Running   0          68m
pod/x10-240-0-13.veritas
.internal                      2/2     Running   0          64m
pod/x10-240-0-14.veritas
.internal                      2/2     Running   0          61m
pod/x10-240-0-15.veritas
.internal                      2/2     Running   0          59m
```

```
NAME                TYPE         CLUSTER-IP     PORT(S)           AGE
service/dedupe1-
uss-controller      ClusterIP    10.0.109.118   10100/TCP         68m
service/dedupe1-
uss-mds             ClusterIP    None           2379/TCP,2380/TCP 75m
service/dedupe1-
uss-mds-client      ClusterIP    10.0.5.208     2379/TCP          75m
service/media1-
media-0             LoadBalancer 10.0.121.115   13782:30648/TCP,

                                   1556:30248/TCP    54m
service/
environment-
sample-primary      LoadBalancer 10.0.206.39    13781:30246/TCP,

                                   13782:30498/TCP,

                                   1556:31872/TCP,

                                   443:30049/TCP,

                                   8443:32032/TCP,

                                   22:31511/TCP     87m
service/
x10-240-0-12
-veritas-internal LoadBalancer 10.0.44.188    10082:31199/TCP   68m
service/
x10-240-0-13
-veritas-internal LoadBalancer 10.0.21.176    10082:32439/TCP,  68m
service/
x10-240-0-14                                  10102:30284/TCP
-veritas-internal LoadBalancer 10.0.25.99     10082:31810/TCP,  68m
service/
x10-240-0-15                                  10102:31755/TCP
-veritas-internal LoadBalancer 10.0.185.135   10082:31664/TCP,  68m
                                              10102:31811/TCP

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/dedupe1
 -uss-controller            1/1     1            1           68m

NAME                        DESIRED   CURRENT   READY   AGE
replicaset.apps/dedupe1-uss
-controller-79d554f8cc      1         1         1       68m
```

```
NAME                           READY   AGE
statefulset.apps/media1-media  1/1     53m
statefulset.apps/environment
 -sample-primary               1/1     86m

NAME                           TAG    AGE   STATUS
primaryserver.netbackup
.veritas.com/environment
-sample                        10.0   88m   Success

NAME                           TAG    AGE   PRIMARY SERVER   STATUS
mediaserver.netbackup.                      x10-240-0-10
veritas.com/media1             10.0   54m   .veritas.internal Success

NAME                           AGE   TAG    SIZE   READY
msdpscaleout.msdp.
veritas.com/dedupe1            75m   16.0   4      4

NAME                           READY   AGE   STATUS
environment.netbackup.
veritas.com/
environment-sample             3/3     88m   Success
```

An environment is deployed successfully if all pods and environment CR display status as "Success".

View operator logs

If environment deployment status is not successful, check operator logs for errors.

Command for MSDP Scaleout operator logs

```
$ kubectl logs pod/msdp-operator-controller-manager-65d8fd7c4d-whqpm
manager -n netbackup-operator-system-c manager
```

Command for NetBackup operator logs

```
$ kubectl logs
pod/netbackup-operator-controller-manager-55d6bf59c8-vltmp
netbackup-operator -n netbackup-operator-system
```

View primary logs

To view primary server logs execute the following command to get a shell to the running container.

```
$ kubectl exec --stdin --tty pod/<primary-server-pod-name> -n
<namespace> -- /bin/bash
```

Once in the primary server shell prompt, to see the list of logs, run:

```
ls /usr/openv/logs/
```

Pod restart failure due to liveness probe time-out

As part of liveness probe for primary and media pods, a health script runs inside the container to check the NetBackup health status.

When there is an issue with a container related to a full disk, CPU, or memory pressure, the liveness probe gets timed out because of no response from the health script. As a result, the Pod does not restart.

To resolve this issue, restart the Pod manually. Delete the Pod using the `kubectl delete pod/<podname> -n <namespace>` command.

The Pod is deleted and Kubernetes creates another Pod.

Socket connection failure

Socket connection failure can happen because of the following reasons:

- Long processing delays
- Azure connection reset (default 4 minutes)
- Load on CPU or Memory pressure
- IO saturation and throttling under load

If there are problems with the TCP stacks on the hosts, network between the hosts, or unusual long processing delays, then the connection may drop and the TCP stack on the host is unaware of the situation.

The following error is displayed in the web UI under job details:

```
db_FLISTsend failed: unexpected message received (43)
 *** - Error bptm (pid=14599) get_string() failed,
Connection reset by peer (104), network read error
*** - Info bptm (pid=14599) EXITING with status 42 <----------
*** - Info nbux-systest-media-1 (pid=14599)
```

```
StorageServer=PureDisk:nbux-systest-media-1;
Report=PDDO Stats for (nbux-systest-media-1):
scanned: 4195521 KB, CR sent: 171002 KB, CR sent over FC: 0 KB,
dedup: 95.9%, cache disabled, where dedup space saving:6.6%,
compression space saving:89.3%
*** - Info bpbkar (pid=19109) done. status: 42: network read failed
```

To resolve this issue, update the `sysctl.conf` values for NetBackup servers deployed on the AKS cluster.

NetBackup image sets following values in `sysctl.conf` during AKS deployment:

- net.ipv4.tcp_keepalive_time = 180
- net.ipv4.tcp_keepalive_intvl = 10
- net.ipv4.tcp_keepalive_probes = 20
- net.ipv4.ip_local_port_range = 14000 65535

These settings are persisted at the location `/mnt/nbdata/etc/sysctl.conf`.

There are two ways to modify these values:

- Modify the value in both `/etc/sysctl.conf` and `/mnt/nbdata/etc/sysctl.conf` and run the `sysctl -p` command to load the modified values.
- Modify the values in `/mnt/nbdata/etc/sysctl.conf` and restart the pod. The new values are reflected after the pod restart.

If external media servers are used, perform the steps in the following order:

1. Add the following in `/usr/openv/netbackup/bp.conf`:

   HOST_HAS_NAT_ENDPOINTS = YES

2. Add the following sysctl configuration values in `etc/sysctl.conf` on external media servers to avoid any socket connection issues:
   - net.ipv4.tcp_keepalive_time = 180
   - net.ipv4.tcp_keepalive_intvl = 10
   - net.ipv4.tcp_keepalive_probes = 20
   - net.ipv4.ip_local_port_range = 14000 65535
   - net.core.somaxconn = 4096

3. Save the setting using the `sysctl -p` command.

Resolving an invalid license key issue

The NetBackup is not installed because the license key is invalid.

Pod remains in running state for long time and the installation log at `/mnt/nblogs/setup-server.log` displays the following error:

```
ERROR: No valid license key for NetBackup Server or Enterprise Server
```

When you deploy NetBackup for the first time, perform the steps for primary CR and media CR.

To resolve an invalid license key issue for Primary/Media CR

1. Get the configmap name created for primary CR or media CR using the following command:

   ```
   kubectl get configmap -n <namespace>
   ```

2. Edit the license key stored in configmap using the following command:

   ```
   kubectl edit configmap <primary/media-configmap-name> -n
   <namespace>
   ```

3. Update value for **ENV_NB_LICKEY** key in the configmap with correct license key and save.

4. Delete respective primary/media pod using the following command:

   ```
   kubectl delete pod<primary/media-pod-name> -n <namespace>
   ```

   New pod is auto created with updated license key value.

5. Edit environment CR with updated license key and save using the following command:

   ```
   kubectl edit environments.netbackup.veritas.com -n <namespace>
   ```

To resolve an invalid license key issue for Media CR

1. Delete the media server CR by removing the `mediaServer` section in `environment.yaml` and save the changes.

   Note: Ensure that you copy spec information of the media server CR. The spec information is used to reapply the media server CR.

   ---

2. Apply the new changes using the `kubectl apply -f <environment.yaml>` command.

3 Delete respective persistent volume claim using the `kubectl delete pvc <pvc_name> -n <namespace>` command. Any available persisted data is deleted.

4 Add the **mediaServer** section, update the license key, and reapply the `environment.yaml` using the `kubectl apply -f <environment.yaml>` command.

Resolving an issue where external IP address is not assigned to a NetBackup server's load balancer services

The issue can be because of one of the following reasons:

- The **resourcePrefixName** mentioned in custom resource is not unique and valid.
- The static IP is provided in **networkLoadBalancer** section in CR but it is not created in AKS.
- The subnet or resource group is mentioned in annotations of **networkLoadBalancer** section in CR spec, the IP address is not available in given subnet or resource group.
- The RBAC permissions in your cluster for the given subnet or resource group are not assigned properly for allocating IP addresses.

To resolve an issue where external IP address is not assigned to a NetBackup server's load balancer services

1 Check the event logs of load balancer service using the `kubectl describe service <svc-name> -n <namespace>` command for detailed error information.

2 Run the `kubectl edit Environment <environmentCR-name> -n <namespace>` command.

3 Depending on the output of the command and the reason for the issue, perform the required steps and update the environment CR to resolve the issue.

Resolving the issue where the NetBackup server pod is not scheduled for long time

The NetBackup server (primary server and media server) pods are stuck in Pending state. The issue can be because of one of the following reasons:

- Insufficient resource allocation.
- Persistent volume claims are not bound to persistent volume.
- NetBackup server pods have the anti-affinity rule added.

As a result, primary server and media server pods are scheduled on different nodes. If nodes are not available, pod remains in pending state with event logs indicating nodes are scaling up, if auto scaling is configured in cluster.

To resolve the issue where the NetBackup server pod is not scheduled for long time

1. Check the pod event details for more information about the error using `kubectl describe <PrimaryServer/MediaServer_Pod_Name> -n <namespace>` command.
2. Depending on the output of the command and the reason for the issue, perform the required steps and update the environment CR to resolve the issue.

Resolving an issue where the Storage class does not exist

The Config-Checker checks if the storage class name given in primary server/media server CR is available in the cluster.

The following error is displayed:

```
Error: ERROR Storage class with the <storageClassName> name does not exist.
```

After fixing this error, primary server or media server CR does not require any changes. In this case, NetBackup operator reconciler loop is invoked after every 10 hours. If you want to reflect the changes and invoke the NetBackup operator reconciler loop immediately, delete and reapply the primary server or media server CR.

To resolve an issue where the Storage class does not exist

1. Create storage class with the same name given in primary server or media server CR with ReclaimPolicy as **Retain** in the cluster.

   To create storage class, refer to the following link:

   https://docs.microsoft.com/en-us/azure/aks/concepts-storage#storage-classes

   In this scenario, no change in primary server or media server CR is required. As a result, reconciler loop is not invoked immediately.

2. To invoke the reconciler loop again, delete the respective CR.

   If it is primary server CR, use the `kubectl delete -f <environment.yaml>` command, or if it is media server CR, edit the Environment CR by removing the media server section in the `environment.yaml`.

   Note: To reuse the **mediaServer** section information, you must save it and apply the yaml again with the new changes using the `kubectl apply -f <environment.yaml>` command.

   ---

3. Apply the CR again.

   If it is primary server CR then reapply it using the `kubectl apply -f <environment.yaml>` command or if it is media server CR, add **mediaServer** section that was deleted earlier with required data in `environment.yaml` and reapply it using the `kubectl apply -f <environment.yaml>` command.

Resolving an issue where the primary server or media server deployment does not proceed

primary server or media server deployment does not proceed even if **configcheckmode = default** in primary server or media server CR spec and no other child resources are created. It is possible that the Config-Checker job has failed some of the configuration checks.

To resolve an issue where the primary server or media server deployment does not proceed

1. Check the status of Config-Checker **Configcheckerstatus** mentioned in primary server or media server CR status using the `kubectl describe <PrimaryServer/MediaServer> <CR name> -n <namespace>` command.

   If the state is **failed**, check the Config-Checker pod logs.

2. Retrieve the Config-Checker pod logs using the `kubectl logs <config-checker-pod-name> -n <operator-namespace>` command.

   Config-Checker pod name can be in the following format:

   `<serverType>-configchecker-<configcheckermode>-randomID`, for example if its Config-Checker for primary server with **configcheckermode = default**, pod name is `primary-configcehcker-default-dhg34`.

3. Depending on the error in the pod logs, perform the required steps and edit the environment CR to resolve the issue.

Resolving an issue of failed probes

If pod is not in ready state for log time, the `kubectl describe pod/<podname> -n <namespace>` command displays the following errors:

-
  ```
  Readiness probe failed: The readiness of
  the external dependencies is not set.

  Server setup is still in progress.
  ```

-
  ```
  Liveness probe failed: bpps command did
  not list nbwmc process. nbwmc is not alive.

  The Primary server is unhealthy.
  ```

To resolve an issue of failed probes

1. If you are deploying NetBackup on Azure Kubernetes Cluster for the first time, check the installation logs for detailed error.

   Use any of the following methods:

   - Execute the following command in the respective primary server or media server pod and check the logs in `/mnt/nblogs/setup-server.logs`:

     ```
     kubectl exec -it -n <namespace> <pod-name> -- /bin/bash
     ```

- Run the `kubectl logs pod/<podname> -n <namespace>` command.

2 Check pod events for obtaining more details for probe failure using the following command:

  ```
  kubectl describe pod/<podname> -n <namespace>
  ```

  Kubernetes will automatically try to resolve the issue by restarting the pod after liveness probe times out.

3 Depending on the error in the pod logs, perform the required steps or contact technical support.

Resolving token issues

Media server installation log displays the following error in `/mnt/nblogs/setup-server.logs:`

```
nbcertcmd: The -getCertificate operation
failed for server <primaryServerName>,
EXIT STATUS 5940: Reissue token is mandatory,
please provide a reissue token
```

NetBackup media server and NetBackup primary server were in running state. Media server persistent volume claim or media server pod is deleted. In this case, reinstallation of respective media server can cause the issue.

To resolve token issues

1 Open the NetBackup web UI using primary server hostname given in the primary server CR status.

2 Navigate to **Security > Host Mappings**.

3 Click **Actions > Allow auto reissue certificate** for the respective media server name.

4 Delete data and logs PVC for respective media server only using the `kubectl delete pvc <pvc-name> -n <namespace>` command.

  The persisted data is deleted.

5 Delete respective media server pod using `kubectl delete <pod-name> -n <namespace>` command.

  New media server pod and new PVCs for the same media server are created.

Resolving an issue related to insufficient storage

Setup-server.logs of NetBackup primary server displays an error.

Insufficient storage on the node can cause this issue. Minimum hardware requirements for NetBackup may not be completed. During fresh deployment of primary server, the following error is displayed in `/mnt/nblogs/setup-server.logs`:

```
DBSPAWN ERROR: -86
Not enough memory to start
```

To resolve an issue related to insufficient storage

1. Create a new nodepool with the hardware specifications as mentioned in the `NetBackup Deployment on AKS Administrator's Guide`.
2. Run the `kubectl get nodes` command to ensure that the nodes from the newly created nodepool are used in your cluster.
3. Delete the primary server CR using the `kubectl delete -f <environment.yaml>` command.
4. Update **nodeSelector** spec in primary section and apply the `environment.yaml` again using the `kubectl apply -f <environment.yaml>` command.

Resolving an issue related to invalid nodepool

Invalid nodepool is mentioned in primary server or media server CR nodeSelector spec. Due to this, primary server or media server pod fails to schedule.

The following error is displayed:

```
Error: Did not match Pod’s node affinity/selector.
```

To resolve an issue related to invalid nodepool

1. If you are deploying NetBackup on Azure Kubernetes Cluster for the first time, delete the respective CR.

   If it is primary server CR:

   - Delete it using the `kubectl delete -f <environment.yaml>` command.

- Update the node selector in primary server section in `environment.yaml` and apply it again using the `kubectl apply -f <environment.yaml>` command.

2. For media server CR: Delete the media server CR by removing the **mediaServer** section in the `environment.yaml` and save the changes.

   Note: Ensure that you copy spec information of the media server CR. The spec information is used to reapply the media server CR.

   ---

3. Apply the new changes using the `kubectl apply -f <environment.yaml>` command.
4. Add the **mediaServer** section, update the nodeSelector, and reapply the `environment.yaml` using the `kubectl apply -f <environment.yaml>` command.

Resolving a token expiry issue

While creating a new media pod, token may expire, and installation of media server is not completed. The installation logs at `/mnt/nblogs/setup-server.logs` display an error on the respective media server.

```
EXIT STATUS 5934: The token has expired.
```

To resolve a token expiry issue

1. Edit the environment server CR using the `kubectl edit environment <environment-CR-name> -n <namespace>` command.
2. In the **mediaServer** section, reduce the replica count.

   For example, if media pod with name xyz-media-2 has the token expired issue and the replica was originally 3, then change the replica count to 2. Save the changes. The extra pods are deleted and statefulset displays new replica count in ready state (2/2).
3. Edit the media server CR using the `kubectl edit MediaServer <mediaServer-CR-name> -n <namespace>` command.
4. Increase replica count to original replica count.

   As given in the example, change the replica count to 3. This creates additional media pods and reissues the token for newly added media server.

Resolve an issue related to inconsistency in file ownership

There is inconsistency between ownership of most of the files (for example, ownership of files in folder `/usr/openv/netbackup/bin/*`) and the service user that is available in the `bp.conf` file.

The issue is observed when you run the `nbserviceusercmd` command to change the service account user but this command causes inconsistency in ownership of files.

To bring system in default state, use the `/opt/veritas/vxapp-manage/change-service-user -user nbsvcusr` command.

Resolve an issue related to KMS database

Installation logs at `/mnt/nblogs/setup-server.logs` display an error message with other details. In this scenario, you must configure KMS manually.

```
Error: Failed to create KMS database
```

To resolve this issue, execute the following command in the primary server pod:

```
kubectl exec -it -n <namespace> <primary-server-pod-name> -- /bin/bash
```

Refer the NetBackup Security and Encryption Guide for configure KMS manually:

For other troubleshooting issue related to KMS, refer the NetBackup Troubleshooting Guide.

Resolve an issue related to pulling an image from the container registry

Primary or media server failed to deploy with **ImagePullBackOff** error. If the pod Status field displays **ImagePullBackOff**, it means that the pod could not start because Kubernetes cannot pull a container image. A misspelled registry or image name or image registry being not reachable can cause a **ImagePullBackOff** status.

Run the `$ k get all -n netbackup-operator-system` command.

The output should look like:

```
            NAME               READY   STATUS                RESTARTS   AGE
pod/msdp-operator
-controller-manager-
```

```
65d8fd7c4d-bsgms            2/2     Running           0         7m9s

pod/netbackup-operator
-controller-manager-
5df6f58b9b-6ftt9            1/2     ImagePullBackOff  0         13s
```

For additional details, use the following command:

```
$ kubectl describe pod/<pod_name> -n netbackup-operator-system
```

Resolve this issue using any of the following methods:

- Check if image name and tag are correct. If not, edit and update the environment CR using the `kubectl edit environment <environment CR-name> -n <namespace>` command with correct image name and tag, and then save the changes.
- Check if the user is authorized and has permissions to access the Azure container registry.

Resolving an issue related to recovery of data

If a PVC is deleted or the namespace where primary or media server is deployed, is deleted or deployment setup is uninstalled, and you want to recover the previous data, attach the primary server and media server PVs to its corresponding PVCs.

In case of recovering data from PV, you must use the same environment CR specs that are used at the time of previous deployment. If any spec field is modified, data recovery may not be possible.

To resolve an issue related to recovery of data

1. Run the `kubectl get PV` command.
2. From the output list, note down PV names and its corresponding claim (PVC name and namespace) that are relevant from previous deployment point of view.
3. Set claim ref for the PV to null using the `kubectl patch pv <pv name> -p '{"spec":{"claimRef": null}}'` command.

   For example, `kubectl patch pv pvc-4df282e2-b65b-49b8-8d90-049a27e60953 -p '{"spec":{"claimRef": null}}'`
4. Run the `kubectl get PV` command and verify bound state of PVs is **Available**.

5. For the PV to be claimed by specific PVC, add the **claimref spec** field with PVC name and namespace using the `kubectl patch pv <pv-name> -p '{"spec":{"claimRef": {"apiVersion": "v1", "kind": "PersistentVolumeClaim", "name": "<Name of claim i.e. PVC name>", "namespace": "<namespace of pvc>"}}}'` command.

   For example,

   ```
   kubectl patch pv <pv-name> -p '{"spec":{"claimRef": {"apiVersion":
   "v1", "kind": "PersistentVolumeClaim", "name":
   "data-testmedia-media-0", "namespace": "test"}}}'
   ```

   While adding claimRef add correct PVC names and namespace to respective PV. Mapping should be as it was before deletion of the namespace or deletion of PVC.

6. Deploy environment CR that deploys the primary server and media server CR internally.

Check primary server status

Check the primary server custom resource status, with the command:

```
$ kubectl get primaryserver.netbackup.veritas.com/environment-sample
-n <namespace>
```

```
NAME                 TAG    AGE    STATUS
environment-sample   10.0   3h1m   Failed
```

If the output shows STATUS as *Failed* as in the example above, check the primary pod log for errors with the command:

```
$ kubectl logs pod/environment-sample-primary-0 -n <namespace>
```

Pod status field shows as pending

If the pod Status field shows Pending state, it indicates that Kubernetes is not able to schedule the pod. To check use the following command:

```
$ kubectl get all -n netbackup-operator-system
```

The output is something like:

```
NAME                         READY   STATUS    RESTARTS   AGE
pod/msdp-operator-
controller-manager-
65d8fd7c4d-bsgms               2/2     Running   0          12m
```

```
pod/netbackup-
operator-controller-
manager-6c9dc8d87f
-pq8mr                    0/2     Pending   0          15s
```

For more details use the following pod describe command:

```
$ kubectl describe
pod/netbackup-operator-controller-manager-6c9dc8d87f-pq8mr -n
netbackup-operator-system
```

The output is something like:

```
  Type     Reason            Age                Message
  ----     ------            ----                  -------
  Warning  FailedScheduling  56s (x3 over 2m24s)   0/4 nodes are
                                                   available:1 node(s)
                                                   had taint {node-
                                                   role.kubernetes.
                                                   io/master: }, that
                                                   the pod didn't
                                                   tolerate, 3 node(s)
                                                   didn't match
                                                   Pod's node
                                                   affinity/selector.
```

To resolve this issue verify the nodeSelector settings in the `operator/patch/operator_patch.yaml` file.

Ensure that the container is running the patched image

There are three copies of the container image present in the Kubernetes environment during deployment or patching.

The first image copy is created on a local docker instance during image load operation. To check this copy, do the following:

1. Run:

   ```
   $ docker load -i images/pdk8soptr-16.0.tar.gz
   ```

   Sample output:

   ```
   Loaded image: msdp-operator:16.0
   ```

2. Taking the image name from step 1, run:

   ```
   $ docker image ls | grep msdp-operator
   ```

   Sample output:

   ```
   msdp-operator   16.0  353d2bd50105   2 days ago  480 MB
   ```

3. Taking the value from step 2, run:

   ```
   $ docker inspect 353d2bd50105 | jq .[].Id

   "sha256:353d2bd50105cbc3c61540e10cf32a152432d5173bb6318b8e"
   ```

The second copy is created in Azure Container Registry (ACR). To check this copy, do the following:

1. Run:

   ```
   $ docker image tag msdp-operator:16.0
   testregistry.azurecr.io/msdp-operator:16.0
   ```

2. Run:

   ```
   $ docker image ls | grep msdp-operator
   ```

   Sample output:

   ```
    msdp-operator            16.0 353d2bd50105 2 days ago 480 MB
    tregistry.azurecr.io/msdp-operator 16.0 353d2bd50105 2 days ago 480 MB
   ```

3. To push the image to the registry, run:

   ```
   $ docker push testregistry.azurecr.io/msdp-operator
   ```

   The push refers to a repository *[testregistry.azurecr.io/msdp-operator]*

   0a504041c925: Layer already exists

   ```
   16.0: digest:
   sha256:d294f260813599562eb5ace9e0acd91d61b7dbc53c3 size:
    2622
   ```

4 To verify local image digest after the push operation, run:

```
$ docker inspect 353d2bd50105 | jq .[].RepoDigests
```

Sample output:

```
[
   "testregistry.azurecr.io/msdp-operator@sha256:
d294f260813599562eb5ace9e0acd91d61b7dbc53c3"
 ]
```

5 To verify image presence in the registry, run:

```
$ az acr repository list --name testregistry
```

Sample output:

```
 [
   "msdp-operator",
 ]
```

6 To verify image digest in registry, run:

```
$ az acr repository show -n testregistry --image
msdp-operator:16.0
```

Sample output:

```
{
   "changeableAttributes": {
     "deleteEnabled": true,
     "listEnabled": true,
     "readEnabled": true,
     "writeEnabled": true
   },
   "createdTime": "2022-02-01T13:43:26.6809388Z",
   "digest": "sha256:d294f260813599562eb5ace9e0acd91d61b7dbc53c3",
   "lastUpdateTime": "2022-02-01T13:43:26.6809388Z",
   "name": "16.0",
   "signed": false
 }
```

The third copy is located on a Kubernetes node running the container after it is pulled from the registry. To check this copy, do the following:

1 Run;

```
$ kubectl get nodes -o wide
```

```
NAME                    STATUS   VERSION   INTERNAL-IP   OS-IMAGE
aks-agentpool-7601-vmss000 Ready  v1.21.7 10.240.0.4 Ubuntu 18.04.6 LTS
```

2 Use kubectl debug to run a container on the node:

```
$ kubectl debug node/aks-nodepool1-7601-vmss000-it
--image=mcr.microsoft.com/aks/fundamental/base-ubuntu:v0.0.11
root@aks-agentpool-7601-vmss000:/#
```

3 You can interact with the node session from the privileged container:

```
chroot /host
```

4 Verify the presence of the image:

```
/usr/local/bin/crictl image | grep msdp
```

Sample output:

```
testregistry.azurecr.io/msdp-operator  16.0  353d2bd50105c  182MB
```

5 Verify the image ID on the Kubernetes node, run:

```
/usr/local/bin/crictl inspecti 353d2bd50105c | jq .[].id
```

Sample output

```
"sha256:353d2bd50105cbc3c61540e10cf32a152432d5173bb6318b8e"
null
```

6 Verify the image digest on the Kubernetes node, run:

```
/usr/local/bin/crictl inspecti 353d2bd50105c | jq .[].repoDigests
```

Sample output

```
[
  "testregistry.azurecr.io/msdp-operator@sha256:
d294f260813599562eb5ace9e0acd91d61b7dbc53c3"
]
null
```

How to make sure that you are running the correct image

Use the steps given above to identify image ID and Digest and compare with values obtained from the registry and the Kubernetes node running the container.

Note: MSDP Scaleout images (uss-engine, uss-mds, uss-controller, msdp-operator) use IfNotPresent imagePullPolicy. A unique image tag is required in order for a Kubernetes node to pull an updated image.

Getting EEB information from an image, a running container, or persistent data

To view the list of installed EEBs, run the `nbbuilder` script provided in the EEB file archive.

```
$ bash nbbuilder.sh -registry_name testregistry.azurecr.io
-list_installed_eebs -nb_src_tag=10.0-2 -msdp_src_tag=16.0-2
```

Sample output:

```
Wed Feb 2 20:48:13 UTC 2022: Listing strings for EEBs
installed in testregistry.azurecr.io/netbackup/main:10.0-2.
EEB_NetBackup_10.0Beta6_PET3980928_SET3992004_EEB1
EEB_NetBackup_10.0Beta6_PET3980928_SET3992021_EEB1
EEB_NetBackup_10.0Beta6_PET3980928_SET3992022_EEB1
EEB_NetBackup_10.0Beta6_PET3980928_SET3992023_EEB1
EEB_NetBackup_10.0Beta6_PET3992020_SET3992019_EEB2
EEB_NetBackup_10.0Beta6_PET3980928_SET3992009_EEB2
EEB_NetBackup_10.0Beta6_PET3980928_SET3992016_EEB1
EEB_NetBackup_10.0Beta6_PET3980928_SET3992017_EEB1
Wed Feb 2 20:48:13 UTC 2022: End
Wed Feb 2 20:48:13 UTC 2022: Listing strings for EEBs
installed in testregistry.azurecr.io/uss-controller:16.0-2.
EEB_MSDP_16.0_PET3980928_SET3992007_EEB1
EEB_MSDP_16.0_PET3992020_SET3992019_EEB2
EEB_MSDP_16.0_PET3980928_SET3992010_EEB2
Wed Feb 2 20:48:14 UTC 2022: End
Wed Feb 2 20:48:14 UTC 2022: Listing strings for EEBs
installed in testregistry.azurecr.io/uss-engine:16.0-2.
EEB_MSDP_16.0_PET3980928_SET3992006_EEB1
EEB_MSDP_16.0_PET3980928_SET3992023_EEB1
EEB_MSDP_16.0_PET3992020_SET3992019_EEB2
```

```
EEB_MSDP_16.0_PET3980928_SET3992009_EEB2
EEB_MSDP_16.0_PET3980928_SET3992010_EEB2
EEB_MSDP_16.0_PET3980928_SET3992018_EEB1
Wed Feb 2 20:48:14 UTC 2022: End
Wed Feb 2 20:48:14 UTC 2022: Listing strings for EEBs
installed in testregistry.azurecr.io/uss-mds:16.0-2.
EEB_MSDP_16.0_PET3980928_SET3992008_EEB1
EEB_MSDP_16.0_PET3992020_SET3992019_EEB2
EEB_MSDP_16.0_PET3980928_SET3992010_EEB2
Wed Feb 2 20:48:15 UTC 2022: End
```

Alternatively, if the `nbbuilder` script is not available, you can view the installed EEBs by executing the following command:

```
$ docker run --rm <image_name>:<image_tag> cat
/usr/openv/pack/pack.summary
```

Sample output:

```
EEB_NetBackup_10.0Beta6_PET3980928_SET3992004_EEB1
EEB_NetBackup_10.0Beta6_PET3980928_SET3992021_EEB1
EEB_NetBackup_10.0Beta6_PET3980928_SET3992022_EEB1
EEB_NetBackup_10.0Beta6_PET3980928_SET3992023_EEB1
EEB_NetBackup_10.0Beta6_PET3992020_SET3992019_EEB2
EEB_NetBackup_10.0Beta6_PET3980928_SET3992009_EEB2
EEB_NetBackup_10.0Beta6_PET3980928_SET3992016_EEB1
EEB_NetBackup_10.0Beta6_PET3980928_SET3992017_EEB1
```

To view all EEBs installed in a running container, run:

```
$ kubectl exec --stdin --tty <primary-pod-name> -n <namespace> --
cat /usr/openv/pack/pack.summary
```

Note: The pack directory may be located in different locations in the `uss-*` containers. For example: `/uss-controller/pack , /uss-mds/pack, /uss-proxy/pack.`

To view a list of installed data EEBs from a running container, run:

```
$ kubectl exec --stdin --tty <primary-pod-name> -n <namespace> --
cat /mnt/nbdata/usr/openv/pack/pack.summary
```

CR template

This appendix includes the following topics:

Secret

The Secret is the Kubernetes security component that stores the MSDP credentials that are required by the CR YAML.

```
# The secret is used to store the MSDP credential, which is required
by the CR YAML as follows.
# This part should be created separately and not be part of CR Template.
# The secret should have a "username" and a "password" key-pairs with the
corresponding username and password values.
# Please follow MSDP guide for the rules of the credential.
#   https://www.veritas.com/content/support/en_US/article.100048511
#   The pattern is "^[\\w!$+\\-,.:;=?@[\\]`{}\\|~]{1,62}$"
# We can create the secret directly via kubectl command:
#   kubectl create secret generic sample-secret --namespace
sample-namespace \
#   --from-literal=username=<username> --from-literal=password=<password>
# Alternatively, we can create the secret with a YAML file in the
following format.
apiVersion: v1
kind: Secret
metadata:
  name: sample-secret
  # The namespace needs to be present.
  namespace: sample-namespace
```

```
stringData:
  # Please follow MSDP guide for the credential characters and length.
  #   https://www.veritas.com/content/support/en_US/article.100048511
  #   The pattern is "^[\\w!$+\\-,.:;=?@[\\]`{}\\|~]{1,62}$"
  username: xxxx
  password: xxxxxx
```

MSDP Scaleout CR

- The CR name must be less than 40 characters.
- The MSDP credentials stored in the Secret must match MSDP credential rules. See Deduplication Engine credentials for NetBackup
- MSDP CR cannot be deployed in the namespace of MSDP operator. It must be in a separate namespace.
- You cannot reorder the IP/FQDN list. You can update the list by appending the information.
- You cannot change the storage class name. The storage class must be backed with Azure disk CSI storage driver "disk.csi.azure.com".
- You cannot change the data volume list other than for storage expansion. It is append-only and storage expansion only. Up to 16 data volumes are supported.
- Like the data volumes, the catalog volume can be changed for storage expansion only.
- You cannot change or expand the size of the log volume by changing the MSDP CR.
- You cannot enable NBCA after the configuration.
- Once KMS and the OST registration parameters set, you cannot change them.
- You cannot change the core pattern.

MSDP Scaleout CR template:

```
# The MSDPScaleout CR YAML
apiVersion: msdp.veritas.com/v1
kind: MSDPScaleout
metadata:
  # The CR name should not be longer than 40 characters.
  name: sample-app
  # The namespace needs to be present for the CR to be created in.
  # It's not allowed to deploy the CR in the same namespace with MSDP
operator.
```

```
  namespace: sample-namespace
spec:
  # Your ACR URL where the docker images can be pulled from by the
AKS cluster on demand
  # The allowed length is in range 1-255
  # It's optional for BYO. The code does nt check the presence or
validation.
  # User needs to specify it correctly if it's needed.
  containerRegistry: sample.azurecr.io
  #
  # The MSDP version string. It's the tag of the MSDP docker images.
  # The allowed length is in range 1-64
  version: "sample-version-string"
  #
  # Size defines the number of Engine instances in MSDP Scaleout.
  # The allowed size is between 1-16
  size: 4
  #
  # The IP and FQDN pairs are used by the Engine Pods to expose the MSDP
services.
  # The IP and FQDN in one pair should match each other correctly.
  # They must be pre-allocated.
  # The item number should match the number of Engine instances.
  # They're not allowed to be changed or re-ordered. New items can be
appended for scaling out.
  # The first FQDN is used to configure the storage server in NetBackup,
automatically if autoRegisterOST is enabled,
  # or manually by the user if not.
  serviceIPFQDNs:
    # The pattern is IPv4 or IPv6 format
  - ipAddr: "sample-ip1"
    # The pattern is FQDN format. `^[a-z][a-z0-9-.]{1,251}[a-z0-9]$`
    fqdn: "sample-fqdn1"
  - ipAddr: "sample-ip2"
    fqdn: "sample-fqdn2"
  - ipAddr: "sample-ip3"
    fqdn: "sample-fqdn3"
  - ipAddr: "sample-ip4"
    fqdn: "sample-fqdn4"
  #
  # Optional annotations to be added in the LoadBalancer services for the
Engine IPs.
  # In case we run the Engines on private IPs, we need to add some
```

```
customized annotations to the LoadBalancer services.
  # See https://docs.microsoft.com/en-us/azure/aks/internal-lb
  # It's optional. It's not needed in most cases if we're
with public IPs.
  # loadBalancerAnnotations:
  #   service.beta.kubernetes.io/azure-load-balancer-internal: "true"
  #
  # SecretName is the name of the secret which stores the MSDP credential.
  # AutoDelete, when true, will automatically delete the secret specified
by SecretName after the
  # initial configuration. If unspecified, AutoDelete defaults to true.
  # When true, SkipPrecheck will skip webhook validation of the MSDP
credential. It is only used in data re-use
  # scenario (delete CR and re-apply with pre-existing data) as the
secret will not take effect in this scenario. It
  # can't be used in other scenarios. If unspecified, SkipPrecheck
defaults to false.
  credential:
    # The secret should be pre-created in the same namespace which has
the MSDP credential stored.
    # The secret should have a "username" and a "password" key-pairs
with the corresponding username and password values.
    # Please follow MSDP guide for the rules of the credential.
    #   https://www.veritas.com/content/support/en_US/article.100048511
    # A secret can be created directly via kubectl command or with the
equivalent YAML file:
    #   kubectl create secret generic sample-secret --namespace
sample-namespace \
    #   --from-literal=username=<username> --from-literal=password=
<password>
    secretName: sample-secret
    # Optional
    # Default is true
    autoDelete: true
    # Optional
    # Default is false.
    # Should be specified only in data re-use scenario (aka delete and
re-apply CR with pre-existing data)
    skipPrecheck: false
  #
  # Paused is used for maintenance only. In most cases you don't need
to specify it.
  # When it's specified, MSDP operator stops reconciling the corresponding
```

```
MSDP-X (aka the CR).
  # Optional.
  # Default is false
  # paused: false
  #
  # The storage classes for logVolume, catalogVolume and dataVolumes should
be:
  #   - Backed with Azure disk CSI driver "disk.csi.azure.com" with the
managed disks, and allow volume
  #     expansion.
  #   - The Azure in-tree storage driver "kubernetes.io/azure-disk" is not
supported. You need to explicitly
  #     enable the Azure disk CSI driver when configuring your AKS cluster,
or use k8s version v1.21.x which
  #     has the Azure disk CSI driver built-in.
  #   - In LRS category.
  #   - At least Standard SSD for dev/test, and Premium SSD or Ultra Disk
for production.
  #   - The same storage class can be used for all the volumes.
  #   -
  #
  # LogVolume is the volume specification which is used to provision a
volume of an MDS or Controller
  # Pod to store the log files and core dump files.
  # It's not allowed to be changed.
  # In most cases, 5-10 GiB capacity should be big enough for one MDS or
Controller Pod to use.
  logVolume:
    storageClassName: sample-azure-disk-sc1
    resources:
      requests:
        storage: 5Gi
  #
  # CatalogVolume is the volume specification which is used to provision a
volume of an MDS or Engine
  # Pod to store the catalog and metadata. It's not allowed to be changed
unless for capacity expansion.
  # Expanding the existing catalog volumes expects short downtime of the
Engines.
  # Please note the MDS Pods don't respect the storage request in
CatalogVolume, instead they provision the
  # volumes with the minimal capacity request of 500MiB.
  catalogVolume:
    storageClassName: sample-azure-disk-sc2
    resources:
      requests:
        storage: 600Gi
  #
  # DataVolumes is a list of volume specifications which are used to
provision the volumes of
  # an Engine Pod to store the MSDP data.
  # The items are not allowed to be changed or re-ordered unless for
capacity expansion.
  # New items can be appended for adding more data volumes to each
Engine Pod.
  # Appending new data volumes or expanding the existing data volumes
expects short downtime of the Engines.
  # The allowed item number is in range 1-16. To allow the other MSDP-X
Pods (e.g. Controller, MDS) running
  # on the same node, the item number should be no more than "<the maximum
allowed volumes on the node> - 5".
  # The additional 5 data disks are for the potential one MDS Pod, one
Controller Pod or one MSDP operator Pod
  # to run on the same node with one MSDP Engine.
  dataVolumes:
    - storageClassName: sample-azure-disk-sc3
      resources:
        requests:
          storage: 8Ti
    - storageClassName: sample-azure-disk-sc3
      resources:
        requests:
          storage: 8Ti
  #
  # NodeSelector is used to schedule the MSDPScaleout Pods on the specified
nodes.
  # Optional.
  # Default is empty (aka all available nodes)
  nodeSelector:
    # e.g.
    # agentpool: nodepool2
    sample-node-label1: sampel-label-value1
    sample-node-label2: sampel-label-value2
  #
  # NBCA is the specification for MSDP-X to enable NBCA SecComm
for the Engines.
```

```
  # Optional.
  nbca:
    # The master server name
    # The allowed length is in range 1-255
    masterServer: sample-master-server-name
    # The CA SHA256 fingerprint
    # The allowed length is 95
    cafp: sample-ca-fp
    # The NBCA authentication/reissue token
    # The allowed length is 16
    # For security consideration, a token with maximum 1 user allowed and
valid for 1 day should be sufficient.
    token: sample-auth-token
  #
  # KMS includes the parameters to enable KMS for the Engines.
  # We support to enable KMS in init or post configuration.
  # We don't support to change the parameters once they have been set.
  # Optional.
  kms:
    # As either the NetBackup KMS or external KMS (EKMS) is configured or
registered on NetBackup master server, then used by
    # MSDP by calling the NetBackup API, kmsServer is the NetBackup master
server name.
    kmsServer: sample-master-server-name
    keyGroup: sample-key-group-name
  #
  # autoRegisterOST includes the parameter to enable or disable the
automatic registration of
  # the storage server, the default disk pool and storage unit when MSDP-X
configuration finishes.
  autoRegisterOST:
    # If it is true, and NBCA is enabled, the operator would register the
storage server,
    # disk pool and storage unit on the NetBackup primary server, when the
MSDP CR is deployed.
    # The first Engine FQDN is the storage server name.
    # The default disk pool is in format "default_dp_<firstEngineFQDN>".
    # The default storage unit is in format "default_stu_<firstEngineFQDN>".
    # The default maximum concurrent jobs for the STU is 240.
    # In the CR status, field "ostAutoRegisterStatus.registered" with value
True, False or Unknown indicates the registration state.
    # It's false by default.
    # Note: Please don't enable it unless with NB_9.1.2_0126+.
    enabled: true
  #
  # CorePattern is the core pattern of the nodes where the MSDPScaleout
Pods are running.
  # It's path-based. A default core path "/core/core.%e.%p.%t" will be
used if not specified.
  # In most cases, you don't need to specify it.
  # It's not allowed to be changed.
  # Optional.
  # corePattern: /sample/core/pattern/path
  #
  # tcpKeepAliveTime sets the namespaced sysctl parameter
net.ipv4.tcp_keepalive_time in Engine Pods.
  # It's in seconds.
  # The minimal allowed value is 60 and the maximum allowed value is 1800.
  # A default value 120 is used if not specified. Set it to 0 to disable
the option.
  # It's not allowed to change unless in maintenance mode (Paused=true),
and the change will not apply until the Engine Pods get restarted
  # For AKS deployment in P release, please leave it unspecified or specify
it with a value smaller than 240.
  # tcpKeepAliveTime: 120
  #
  # TCPIdleTimeout is used to change the default value for Azure Load
Balancer rules and Inbound NAT rules.
  # It's in minutes.
  # The minimal allowed value is 4 and the maximum allowed value is 30.
  # A default value 30 minutes is used if not specified. Set it to 0 to
disable the option.
  # It's not allowed to change unless in maintenance mode (Paused=true),
and the change will not apply
  # until the Engine Pods and the LoadBalancer services get recreated.
  # For AKS deployment in P release, please leave it unspecified or specify
it with a value larger than 4.
  # tcpIdleTimeout: 30
```