



Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

Contents

OpenDedupe OST Connector	3
Requirements:.....	3
Quick Start Instructions.....	3
Installing the OST Connector on a Backup Exec Server	4
Configuring OpenDedupe volumes and OST connections.....	5
Create an OST Storage Device using the Backup Exec Console	8
Creating a backup job using an OpenDedupe device	17
Scaling and Sizing	20
Data Stored on Disk	20
Memory Sizing:	22
CPU Sizing:.....	22
Troubleshooting (Logs).....	22
Using the command line to set up SDFS volumes on a Backup Exec Server	23
Detailed OST XML Configuration setup	24
Configuring OST Plugin logging.....	25
Creating a Cloud Backup Container	27
Creating an S3 Bucket Using Amazon Web Services (AWS).....	27
Creating a Blob Container Using Azure.....	30
Creating a Storage Bucket Using the Google Cloud Platform	34

OpenDedupe OST Connector

The OpenDedupe OST connector provides integration between Backup Exec and OpenDedupe. It supports the following OST features:

- Write/Backup to OpenDedupe volumes
- Read/Restore from OpenDedupe volumes

OpenDedupe provides an open source file system, SDFS, that includes inline deduplication to local or cloud storage targets. Backup specific features of OpenDedupe are as follows:

- **In-Line deduplication to a Cloud Storage Backend** - SDFS can send all the data to AWS, AZURE, Google, or any S3 compliant backend.
- **Performance** - Compressed Multi-Threaded upload and download of data to the cloud.
- **Local Cache** - SDFS caches the most recently accessed data locally. This is configurable but set to 10 GB by default.
- **Security** - All data can be encrypted using AES-CBC 256 when sent to the cloud.
- **Throttling** - Upload and download speeds can be throttled.
- **Cloud Recovery/Replication** - All local metadata is replicated to the cloud and can be recovered back to local volume.
- **AWS Region Support** - Supports all AWS Regions.

Requirements:

Windows Server - This is tested on Windows Server 2008 R2, Windows Server 2012 R2, and Windows Server 2016.

Backup Exec 16 FP1 - This is tested and developed for BE16 FP1

CPU: 2+ CPU System

Memory: 1GB of RAM + 256MB of RAM per TB of unique data stored

Disk:

- 2GB of additional local storage per TB of unique data.
- 21GB of additional storage per TB of logical storage that is used for file system metadata.
- 10GB for local cache of cloud data (Configurable).

NOTE: Currently Backup Exec supports OpenDedupe version 3.4 only.

Quick Start Instructions

If you plan to use OpenDedupe to back up data to a cloud destination, you must create a dedicated cloud container for each Backup Exec server that will run backups to the cloud. Examples of creating cloud containers for different cloud providers are at the end of this document in the section titled "[Creating a Cloud Backup Container](#)"

NOTE: OpenDedupe devices must not be shared between BE server nodes, even in an MMS/CAS environment. Do not use the same cloud bucket for two different BE servers. OpenDedupe does not support device sharing at this time. Although BE may allow you to share the device, you must not share it. Corruption of the dedupe data is a likely result.

Installing the OST Connector on a Backup Exec Server

First, install a minimum of Backup Exec 16 on a server, and select the dedupe option during setup. Or use a previously configured Backup Exec 16 server with the dedupe option. You may also use Backup Exec 16 Feature Pack 1 or later.

On the Backup Exec server perform the following steps:

1. Download the SDFS package from <http://opendedup.org/odd/download>
Select "Windows (Intel 64bit)" SDFS Binaries: *sdfs-latest.exe*.
2. On a Backup Exec Server, run *sdfs-latest.exe* to install the SDFS file system. The default installation directory is "C:\Program Files\sdfs".
A reboot is required at the end of this installation.
3. Download and install the OST package.
 - a. Download the OST plugin package from <http://www.opendedup.org/downloads/windows-ost-2.0.zip>
 - b. The content of the zipped file:
 - i. *libtspiopendedupe.dll* (OST plugin file for *Backup Exec*)
 - ii. *ostconfig.xml* (More details are included in the '**Detailed OST XML Configuration setup**' section)
 - iii. *ost-logging-global.conf* (script file to set up OpenDedupe logging)
 - iv. *readme.txt* (OpenDedupe OST plugin readme)
 - v. *vcredist_x64.exe* (Visual Studio 2013 runtime installer)
 - c. Extract everything from the zip file and copy the file named "*libtspiopendedupe.dll*" to the Backup Exec installation directory.
 - d. Run *vcredist_x64.exe* to install the Visual Studio 2013 library if it is not already installed.
4. Use the Veritas Backup Exec™ Configuration Tool for OpenDedupe described in the next section to configure OpenDedupe volumes and OST connections.

Configuring OpenDedupe volumes and OST connections

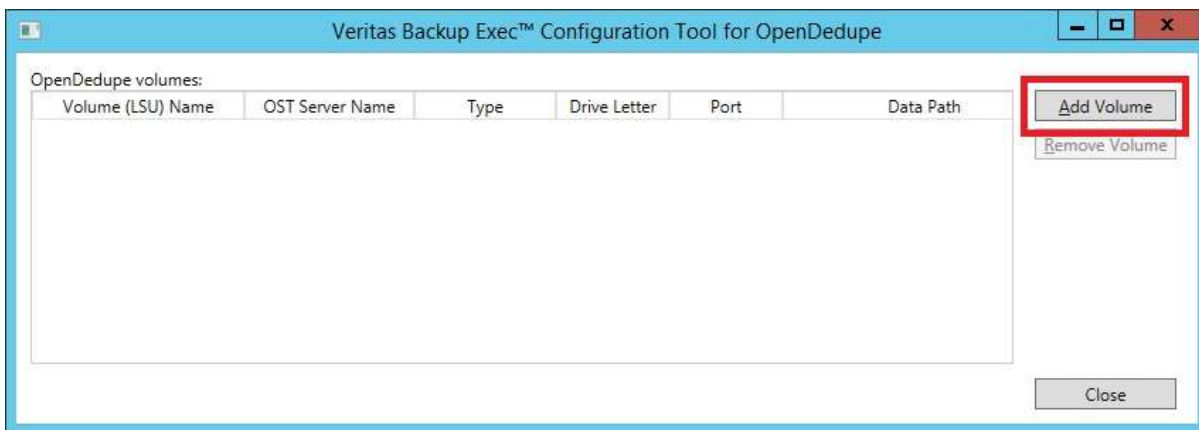
You can configure OpenDedupe volumes using the Veritas Backup Exec™ Configuration Tool for OpenDedupe found on the Backup Exec distribution medium in the path:

"BE\WinNT\UTILS\x64\OpenDedupe\OpenDedupeConfigUI.exe"

Copy this file to anywhere on your Backup Exec server after installing Backup Exec and the OpenDedupe packages described above. Use the tool to configure one or more OpenDedupe volumes and set them up to be used in Backup Exec.

As an alternative, you can use command-line tools described in the section entitled [Using the command line to set up SDFS volumes on a Backup Exec Server](#)

1. Click "Add Volume" to configure an OpenDedupe volume.



2. Select a Volume Name to be used for an OpenDedupe volume. This name corresponds to an OST Logical Storage Unit, or "LSU".

Add OpenDedupe Volume

Volume Settings

Volume (LSU) name: aws_dev_1

Volume type: AWS

Volume capacity: 1 TB

Mount drive letter: Q

Cached data path: D:\Program Files\sdfs\volumes\aws_dev_1

AWS

Access key: AKIAIOSFODNN7EXAMPLE

Secret key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

Bucket: sample-bucket

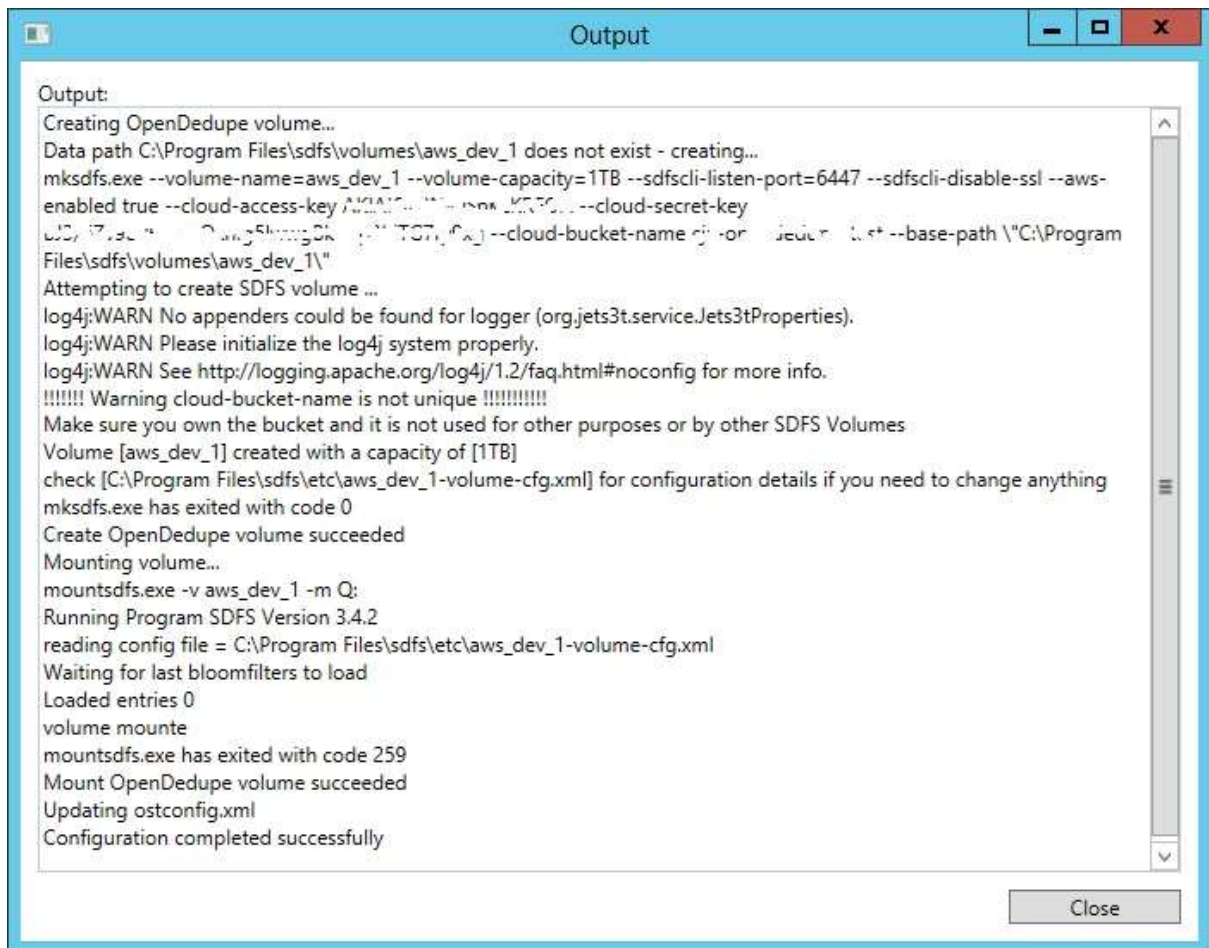
OST Settings

OST server name for Backup Exec: aws_server_1

OK Cancel

3. Select the type of volume to create. The choices are to create a "Local" volume where all the deduplication storage is on a local disk, or to create a volume where the primary storage is in a cloud provider container. The cloud provider choices are "AWS", "Azure", or "Google".
4. Select the volume capacity in units of GB or TB. This is the maximum size to which the volume will grow as more backups are sent to it. The OpenDedupe software also uses this number to determine the amount of RAM to allocate for an in-memory cache of data hash codes. See memory guidelines in the section entitled [Memory Sizing](#).
5. Select an unused drive letter to use for mounting the OpenDedupe volume. Data in the volume is visible through this drive letter.
6. Pick a local disk path where cloud data will be cached. In the case of a Local OpenDedupe volume, this path is where all the deduplicated data is stored. For a volume associated with a cloud provider, it is the location of a data cache, which has a default size of 10 GB. The default path is located in a subfolder of the SDFS installation path.

7. Supply the authentication credentials and container or "bucket" name if the volume is associated with a cloud provider. The captions change to reflect the proper terms for each provider.
8. Supply an OST server name which will be used when configuring a new storage device in Backup Exec
9. Finally, click the "OK" button. The utility shows the output from several OpenDedupe commands.



```
Output:
Creating OpenDedupe volume...
Data path C:\Program Files\sdfs\volumes\aws_dev_1 does not exist - creating...
mksdfs.exe --volume-name=aws_dev_1 --volume-capacity=1TB --sdfscli-listen-port=6447 --sdfscli-disable-ssl --aws-
enabled true --cloud-access-key AKIAJ552... --cloud-secret-key ... --cloud-bucket-name ... --base-path \"C:\Program
Files\sdfs\volumes\aws_dev_1\"
Attempting to create SDFS volume ...
log4j:WARN No appenders could be found for logger (org.jets3t.service.Jets3tProperties).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
!!!!!! Warning cloud-bucket-name is not unique !!!!!!!!!
Make sure you own the bucket and it is not used for other purposes or by other SDFS Volumes
Volume [aws_dev_1] created with a capacity of [1TB]
check [C:\Program Files\sdfs\etc\aws_dev_1-volume-cfg.xml] for configuration details if you need to change anything
mksdfs.exe has exited with code 0
Create OpenDedupe volume succeeded
Mounting volume...
mountsdfs.exe -v aws_dev_1 -m Q:
Running Program SDFS Version 3.4.2
reading config file = C:\Program Files\sdfs\etc\aws_dev_1-volume-cfg.xml
Waiting for last bloomfilters to load
Loaded entries 0
volume mounte
mountsdfs.exe has exited with code 259
Mount OpenDedupe volume succeeded
Updating ostconfig.xml
Configuration completed successfully
```

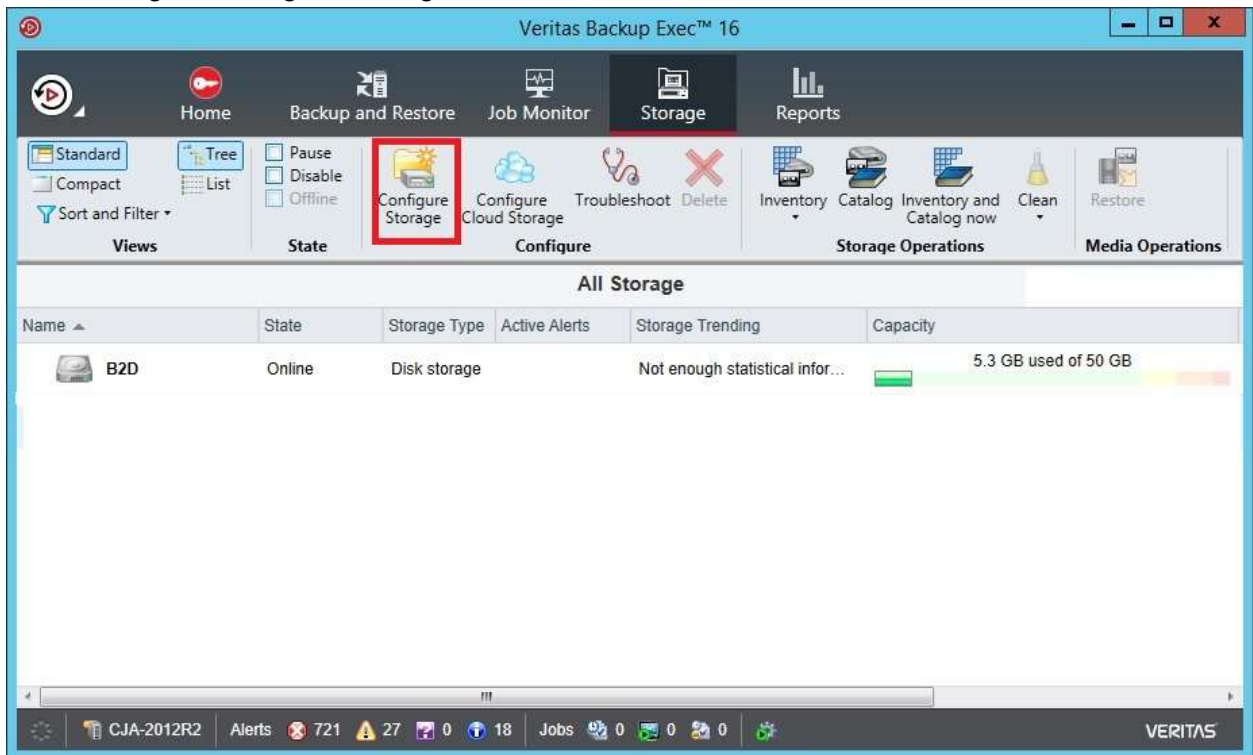
You may repeat this process to add more OpenDedupe volumes.

Veritas Backup Exec™

Installation and Configuration Guide for OpenDedupe OST Connector

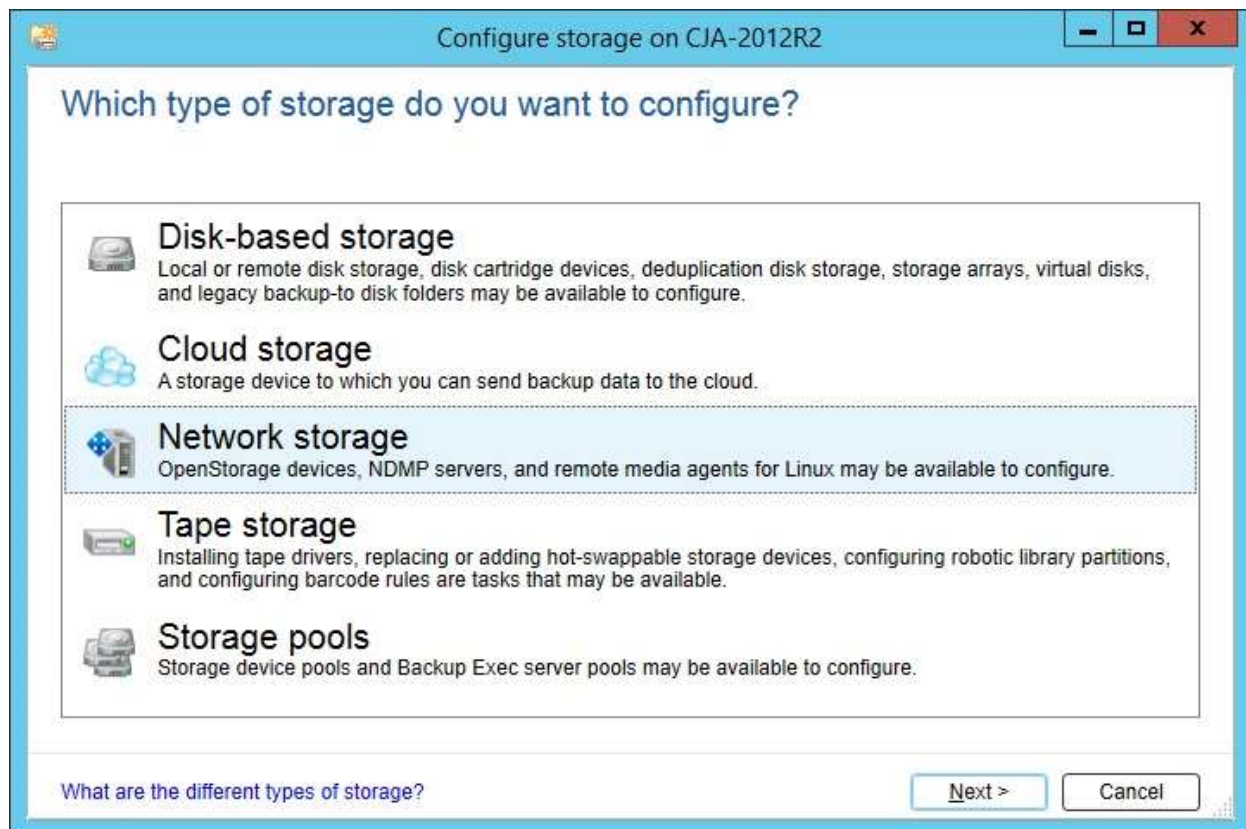
Create an OST Storage Device using the Backup Exec Console

1. Start the Backup Exec Console.
2. Click Storage > Configure Storage.

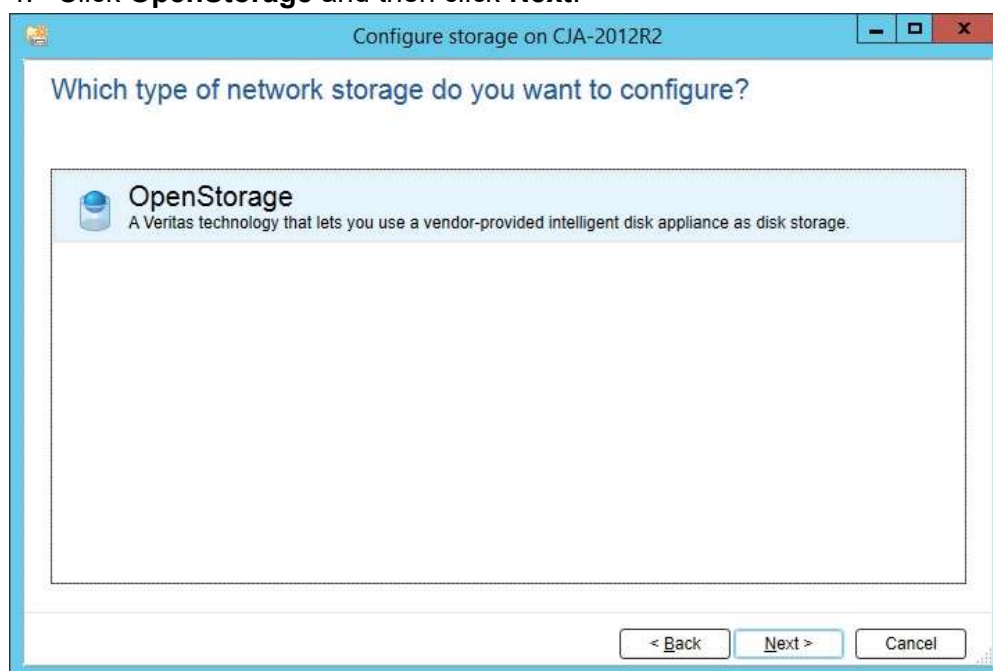


Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

3. Click **Network storage** and then click **Next**.

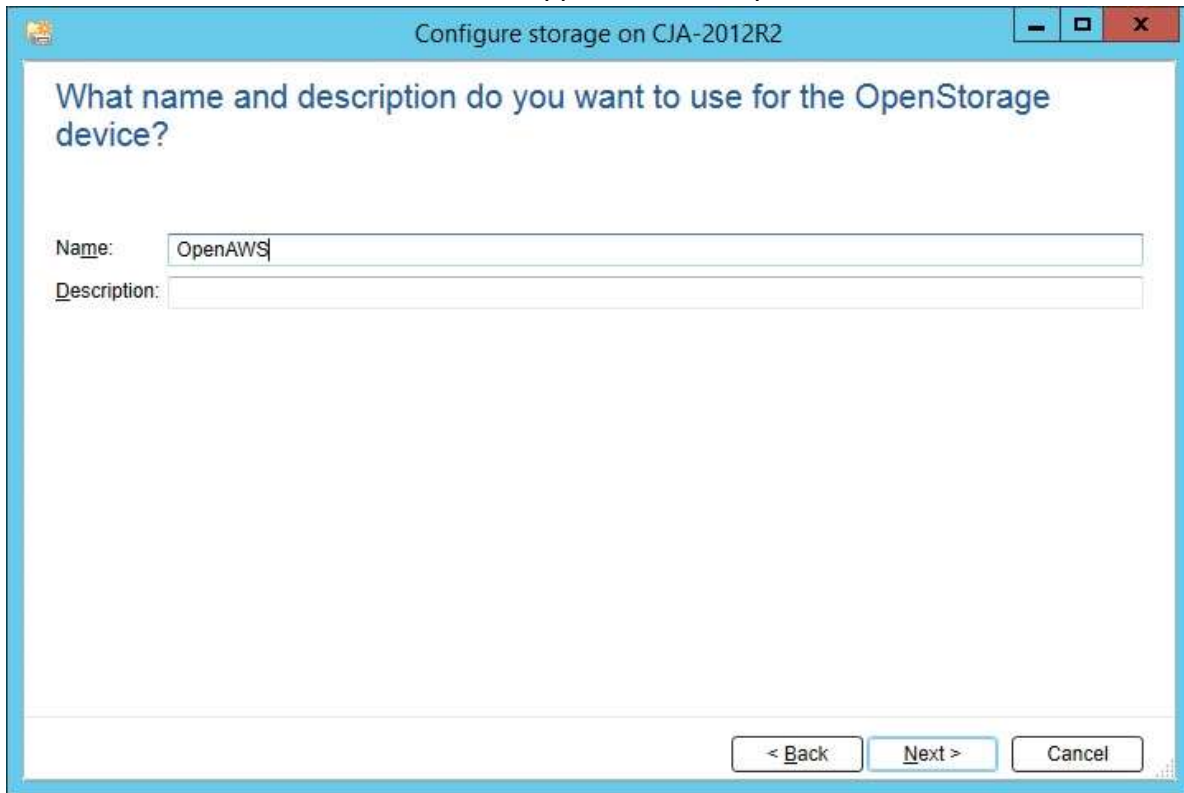


4. Click **OpenStorage** and then click **Next**.



Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

5. Enter a name for the device. This name appears in Backup Exec



Configure storage on CJA-2012R2

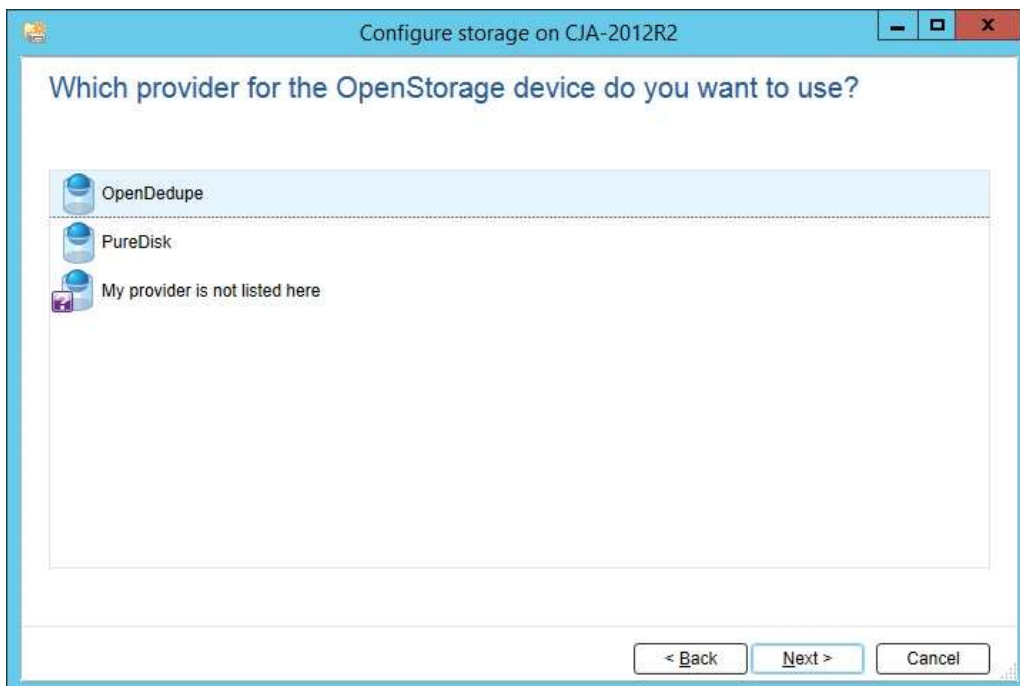
What name and description do you want to use for the OpenStorage device?

Name: OpenAWS

Description:

< Back Next > Cancel

6. Click **OpenDedupe** as the OpenStorage provider and then click **Next**.



Configure storage on CJA-2012R2

Which provider for the OpenStorage device do you want to use?

OpenDedupe

PureDisk

My provider is not listed here

< Back Next > Cancel

Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

7. Specify the connection information including the OST server name that is configured in the SDFS ostconfig.xml file and then click **Next**.

Note: The **Logon account** information is not used and can be set to any Backup Exec logon account.

Configure storage on CJA-2012R2

What is the connection information for the OpenStorage device?

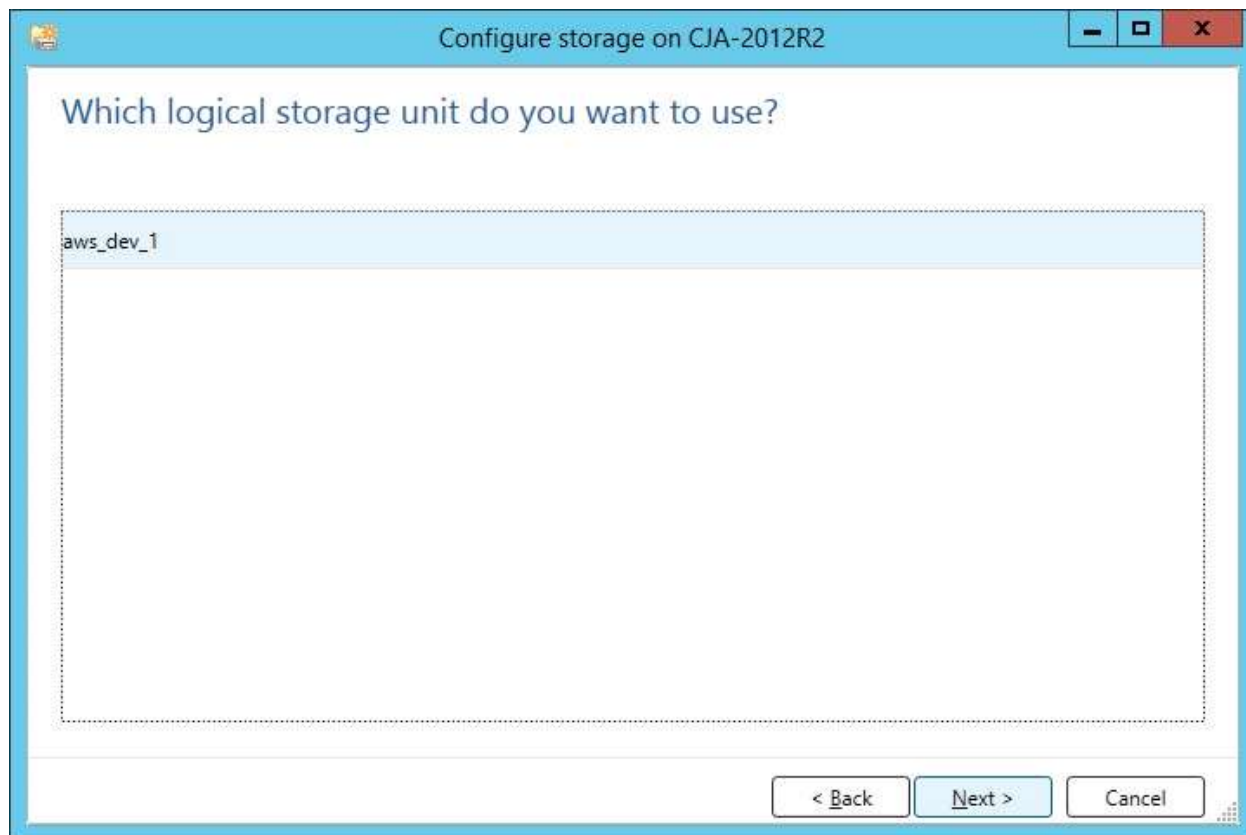
Server name:
Use the host name or fully qualified domain name of the server.

Logon account:

Veritas recommends that you select or create a logon account that is used exclusively for the OpenStorage device. The account should not be used for any other purpose and should not contain credentials that are subject to password update policies.

Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

8. Select the LSU name that is configured in ostconfig.xml for this device and then click **Next**.



9. Type or select the device concurrency (number of simultaneous jobs the device can handle) and then click **Next**.

Configure storage on CJA-2012R2

How many concurrent operations do you want to let run at the same time on this OpenStorage device?

Concurrent operations:

This setting determines how many jobs can run at the same time on this device. The number of jobs that this device can handle varies depending on your hardware and environment, so you may need to adjust this setting later. Veritas recommends that you set it low enough to avoid overloading your system, but high enough to process your jobs in a timely manner.

< Back Next > Cancel

Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

10. View the summary for the device configuration and then click **Finish**.

The screenshot shows a Windows-style window titled "Configure storage on CJA-2012R2". Inside, there's a section titled "Storage configuration summary". This section contains several expandable/collapsible items, each with a double-up arrow icon on the right. The items and their values are:

- Storage category**: Network storage
- Storage type**: OpenStorage
- Name and description for the OpenStorage device**:
 - Name: OpenAWS
 - Description:
- Provider name**:
 - OpenStorage provider name: OpenDedupe
- Connection information**:
 - Server name: aws_server_1
 - Logon account: dedupe
- Logical storage unit**:
 - Storage location: aws_dev_1

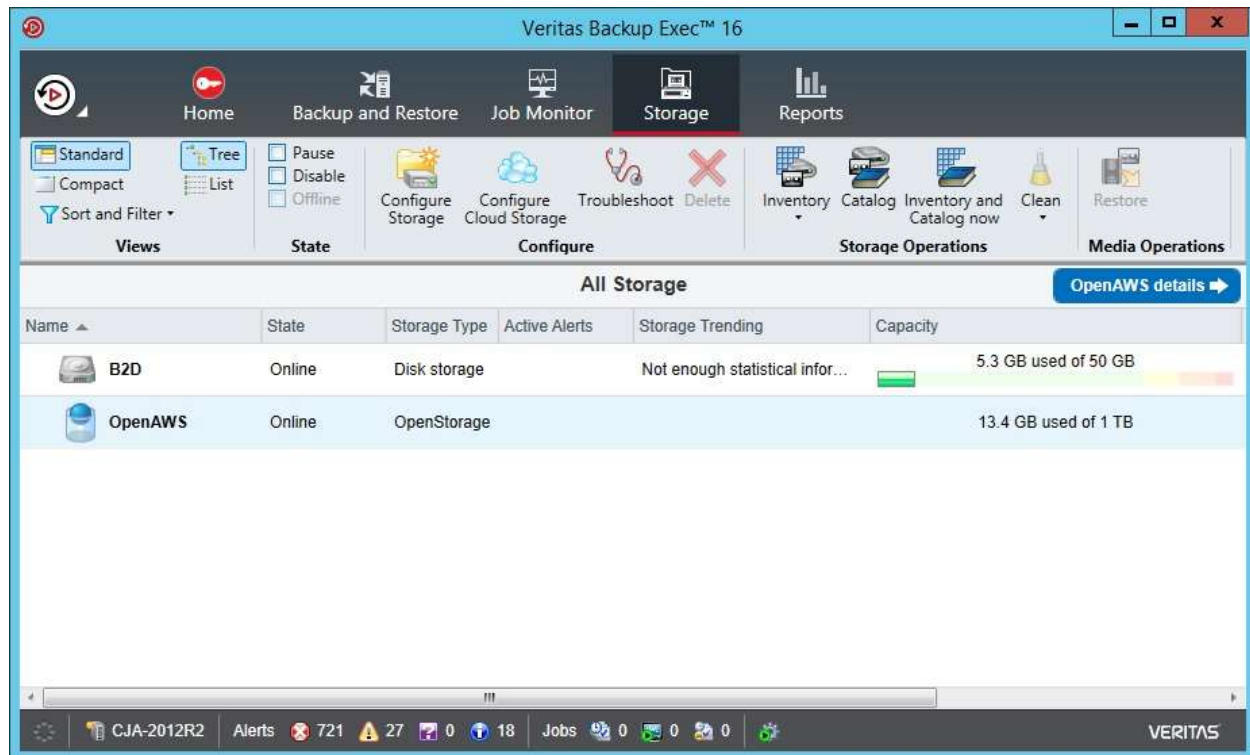
At the bottom right of the window, there are three buttons: "< Back", "Finish", and "Cancel".

10. Restart the Backup Exec services.

Veritas Backup Exec™

Installation and Configuration Guide for OpenDedupe OST Connector

11. After Backup Exec services restart, the new device is displayed in the list of available backup devices.

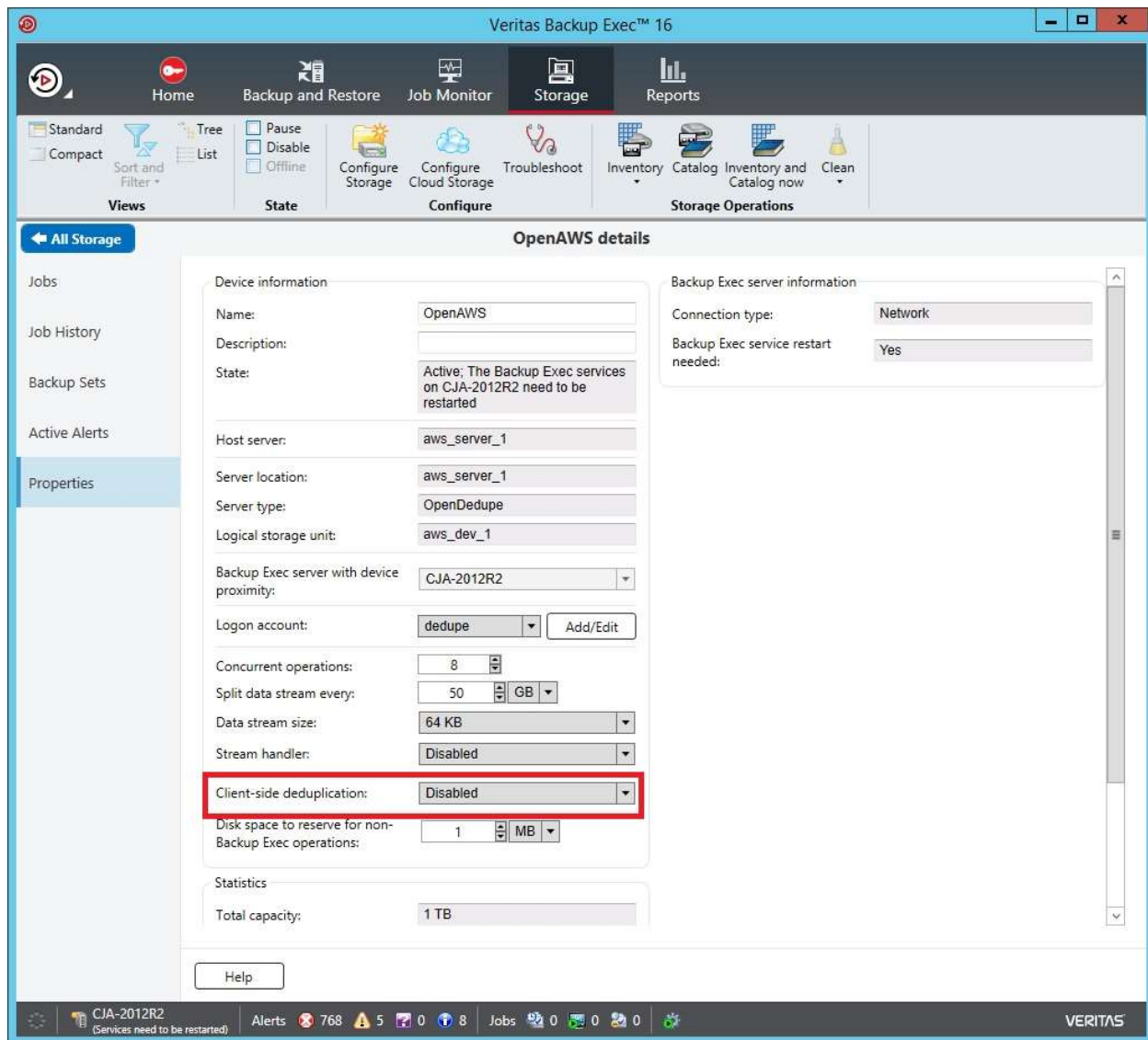


Veritas Backup Exec™

Installation and Configuration Guide for OpenDedupe OST Connector

12. Disable the client-side deduplication for OpenDedupe devices to suppress Backup Exec job exception messages.

Note: OpenDedupe devices do not support client-side deduplication.

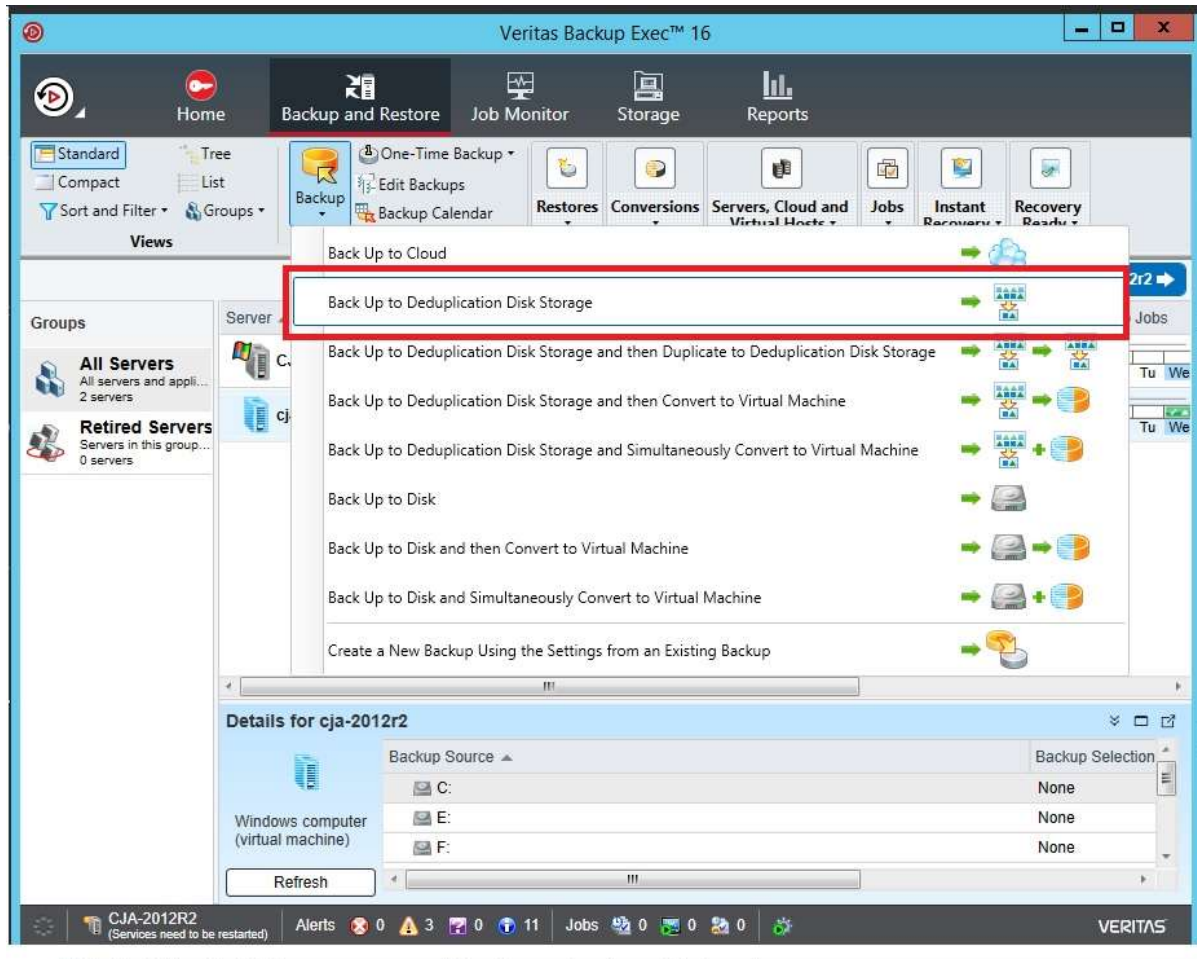


Veritas Backup Exec™

Installation and Configuration Guide for OpenDedupe OST Connector

Creating a backup job using an OpenDedupe device

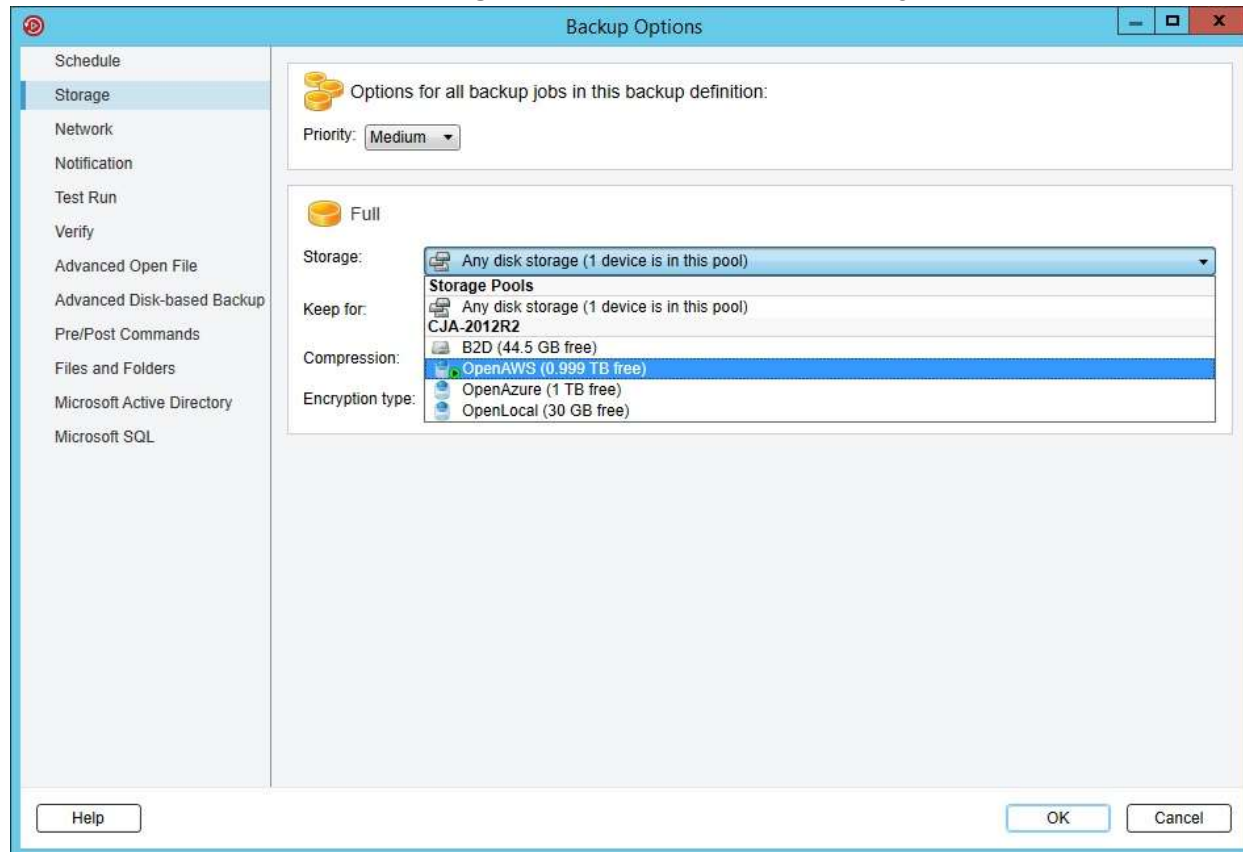
1. Start the Backup Exec Console.
2. Click Backup > Backup To Deduplication Disk Storage.



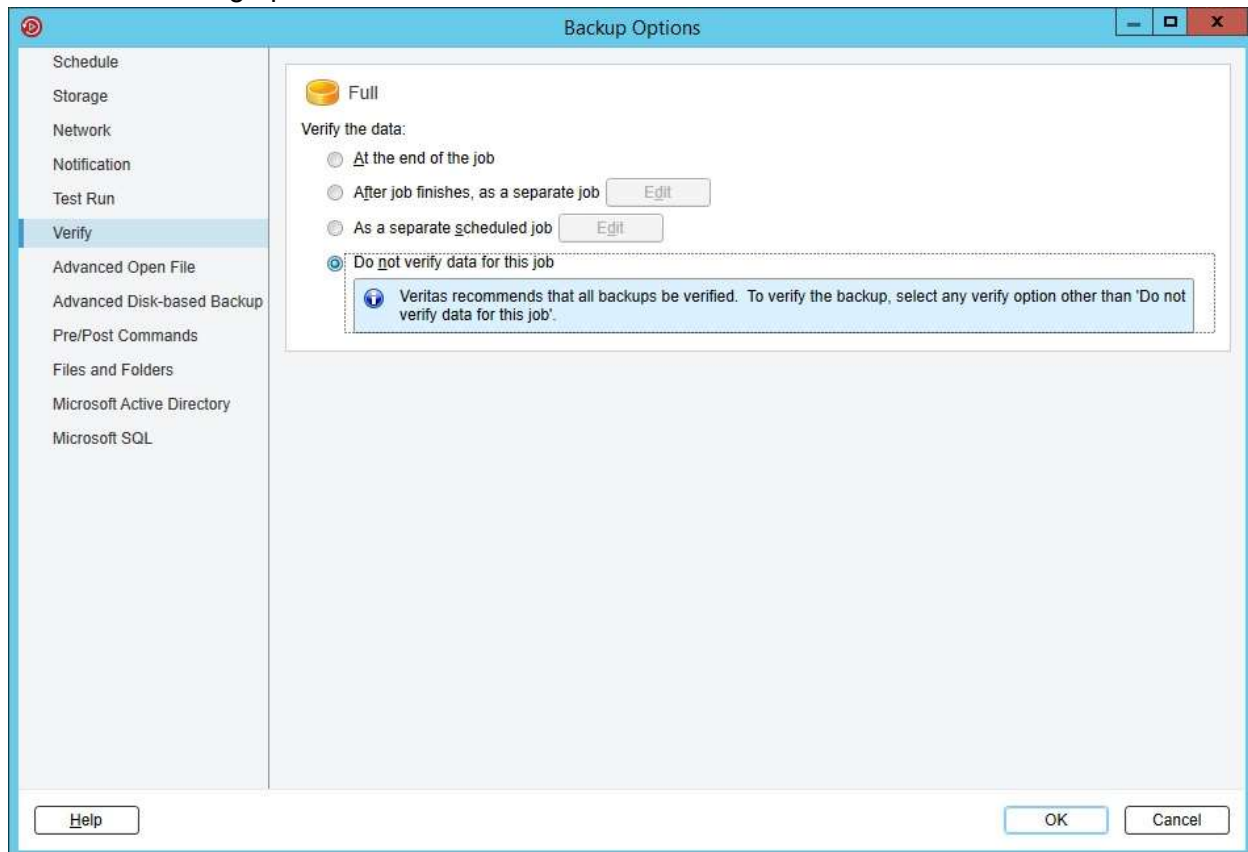
Veritas Backup Exec™

Installation and Configuration Guide for OpenDedupe OST Connector

3. In **Backup Options > Storage**, select an OpenDedupe storage device.



4. For backup jobs to cloud destination, in **Backup Options > Verify**, click **Do not verify data for this job**, as a verify job pulls data back from the cloud, which can be an expensive and time-consuming operation.



Scaling and Sizing

When running OpenDedupe at scale, disk, CPU and memory requirements need to be considered to size appropriately.

Data Stored on Disk

- **Cloud Volumes** - SDFS Stores file metadata, a local hashtable, and a cache of unique blocks on the local disk.
- **Local Volumes** - SDFS Stores file metadata, a local hashtable, and all unique blocks on the local disk.

Data Types:

File MetaData - Information about files and folders stored on openedupe volumes. This data is also stored in the cloud for DR purposes when using cloud storage. File MetaData represents 2.1% of the non deduplicated size of the file stored.

HashTable - The hashtable is the lookup table that is used to identify whether incoming data is unique. The hashtable is stored on the local disk and in the cloud for object storage backed instances. For local instances the hashtable is stored on the local disk only. The hashtable is .4% of the unique storage size.

Local Cache - For Object storage backed volumes, active data is cached locally. The local cache stores compressed, deduplicate blocks only. This local cache size is set to 10GB by default but can be set to any capacity required with a minimum of 1GB. The local cache helps with restore performance and accelerated backup performance.

Local Unique Data Store - OpenDedupe stores all unique blocks locally for volumes not backed by object storage. For Object storage backed volumes, this is not used. Local storage size depends on the data being backed up and retention, but typically represents 100% of the front end data for a 60 Day retention. OpenDedupe uses a similar variable block deduplication method to a DataDomain so it is comparable to its sizing requirements.

Storage Performance:

Minimum local disk storage performance:

- 2000 random read IOPS
- 2400 random write IOPS
- 180 MB/s of streaming reads
- 120 MB/s of streaming writes

Storage Requirements:

The following percentages should be used to calculate local storage requirements for Object Backed dedupe Volumes:

- Metadata: 2.1% of Non-Deduped Data Stored
- Local Cache: 10GB by default
- HashTable: .2% of Deduped Data

An example for 100TB of deduped data with an 8:1 dedupe rate would be as follows:

- Logical Data Stored on Disk = $8 \times 100\text{TB} = 800\text{TB}$
- Local Cache = 10GB
- Unique Data Stored in the Object Store 100TB
- Metadata
 - $2.1\% \text{ Logical Data Stored on Disk} = \text{Metadata Size}$
 - $.021 \times 800\text{TB} = 1.68\text{TB}$
- HashTable
 - $.2\% * \text{Unique Storage}$
 - $.002 * 100\text{TB} = 400\text{GB}$
- Total Volume Storage Requirements
 - Local Cache + Metadata + Hashtable
 - $10\text{GB} + 1.68\text{TB} + 400\text{GB} = 2.09\text{TB}$

The following percentages should be used to calculate local storage requirements for local dedupe Volumes:

- Metadata: 2.1% of Non-Deduped Data Stored
- Local Cache: 10GB by default
- HashTable: .2% of Deduped Data
- Unique Data

An example for 100TB of deduped data with an 8:1 dedupe rate would be as follows:

- Logical Data Stored on Disk = $8 \times 100\text{TB} = 800\text{TB}$
- Unique Data Stored on disk 100TB
- Metadata
 - $21\% \text{ Logical Data Stored on Disk} = \text{Metadata Size}$
 - $.021 \times 800\text{TB} = 1.68\text{TB}$
- HashTable
 - $.2\% * \text{Unique Storage}$
 - $.002 * 100\text{TB} = 400\text{GB}$
- Total Volume Storage Requirements
 - Unique + Metadata + Hashtable +
 - $100\text{TB} + 1.68\text{TB} + 400\text{GB} = 102.08\text{TB}$

Memory Sizing:

Memory for OpenDedupe is primarily used for internal simplified lookup tables (bloom filter) that indicate, with some likelihood that a hash is already stored or not. These data structures take about 256MB per TB of data stored. 1GB additional base memory is required for other uses.

In addition to memory used by OpenDedupe you want memory available for file system cache to cache the most active parts of the lookup hashtable into RAM. For a volume less than 1TB you require an additional 1GB RAM. For a volume less than 100TB you require an additional 8GB RAM. For a volume over 100TB you require additional 16GB RAM.

An example for 100TB of deduped data:

- Hash Table Memory
 - 256MB per 1TB of Storage
 - 256MB x 100TB = 25.6 GB
- 1GB base memory
- 8GB free RAM for Disk Cache
- Total = 25.6+1+8=34.6GB RAM

CPU Sizing:

As long as the disk meets minimum IO and IOPs requirements, the primary limiter for OpenDedupe performance is CPU at higher dedupe rates. At lower dedupe rates volumes are limited by the speed of the underlying disk.

For a single 16 Core CPU, SDFS performs at

- 2GB/s for 2% Unique Data
- Speed of local disk for 100% unique data. Using minimum requirements this would equal 120MB/s.

Troubleshooting (Logs)

For help in troubleshooting issues associated with SDFS of the OST plugin, you can send an email to the sdfs forum.

<http://groups.google.com/group/dedupfilesystem-sdfs-user-discuss?pli=1>

SDFS Logs:

SDFS creates logs under **C:\Program Files\sdfs\logs\<volume-name>-volume-cfg.xml.log**. Errors can be identified in this log file.

OST Plugin Logs:

The OpenDedupe OST plugin log can be found in the path specified by the FILENAME parameter in **C:\Program Files\sdfs\etc\ost-logging-global.conf**.

Using the command line to set up SDFS volumes on a Backup Exec Server

Instead of using the OpenDedupeConfigUI tool described earlier to set up OpenDedupe volumes and OST connection information, you can configure those manually using command-line tools. This section guides you through that process.

1. Create an SDFS volume, running under the Backup Exec Administrator account. Select a volume name that will be used as an OST LSU name.

For example, for AWS storage, you can use “aws_dev_1”

The following are command-line examples for setting up various kinds of OpenDedupe storage. Due to the line wrapping in this document, each command example spans several lines. However, when you run a command, it is meant to be on a single line. Lines in this document that appear to end with the ‘-’ (hyphen) character are meant to be adjacent to the word on the next line. For example, the section that looks like this:

```
--azure-  
enabled true
```

is meant to be this:

```
--azure-enabled true
```

Text surrounded with “[]” (square brackets) signifies optional arguments. Text surrounded with “<>” (angle brackets) means you should fill in your own value (without the brackets). For example:

```
--cloud-access-key <access-key>
```

Should be something like this when you enter the command line:

```
--cloud-access-key ABCDEFGHIJKL
```

Local Storage

```
mksdfs --nossf --volume-name=local_dev_1 --volume-capacity=256GB --nossf [--base-path <PATH for data>]
```

AWS Storage

```
mksdfs --nossf --volume-name=aws_dev_1 --volume-capacity=1TB --aws-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> [--base-path <PATH for cached data>]
```

Azure Storage

```
mksdfs --nossf --volume-name=azure_dev_1 --volume-capacity=1TB --azure-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> [--base-path <PATH for cached data>]
```

Google Storage

```
mksdfs --nossf --volume-name=google_dev_1 --volume-capacity=1TB --google-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> [-base-path <PATH for cached data>]
```

The "--base-path" parameter is optional. If you specify a base path for a cloud storage device, it is a path where all locally cached data is stored. The default size for the local cache is 10 GB. The default base path is "C:\Program Files\sdfs\volumes\volume-name".

In the case of local OpenDedupe storage, the "--base-path" value is the path where all deduplicated data is stored. The required storage size is the value of the "--volume-capacity" parameter.

When you run the "mksdfs" command, you may see a warning similar to the following:

```
log4j:WARN No appenders could be found for logger (org.jets3t.service.Jets3tProperties).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

You may safely ignore the warning.

2. Mount the SDFS Volume with an unused drive letter (for example Q:).

```
mountsdfs -v aws_dev_1 -m Q:
```

3. Create or edit C:\Program Files\SDFS\etc\ostconfig.xml to add information about each SDFS volume you create so it can be used as an OST device in Backup Exec.

Detailed OST XML Configuration setup

The OST driver is configured from the XML file located at:

C:\Program Files\sdfs\etc\ostconfig.xml. It contains all the configuration parameters required to allow SDFS to communicate with Backup Exec over OST.

Below is a typical ostconfig.xml :

```
<!-- This is the config file for the OST connector for openedup and Backup Exec -->
<CONNECTIONS>
<CONNECTION>
<!--NAME is the OST server name that you will reference within Backup Exec
It can be any name that is meaningful to you -->
<NAME>
aws_server_1
</NAME>
<!--LSU_NAME is the name of the volume you created with mksdfs -->
<LSU_NAME>
aws_dev_1
</LSU_NAME>
<!--AUTOMOUNT causes the OpenDedupe plugin to mount the volume named by the LSU
If it is not already mounted -->
<AUTOMOUNT>
true
</AUTOMOUNT>
<!--VOLUME_PATH is the drive letter where the SDFS file system will be mounted
Make sure to include the trailing '\' character -->
<VOLUME_PATH>
```



```
Q:\
</VOLUME_PATH>
<!--URL - this is the url that the plugin will use to communicate with sdfs volume.
If the volume is local and the only one url should be set to https://localhost:6442.
-->
<URL>
http://localhost:6442/
</URL>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>passwd</PASSWD>
</CONNECTION>
</CONNECTIONS>
```

Configuring OST Plugin logging

To configure the OST plugin logging, create a file named “**c:\Program Files\sdfs\etc\ost-logging-global.conf**” that has the following lines:

```
## Comment line
*GLOBAL:
  FILENAME = "c:/Program Files/SDFS/logs/OpenDedupe.log"
  FORMAT   = "%datetime | %level| %func | %msg"
  ENABLED   = true
  TO_FILE   = true
  TO_STANDARD_OUTPUT = false
  MILLISECONDS_WIDTH = 6
  PERFORMANCE_TRACKING = false
  MAX_LOG_FILE_SIZE    = 2097152 ## Throw log files away after 2MB
  LOG_FLUSH_THRESHOLD  = 1 ## Flush after every 1 logs
*ERROR:
  FORMAT = "%datetime | %level| %func | %loc | %msg"
*DEBUG:
  ENABLED = false
```

You can specify any path for the FILENAME parameter. The earlier example places the log in the standard Backup Exec location. To collect more detailed OST plugin detail information, you can set the DEBUG ENABLED parameter to “true”.

The OST connector supports multiple SDFS volumes on the same Backup Exec Server but additional steps are required to support this configuration.

Step 1: Follow the [Installing the OST Connector](#) instructions provided in the document on each Backup Exec Server that will use the OST Connector.

Step 2: Run the mksdfs command for each additional SDFS volume. For example:

```
mksdfs --nossli --volume-name=azure_dev_1 --volume-capacity=1TB --azure-enabled true --cloud-access-key <access-key> --
cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name>
```

Step 3: Pick a new free driver letter and mount the new volume. For example:

```
mountsdfs -v azure_dev_1 -m R:
```

Step 4: Mount the new volume and get the control port number of the additional volume. In the example below, aws_dev_1 has a tcp control port of 6442 and azure_dev_1 has a control port of 6443. This information is available in the xml config file created by the mksdfs command.

For example, using the name from step 3, look for a file named *C:\Program Files\sdfs\etc\azure_dev_1-volume-cfg.xml*. Search for the <sdfscli> tag in this file, where you can find a value for “port” that states which port is assigned when the volume was mounted.

```
<sdfscli enable="true" enable-auth="false" listen-address="localhost" port="6443"/>
```

Step 5: Edit the *C:\Program Files\sdfs\etc\ostconfig.xml* and add a new <CONNECTION> tag inside of the <CONNECTIONS> tag for the new volume. In the new <CONNECTION> tag add the port identified in Step 4 to the <URL> tag (<http://localhost:6443/>), add a name that is unique the <NAME> tag and specify the new LSU name in the <LSU_NAME> tag (azure_dev_1), and a new drive letter path in the <VOLUME_PATH> tag. Following is an example of an ostconfig.xml with two volumes.

```
<!-- This is the config file for the OST connector for openedup and Backup Exec -->
<CONNECTIONS>
<CONNECTION>
<!--NAME is the OST server name that you will reference within Backup Exec
    It can be any name that is meaningful to you -->
<NAME>
aws_server_1
</NAME>
<!--LSU_NAME is the name of the volume you created with mksdfs -->
<LSU_NAME>
aws_dev_1
</LSU_NAME>
<!--AUTOMOUNT causes the OpenDedupe plugin to mount the volume named by the LSU
    If it is not already mounted -->
<AUTOMOUNT>
true
</AUTOMOUNT>
<!--VOLUME_PATH is the drive letter where the SDFS file system will be mounted
    Make sure to include the trailing '\ ' character -->
<VOLUME_PATH>
Q:\
</VOLUME_PATH>
<!--URL - this is the url that the plugin will use to communicate with sdfs volume.
    If the volume is local and the only one url should be set to http://localhost:6442.
-->
<URL>
http://localhost:6442/
</URL>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>passwd</PASSWD></CONNECTION>
<!-- Below is the new volume-->
<CONNECTION>
<!--NAME is the OST server name that you will reference within Backup Exec
    It can be any name that is meaningful to you -->
<NAME>
azure_server_1
</NAME>
<!--LSU_NAME is the name of the volume you created with mksdfs -->
<LSU_NAME>
azure_dev_1
</LSU_NAME>
<!-- AUTOMOUNT causes the OpenDedupe plugin to mount the volume named by the LSU
    If it is not already mounted -->
<AUTOMOUNT>
true
```

```
</AUTOMOUNT>
<!--VOLUME_PATH is the drive letter where the SDFS file system will be mounted
Make sure to include the trailing '\ ' character -->
<VOLUME_PATH>
R:\
</VOLUME_PATH>
<!--URL - this is the url that the plugin will use to communicate with sdfs volume.
If the volume is local and the only one url should be set to http://localhost:6442.
-->
<URL>
http://localhost:6443/
</URL>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>passwd</PASSWD>
</CONNECTION>
</CONNECTIONS>
```

Creating a Cloud Backup Container

If you plan to use the OpenDedupe OST connector to back up data to a cloud provider, you must establish an account with the provider if you don't already have one, or get the provider authentication information from your organization. After signing into a provider's web site, you must create a new empty cloud storage container, or "bucket" to hold your backup data. The container should be used to hold backup data from only a single Backup Exec server. If you have already created a cloud container to use with an existing Backup Exec Cloud Storage device, you should create a separate container to hold OpenDedupe data.

The details of creating a cloud container are different for each provider. The following examples show how to create a container for each of the three supported providers.

NOTE: The methods for creating containers may change, and there are many options not shown here. These examples show typical methods for container creation as of June 2017.

Creating an S3 Bucket Using Amazon Web Services (AWS)

For AWS, you create an AWS S3 "bucket" to hold deduplicated backup data.

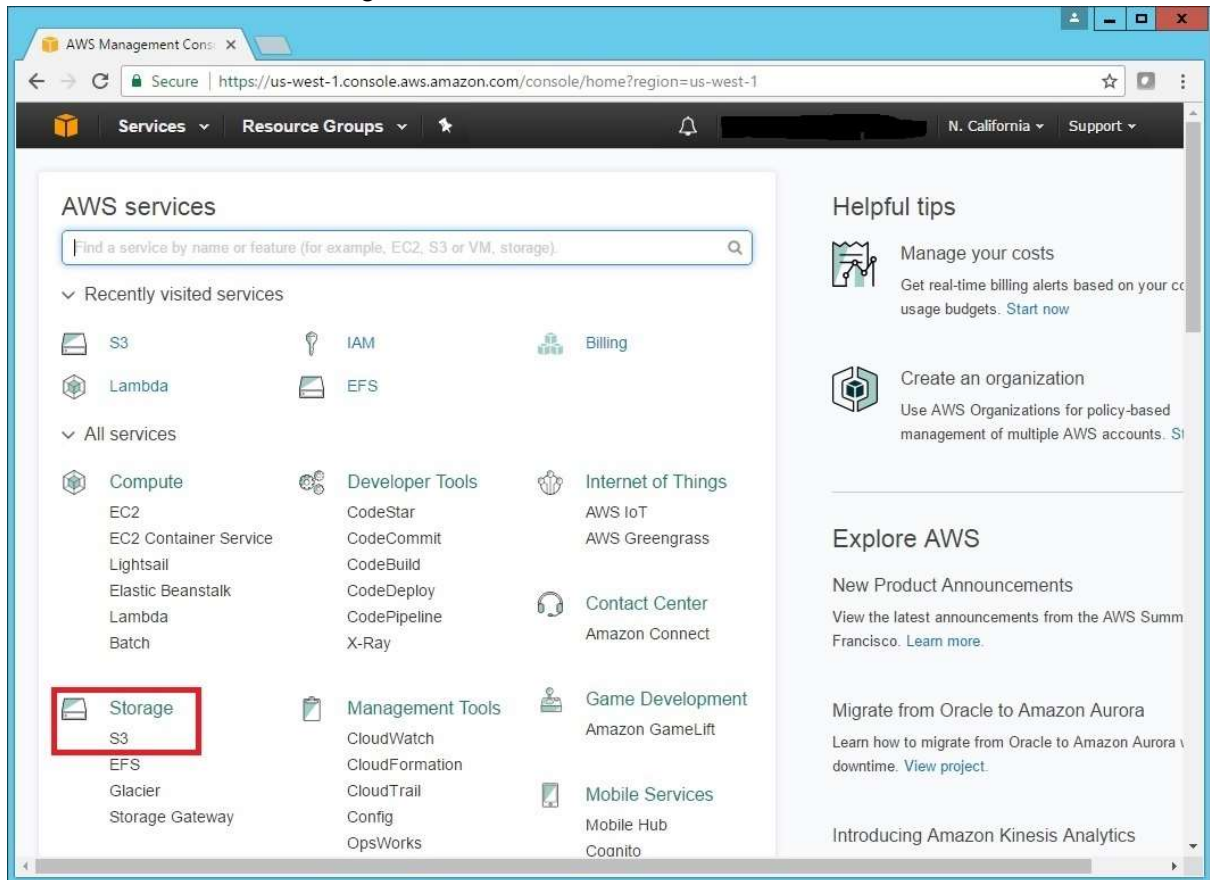
1. Use a web browser to go to <https://aws.amazon.com> and sign into the console using your Amazon AWS credentials.

NOTE: For AWS, you must remember the "Access Key" and "Secret Key" that were assigned to you when your account was created. You cannot retrieve them using the AWS console. They will be needed later when you configure an OpenDedupe device in Backup Exec.

Veritas Backup Exec™

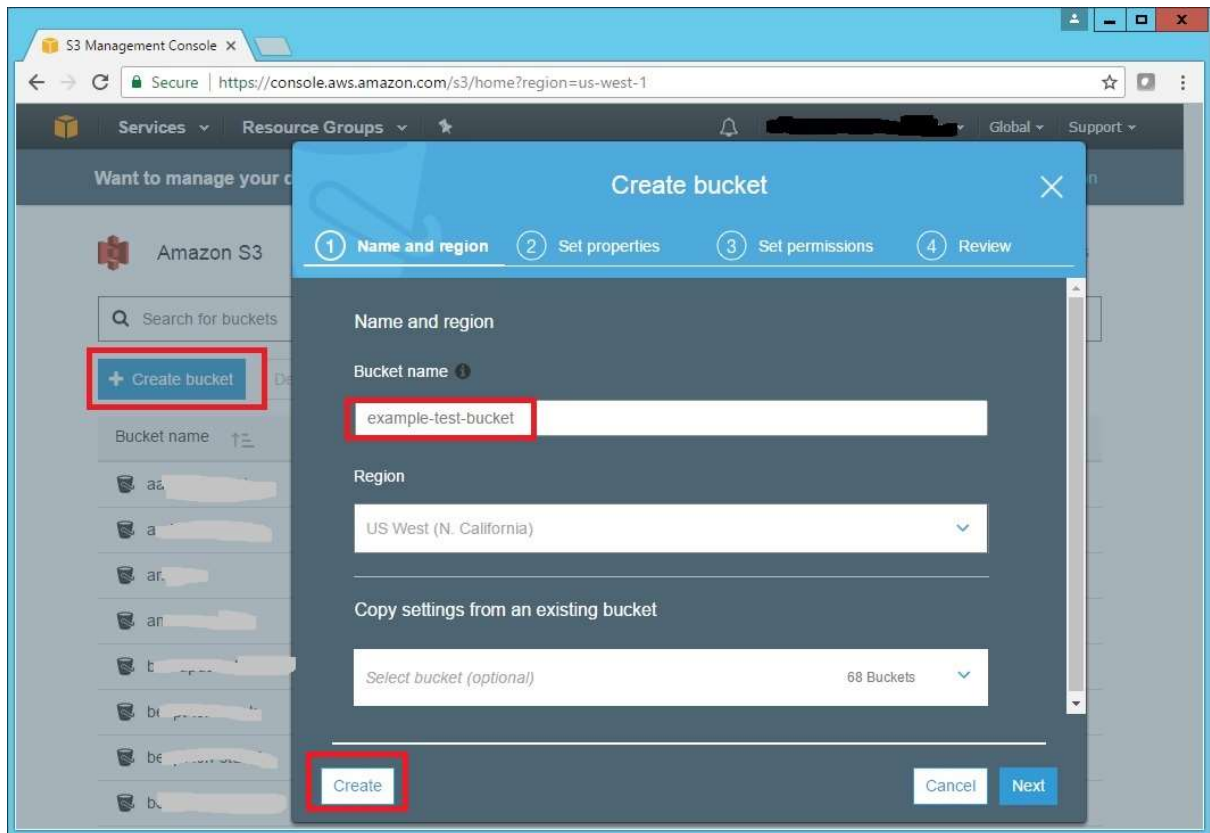
Installation and Configuration Guide for OpenDedupe OST Connector

2. Under AWS Services Storage, select S3



Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

3. Create a new bucket. The bucket name will be used later when you configure OpenDedupe. This bucket will contain deduplicated data written by Backup Exec using the OpenDedupe OST connector.

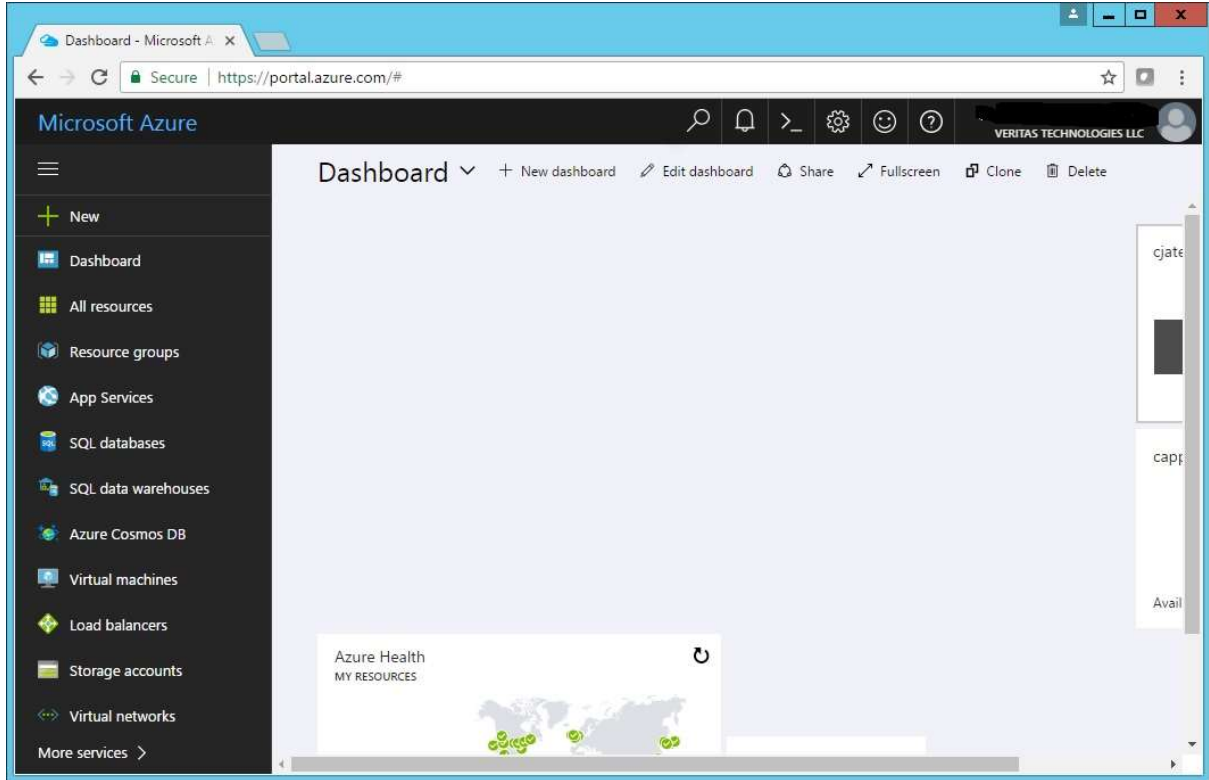


Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

Creating a Blob Container Using Azure

With Azure, you use a simple "blob" storage container to hold backup data.

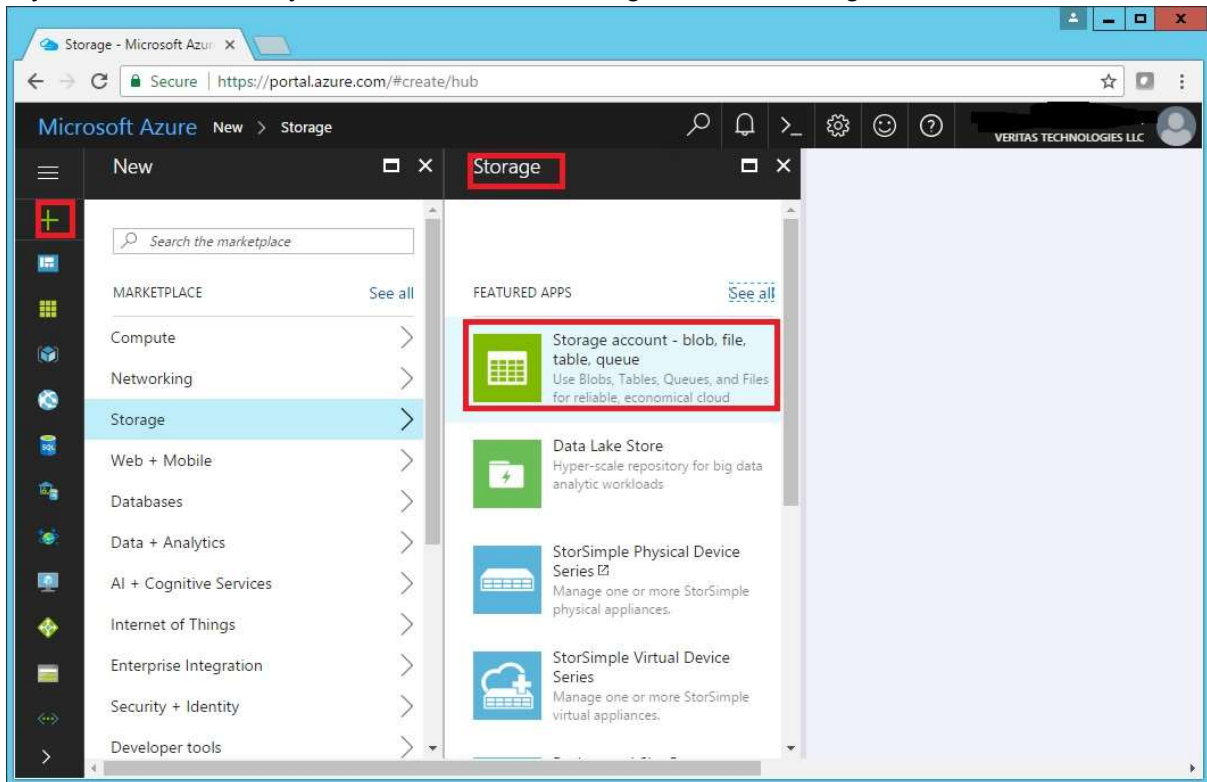
1. Login to the Azure portal at <https://portal.azure.com> using your Azure account information



Veritas Backup Exec™

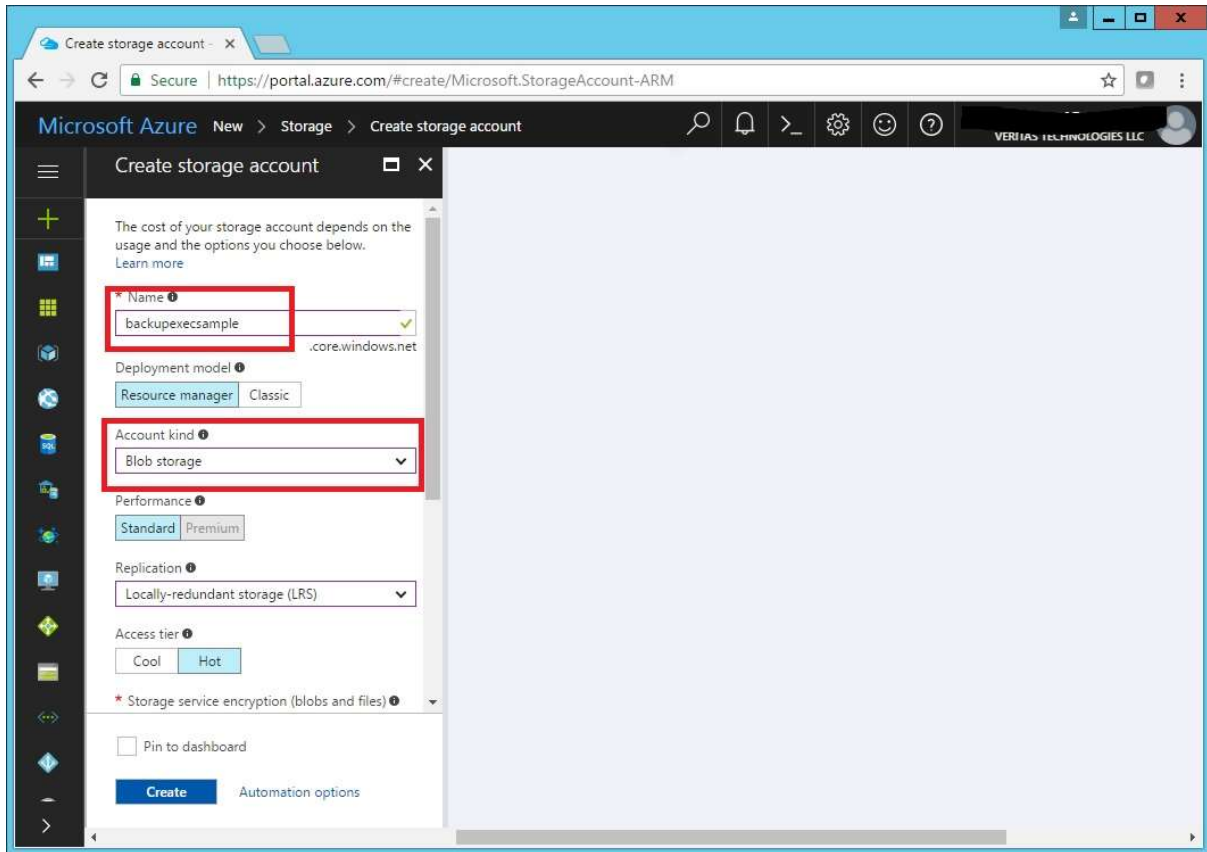
Installation and Configuration Guide for OpenDedupe OST Connector

2. If you have not already done so, create a "Storage Account" using the "+New" button



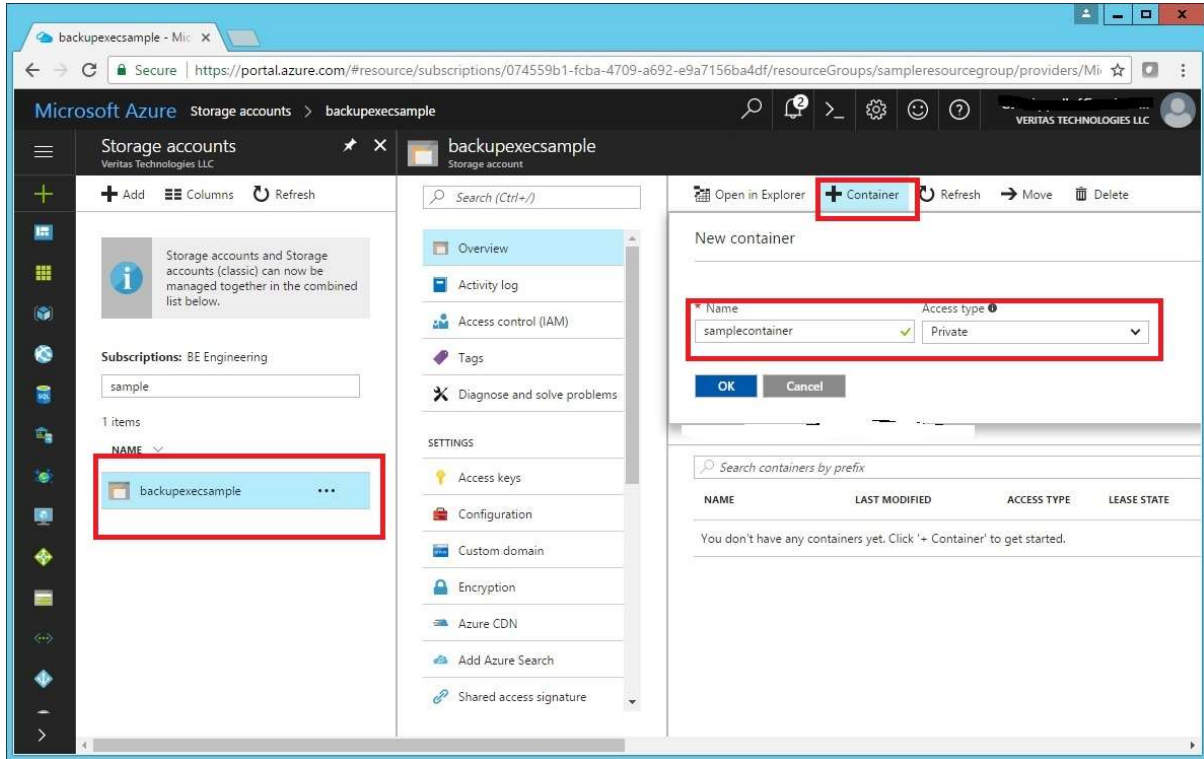
Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

3. You can choose a "General Purpose" storage account type or "Blob Storage" account type. The OpenDedupe connector requires only blob storage.



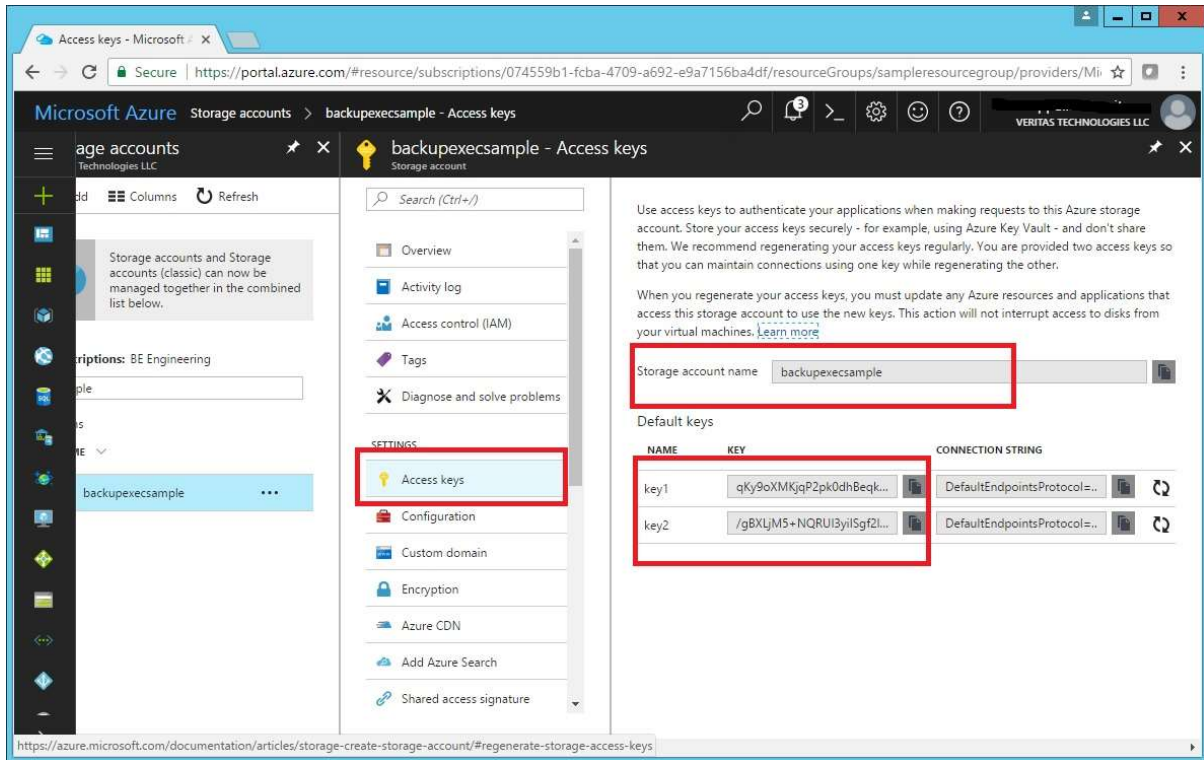
Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

4. Select your storage account and create a new container. You should select access type "Private" for maximum security, but the "Blob" or "Container" access types work with OpenDedupe too.



Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

5. You can use the "Access keys" button to retrieve the storage account name and access keys that are needed to configure an OpenDedupe device later. Either key will work for OpenDedupe.

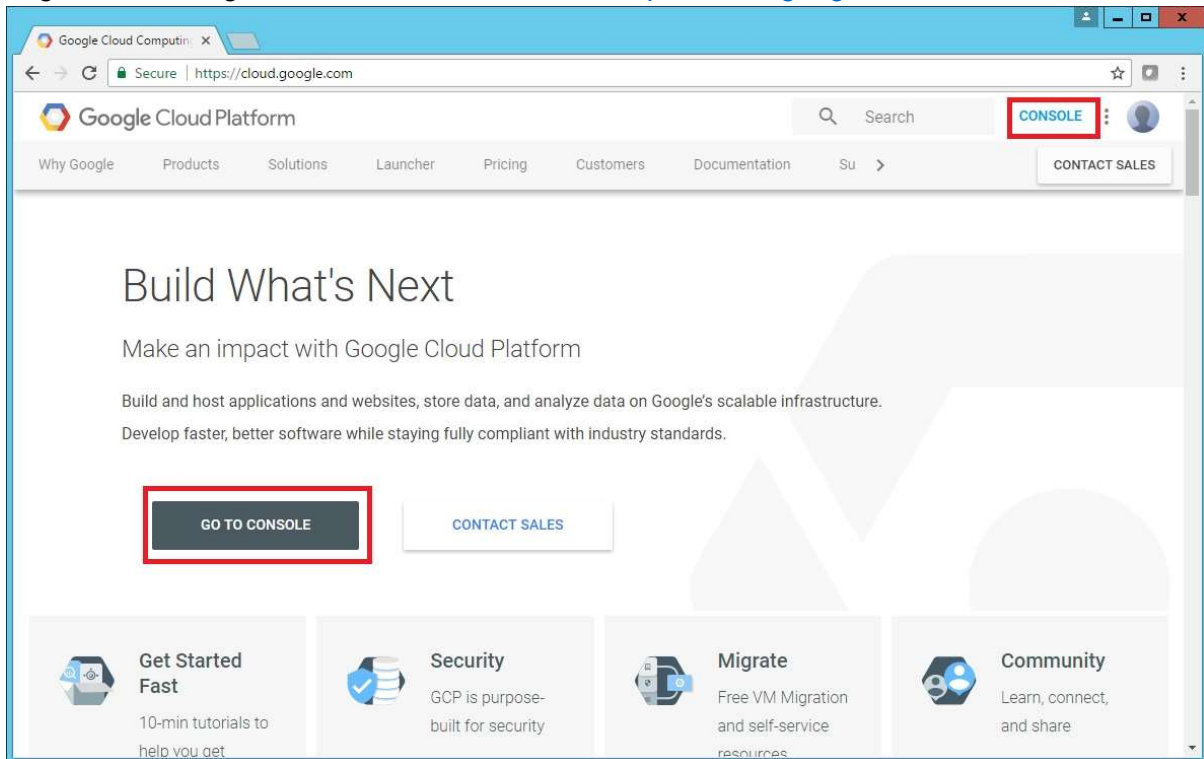


Creating a Storage Bucket Using the Google Cloud Platform

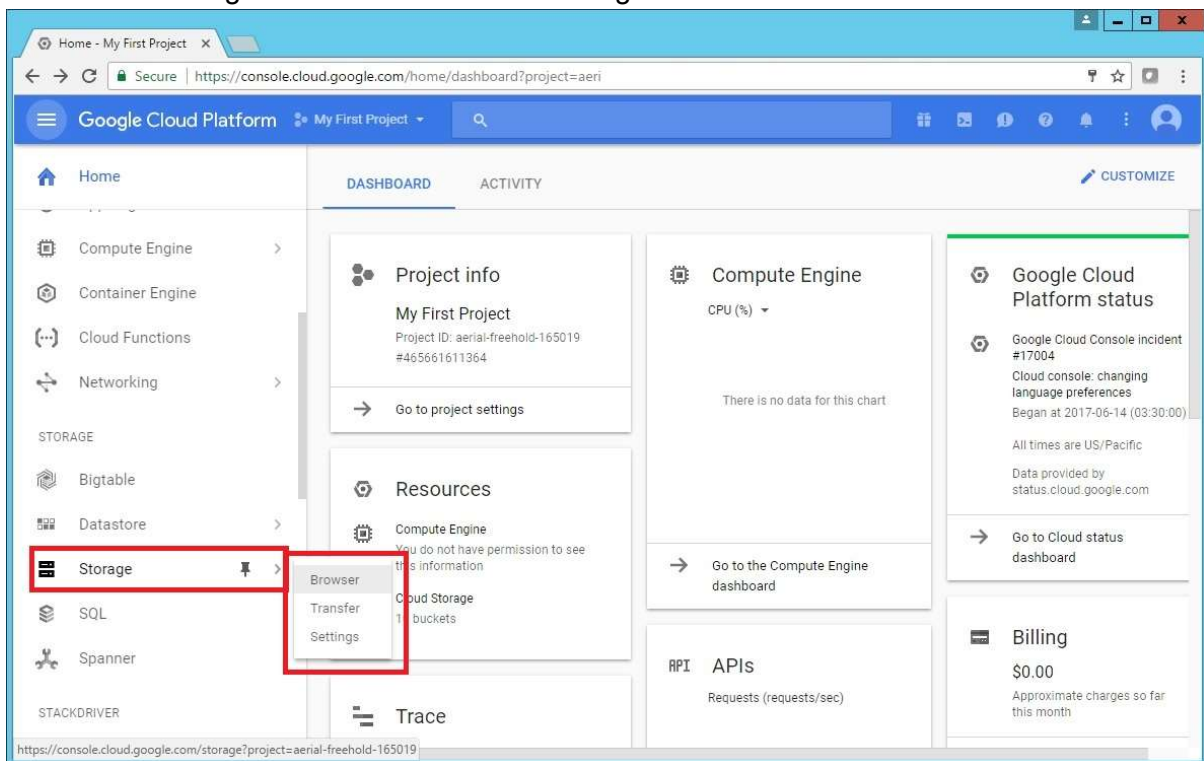
With the Google Cloud Platform, you create a "storage bucket" to hold backup data for OpenDedupe.

Veritas Backup Exec™ Installation and Configuration Guide for OpenDedupe OST Connector

1. Login to the Google Cloud Platform console at <https://cloud.google.com>:



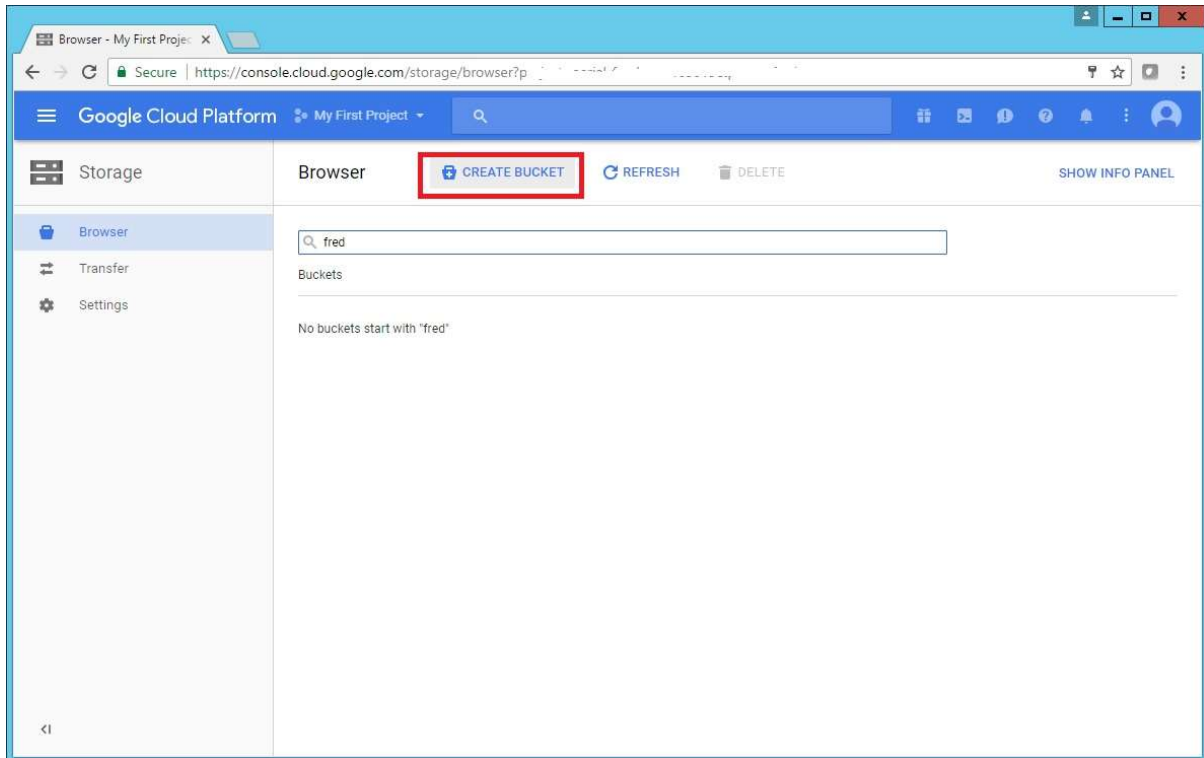
2. Select the "Storage" menu item on the left and go to "Browser"



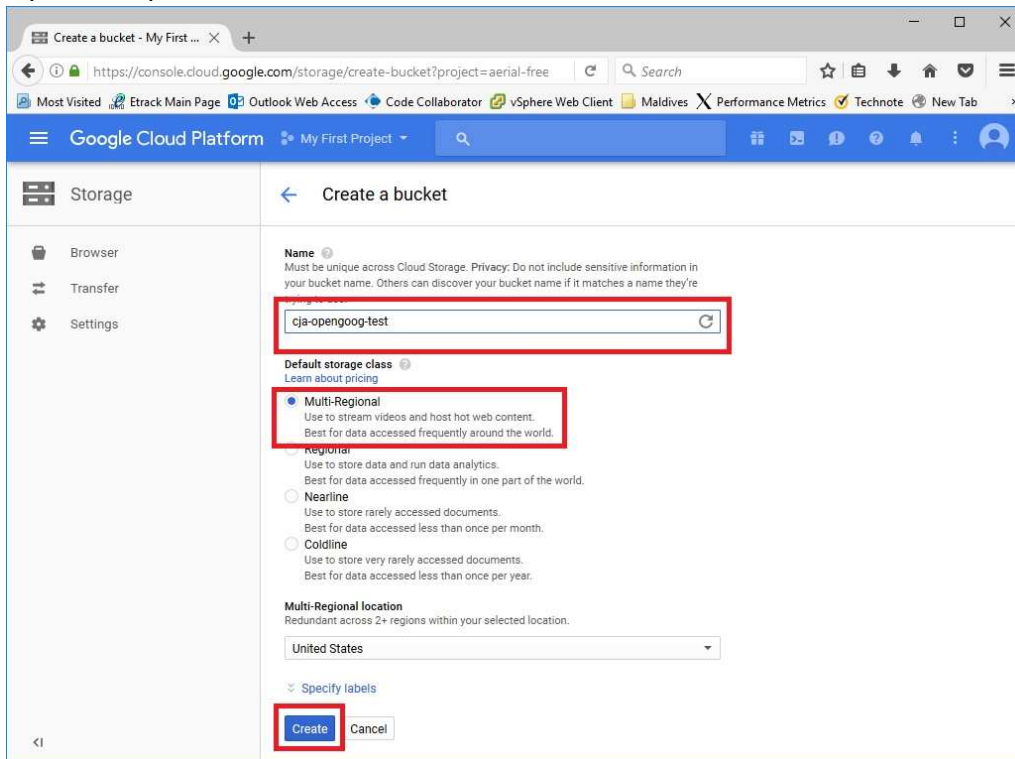
Veritas Backup Exec™

Installation and Configuration Guide for OpenDedupe OST Connector

3. In the Storage Browser window, select "Create Bucket" to create a new storage bucket for backups



4. Pick a bucket name and storage class. "Multi-Regional" or "Regional" work well for OpenDedupe data.



Veritas Backup Exec™

Installation and Configuration Guide for OpenDedupe OST Connector

5. Select "Settings" and "Interoperability" to retrieve your storage access keys. Save them to use for configuring an OpenDedupe device.

