Veritas Cloud Storage Deployment Guide

Linux

1.0.2

VERITAS™

Veritas Cloud Storage Deployment Guide

Last updated: 2018-03-26

Document version: 1.0.2 Rev 1

Legal Notice

Veritas Technologies LLC
500 E Middlefield Road
Mountain View, CA 94043

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 document version appears on page 2 of each guide. 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:

doc.feedback@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

Section 1

Overview and planning

Chapter 1

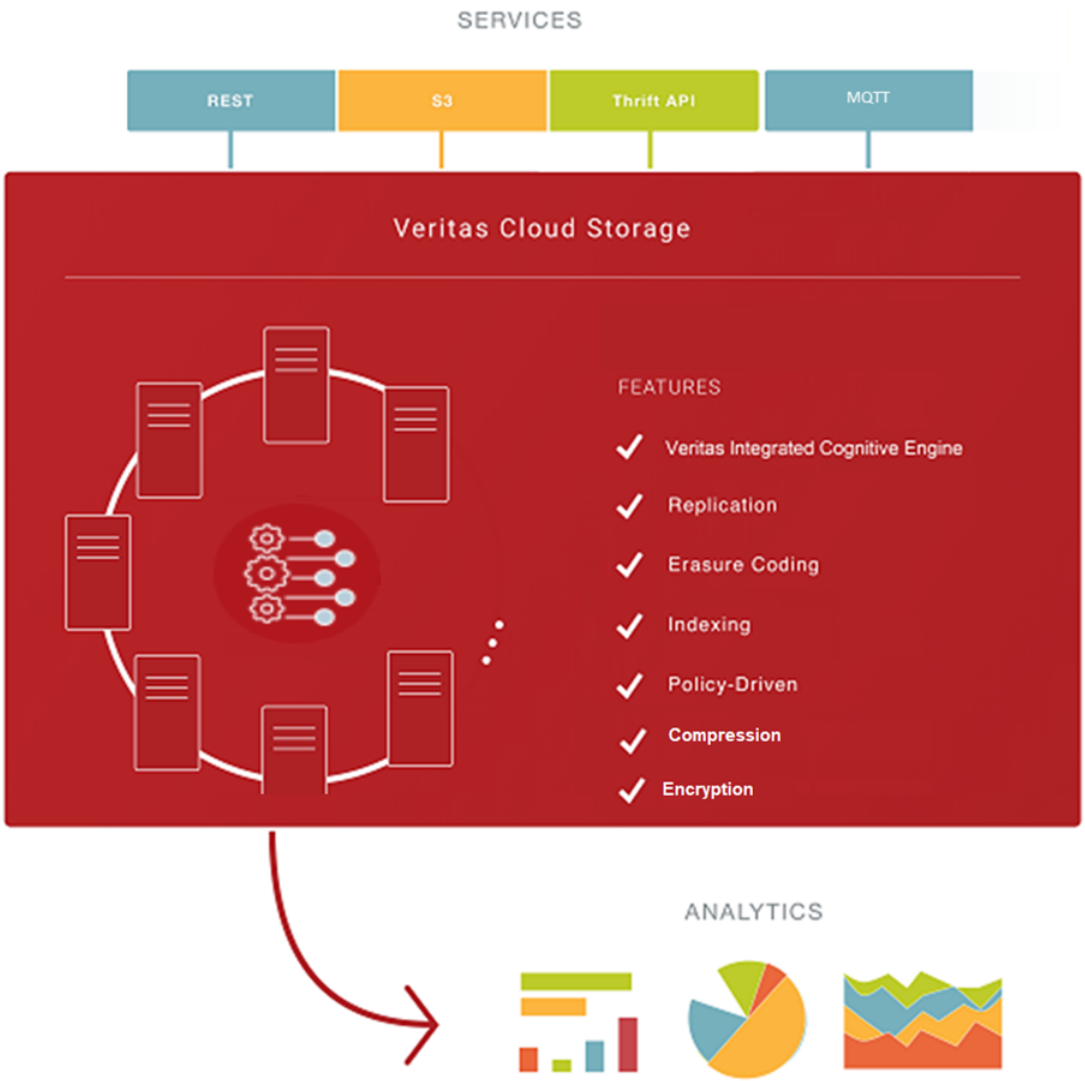
Overview of Veritas Cloud Storage

This chapter includes the following topics:

About Veritas Cloud Storage

Veritas Cloud Storage is a software-defined, scale-out storage platform for unstructured data managed by an included browser-based user interface or RESTful APIs. Its policy-based control plane delivers massive scale, automates data classification, and enables analytics all at commodity pricing. Veritas Cloud Storage empowers enterprise IT to provide a true scale-out, geo-distributed information management platform for the Internet of Things (IoT), General Data Protection Regulation (GDPR), and next-generation workloads.

Figure 1-1 Veritas Cloud Storage architecture diagram



You can use and manage Veritas Cloud Storage in either of the following ways.

Table 1-1 Interfaces for managing Veritas Cloud Storage

Interface	Description
GUI	Centralized dashboard with operations for managing your Veritas Cloud Storage cluster. See the GUI and the online Help for more information.
RESTful APIs	Enables automation using scripts, which run storage administration commands against the Veritas Cloud Storage cluster. You can access the RESTful APIs by accessing the following URL: `https:/`*`hostname_for_cloudstorage`*`/swagger/` *hostname_for_cloudstorage* is the host name designated for the Veritas Cloud Storage cluster.

About Veritas Cloud Storage features

The following is a brief introduction to Veritas Cloud Storage key components and their relationships. Storage administrators responsible for deploying and configuring the product need to understand these features in more detail.

Resiliency

Provides data protection against node failures by storing multiple copies of data using either replication or erasure coding.

Erasure coding only adds additional parity bits, less than one additional copy of the data.

See "About erasure coding and how it is implemented in Veritas Cloud Storage" on page 59.

See "About namespaces, buckets, objects, and attributes" on page 55.

Policy manager

Allows you to specify policies for classifying your data.

See "About classification of data" on page 48.

Allows you to specify the time to live (TTL) period for storing objects.

See "About time to live (TTL)" on page 51.

Allows you to specify a SSD or HDD device for storing objects.

See "About storage discovery" on page 45.

Indexing

Allows you to query metadata attributes using the **Metadata Search** builder.

See "About classification of data" on page 48.

Consistency level	Guarantees the last updated value for the data. Eventual consistency - provides improved write performance by only requiring a majority of copies to be written before acknowledging the write operation back to the application as opposed to requiring all of the copies to be written. See "About namespaces, buckets, objects, and attributes" on page 55.
Services	Includes support for the following protocol services: ■ S3 ■ REST ■ MQTT ■ Apache Thrift See "Supported services" on page 66. See "About MQTT" on page 66. See "About Apache Thrift" on page 76.
Licensing	Includes support for the following types of licenses: ■ Perpetual ■ Subscription ■ Trialware See "About Veritas Cloud Storage licensing" on page 20.
Security	Includes support for the following: ■ Encryption at rest ■ Encryption on the wire See "About encryption at rest" on page 84. See "About encryption on the wire" on page 85.
Scalability	Allows you to increase the storage capacity by increasing the number of nodes, or the capacity per node. You can start with partially populated nodes and then add more disks, or swap out existing disks with higher capacity models at a later date. See "About storage discovery" on page 45.
Global namespace	Provides a global namespace across data centers to provide users with the ability to write to and access objects within a common pool of storage. See "About namespaces, buckets, objects, and attributes" on page 55.

Disaster Recovery	Supports disaster recovery for protecting your data against natural or man-made disasters. See “Configuring multiple locations for disaster recovery” on page 57.
Compression	Compression is enabled by default within each node when creating a bucket and is performed at the disk block level. See “About compression” on page 56.

Chapter 2

System requirements

This chapter includes the following topics:

Important release information

Review the *Veritas Cloud Storage Release Notes* for the latest information before you install the product.

For important updates regarding this release, review the Late-Breaking News TechNote on the Veritas Technical Support website:

https://www.veritas.com/support/en_US/article.100040603.html

System requirements

Recommended requirements per node for running Veritas Cloud Storage system software:

- CPU - 16 physical cores
- Memory - 64 GB
- Storage - directly attached (RAID 1+0 recommended)
- Network - Bonded 10 GbE (data and inter-node communication)

Note: Requirements are dependent on workloads.

Note: RAID requirements are for performance.

Linux requirements

Table 2-1 Operating system requirements

Operating System	Version
Red Hat Enterprise Linux	7.2 and above
CentOS	7 1708 Kernel version: 3.10.0-693

Default ports

Table 2-2 displays the default ports that Veritas Cloud Storage uses to transfer information.

All of the default ports can be changed.

See the *Veritas Cloud Storage Quick Start Guide* for more information on changing the default ports.

Table 2-2 Default Veritas Cloud Storage ports

Port	Protocol or service	Purpose	Impact if blocked
80	GUI, REST, S3 HTTP	Access to the Veritas Cloud Storage GUI, RESTful APIs, and S3.	Cannot access the Veritas Cloud Storage GUI, the RESTful APIs. or S3. Cannot create or delete buckets using the S3 protocol.
443	GUI, REST, S3 HTTPS	Access to the Veritas Cloud Storage GUI, RESTful APIs If SSL is configured, and S3.	Cannot access the Veritas Cloud Storage GUI, the RESTful APIs, or S3.
2112	Apache Thrift	Access to the Apache Thrift APIs.	Cannot access the Apache Thrift APIs.
4369	EPMD (Erlang Port Mapper Daemon)	Used when one node running the Erlang process needs to communicate with a different node. **Note:** You do not need to enable this port when running a single node.	Cannot access the cluster.
6000	Distributed Erlang	Access to Distributed Erlang.	Cannot access the cluster.
8098	vnode handoff	Allows nodes to communicate with each other.	Nodes cannot communicate with one another.
8180	Veritas Integrated Cognitive Engine (VICE)	Access to VICE policies for performing classification.	No other application should use this port. Classification will not work.

Verifying firewall settings and enabling ports

If the `vrts-viking` service fails to start, check if one or more of the default ports are blocked.

See "Default ports" on page 15.

Verifying firewall settings and enabling ports

1. Determine if the Veritas Cloud Storage firewall is running or not.

```
# systemctl status firewalld
```

2. Allow the following ports in Veritas Cloud Storage.

   - TCP protocol

```
# firewall-cmd --permanent --zone=public --add-protocol=tcp
```

   - HTTP port

```
# firewall-cmd --permanent --zone=public --add-port=80/tcp
```

   - Veritas Cloud Storage interaction port

```
# firewall-cmd --permanent --zone=public --add-port=6000/tcp
```

   - Veritas Cloud Storage handoff port

```
]# firewall-cmd --permanent --zone=public --add-port=8098/tcp
```

   - Veritas Cloud Storage Thrift port

```
# firewall-cmd --permanent --zone=public --add-port=2112/tcp
```

Note: These commands work if the port is at the operating system level. Some firewalls that work at the network level, and firewalls at the network level require the network teams to allow the ports in the network.

---

3. Execute the following command after adding the port for the changes to take effect.

```
# firewall-cmd --reload
```

4 You can change the default ports by updating the respective port parameters in the `/etc/viking.conf` file or the `/etc/default/vikings` file. Restart the `vrts-viking` service for the port changes to take effect.

```
# systemctl restart vrts-viking
```

For example, to configure a different port for HTTP/REST, update the parameter `viking.http.port` in the`/etc/viking.conf` file.

5 Verify that the ports are added or not.

```
# firewall-cmd --permanent --list-all
```

6 Perform the same steps for all the nodes.

7 You can also verify the ports using the Red Hat GUI.

Supported web browsers for Veritas Cloud Storage

Veritas Cloud Storage does not require that you have a browser installed. You can access the UI directly using a laptop or desktop system. The browser version is not a server requirement. Browser support is a requirement for the clients that need to the access the Veritas Cloud Storage UI.

The following are the supported web browsers for Veritas Cloud Storage:

Table 2-3 Supported web browsers

Browser	Version	Comments
Google Chrome	49 and above	JavaScript: Enabled, Cookies: Enabled, WebGl: Enabled
Mozilla Firefox	55 and above	JavaScript: Enabled, Cookies: Enabled, WebGl: Enabled

Your browser must support JavaScript 1.2 or later. If you use pop-up blockers (including Yahoo Toolbar or Google Toolbar), either disable them or configure them to accept pop-ups from the Veritas Cloud Storage node to which you connect. Ensure that the site is included in the list of trusted sites. WebGL support is dependent on GPU support and may not be available on older devices. This is due to the additional requirement for users to have up-to-date video drivers.

Supported S3 clients and versions

The following are the supported S3 client and versions.

Table 2-4 Supported S3 clients and versions

S3 client	Version
s3cmd	2.0.1
AWS s3 CLI	1.11.182
S3 browser	6.5.9 pro version
CloudBerry	5.2.0.26 (freeware)

Maximum tested configurations

The maximum tested configurations for Veritas Cloud Storage are as follows:

Table 2-5 Maximum tested configurations

Veritas Cloud Storage software	Tested with
Number of buckets per namespace	500,000 buckets
Number of buckets per cluster	500,000 buckets

Minimum number of nodes

Veritas Cloud Storage requires using a minimum of a four-node cluster. Add nodes in increments of four instead of one.

Chapter 3

Licensing in Veritas Cloud Storage

This chapter includes the following topics:

About Veritas Cloud Storage licensing

You have to obtain a valid license to install and use Veritas Cloud Storage.

You can choose one of the following licensing methods when you install a product:

- Enter a valid perpetual license key file matching the functionality in use on the systems.
  A perpetual license is like a permanent license for using Veritas Cloud Storage.
- Enter a valid subscription license key file matching the functionality in use on the systems.
  A subscription license is a license with validity from one to three years.
- Continue with evaluation mode and complete system licensing later.
  This license is a trialware license that can be used for 60 days.
  Installation without a license does not eliminate the need to obtain a license. The administrator and company representatives must ensure that a server or cluster is entitled to the license level for the products installed. Veritas reserves the right to ensure entitlement and compliance through auditing.

To comply with the terms of the End User License Agreement (EULA), you have 60 days to either enter a valid subscription or perpetual license key or continue in evaluation mode.

A trialware license key is installed by default as part of the Veritas Cloud Storage product installation.

If you encounter problems while licensing this product, visit the Veritas Licensing Support website.

https://www.veritas.com/licensing/process

The Veritas Cloud Storage licensing has a few functional enforcements.

Table 3-1 Functional enforcement of Veritas Cloud Storage licensing

Enforcement	Action
During validity	None
During grace period	Persistent message (in the GUI only)
Post-grace period	Before you restart the node, you can stop the Veritas Cloud Storage service, but you cannot start the services again (even if you have not restarted the node). After you restart the node, the Veritas Cloud Storage service does not come online on the restarted node until a valid subscription or perpetual license has been entered. Any data stored in the Veritas Cloud Storage cluster during the grace period remains and is accessible as long as the cluster is online; however, no new nodes or configuration of the existing cluster is allowed once the grace period has been exceeded.

Section 2

Deploying Veritas Cloud Storage

Chapter 4

Deploying Veritas Cloud Storage

This chapter includes the following topics:

Overview of Veritas Cloud Storage deployment

You can install Veritas Cloud Storage using the following methods:

- Installation of a single RPM on a physical setup
  See “Installing Veritas Cloud Storage on a physical set up” on page 24.
- Installation using Ansible or Puppet for large-scale deployments
  See “Installing Veritas Cloud Storage using Ansible or Puppet” on page 25.

Installing Veritas Cloud Storage on a physical set up

To install Veritas Cloud Storage on a physical set up

1. Install the Red Hat Enterprise Linux operating system on all the nodes in the cluster.

   See the *Veritas Cloud Storage Release Notes* for the supported version of Red Hat Enterprise Linux.

   ```
   # cat /etc/redhat-release
   ```

2. Make sure that the required storage is available for storing data.

   ```
   # lsblk
   ```

   Verify that the disk devices do not have partitions.

3. Access the RPM installation script on the Veritas Entitlement Management System (VEMS) website at: https://www.veritas.com/support.

   On the VEMS page, click on **Licensing**. You are prompted to log on.

4. The RPM installation script requires that the following packages be installed:

   - bzip2-libs
   - java-1.8.0-openjdk-headless
   - libstdc++
   - lvm2
   - ncurses-libs
   - openssl
   - shadow-utils
   - sudo
   - xfsprogs
   - zlib

5. Make the RPM installation script executable.

   ```
   # chmod +x VRTScloudstorage.sh
   ```

6. Run the RPM installation script.

   ```
   # ./VRTScloudstorage.sh
   ```

7. After running the installation script, you are prompted to press **Enter** to see the End-User License Agreement (EULA).
8. At the end of the EULA agreement, enter **Y** to accept the EULA.

   After accepting the EULA, the RPM installation starts.
9. You can change the default ports by updating the respective port parameters in the `/etc/viking.conf` file or the `/etc/default/vikings` file.

   Restart the `vrts-viking` service for the port changes to take effect.

   ```
   # systemctl restart vrts-viking
   ```

   For example, to configure a different port for HTTP/REST, update the parameter `viking.http.port` in the `/etc/viking.conf` file.

   Ensure that the same ports are specified on all the nodes to avoid connection failures on the clients. This is applicable only for the `viking.xyz.port` type of parameter in the configuration file. All the configuration parameters in the`/etc/viking.conf` file across all the nodes should be the same to avoid unpredictable behavior.
10. Open a browser window and access the GUI https://*node1-ipaddress*, where the *node1-ipaddress* is the IP address of node1.

    On the log-on screen, enter the default user name and password, `admin/admin`.
11. Configure the initial node by selecting the disk devices and optionally providing the location and rack number.
12. Add any additional nodes using the Veritas Cloud Storage GUI.

Installing Veritas Cloud Storage using Ansible or Puppet

Veritas Cloud Storage supports deployments using the following Configuration Management (CM) and Remote Execution (RE) tools:

- Ansible
- Puppet

Section 3

Configuring the HAProxy load balancer

Chapter 5

Configuring the HAProxy load balancer

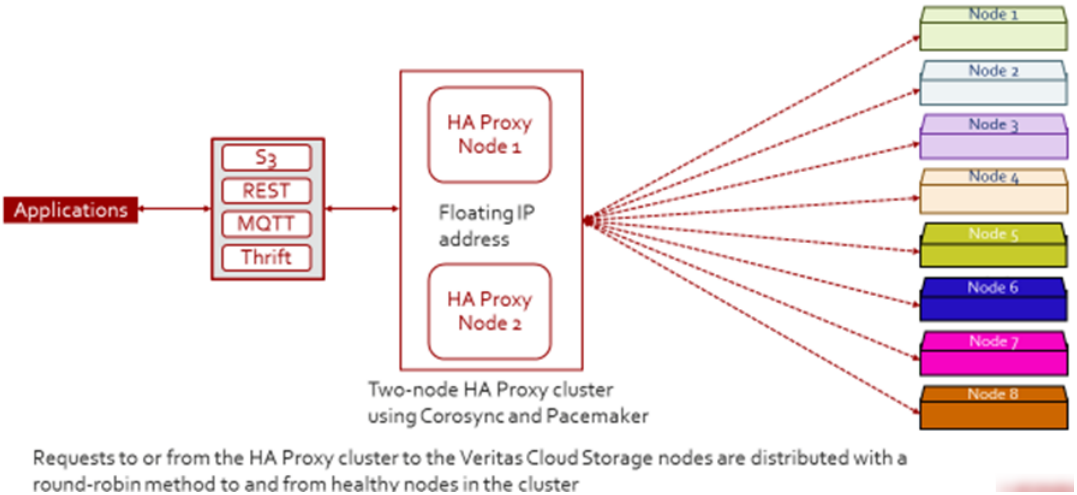
This chapter includes the following topics:

About HAProxy

HAProxy is an open source load balancer, available with most mainstream Linux distributions. It offers scalable and reliable proxying for TCP-based applications.

Note: Using HAProxy for load balancing is optional; it is not required as part of the Veritas Cloud Storage installation.

Figure 5-1 Veritas Cloud Storage with load balancer



For more information on HAProxy, see http://www.haproxy.org/.

Configuring HAProxy

HAProxy should be run on a node that is not part of the Veritas Cloud Storage cluster.

HAProxy host should be connected using the same private network links as the Veritas Cloud Storage nodes.

To install and configure HAProxy

1 Install HAProxy.

```
# yum install haproxy
```

2 Add the following lines in the `/etc/firewalld/services/haproxy-http.xml` file.

```
<?xnl version="1.0" encoding="utf-8"?>
<service>
<description>HAProxy load-balancer</description>
<port protocol="tcp" port="80"/>
</service>
```

3 As root, assign the correct SELinux context and file permissions to the `haproxy-http.xml` file.

```
# cd /etc/firewalld/services
# restorecon haproxy-http.xml
# chmod 640 haproxy-http.xml
```

4 Update the `/etc/haproxy/haproxy.cfg` file with the following frontend and backend server information.

In this example, 10.10.192.22 is the IP address of the HAProxy server.

10.10.192.18, 10.10.192.19, and 10.10.192.20 are the IP addresses of Veritas Cloud Storage.

```
frontend http
bind 10.10.192.22:80
mode http
default_backend nodes
...
backend nodes
    balance     roundrobin
    option forwardfor
    http-request set-header X-Forwarded-Port %[dst_port]
    server   zeus-b006 10.10.192.18:80 check
    server   zeus-b007 10.10.192.19:80 check
    server   zeus-b009 10.10.192.20:80 check
```

Note: If the backend services on Veritas Cloud Storage services are running on a different port, use the appropriate port numbers instead. For example, for the S3 service, use (default) port 8080.

---

5 Enable and start haproxy.

```
# systemctl enable haproxy
# systemctl start haproxy
```

6 Verify by writing a new K,V pair locally (127.0.0.1) from one of the Veritas Cloud Storage (backend) nodes and read (GET) from the HAProxy node.

```
PUT
# curl --insecure -i -X PUT --header 'Content-Type: application/json'
--header 'Accept: application/json'
--header "Authorization: Bearer $TOKEN" -d '{ "value": "local write from
backend server"
, "attrs": {} }' 'https://127.0.0.1/api/v1/namespaces/test-ns/buckets/
test-bkt/keys/testkey'
HTTP/1.1 200 OK
server: Cowboy
date: Fri, 01 Dec 2017 03:49:14 GMT
content-length: 20
content-type: application/json; charset=utf-8
cache-control: max-age=0, private, must-revalidate
x-request-id: i1gcpkreknvg14jud8bg7ncvvf8onf4f

{"status":"success"}
#

GET
# curl --insecure -i -X GET --header 'Content-Type: application/json'
--header "Authorization: Bearer $TOKEN"
'https://<HA-PROXY-SERVER-IP>/api/v1/namespaces/EMEA-marketing/buckets/
mkt-france/keys/testkey'
HTTP/1.1 200 OK
server: Cowboy
date: Fri, 01 Dec 2017 03:50:25 GMT
content-length: 229
content-type: application/json; charset=utf-8
cache-control: max-age=0, private, must-revalidate
x-request-id: icbvata13g2uoso5e20r9h051vs31kdo

{"value":"local write from backend server","namespace":"test-ns",
"key":"testkey","bucket":"test-bkt",
"attrs":{"_size":31,"_owner":"admin","_md5"
:"baefe5e1d22c45700540fa9bc5a8175c",
"_creation_time":1512100155,"_acls":[]}}
#
```

Note: $TOKEN used in the above command is a shell variable storing the value of the Authentication Token for the admin user. The Authentication token is obtained using the authenticate API.

Example:

```
curl --insecure -H "Content-Type: application/json" -X POST -d
'{"username":"admin", "password":"admin"}' https://localhost/api/authenticate
{"token":"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJVc2VyOmFkbWluIiwiZXhwI
joxNTEyMTg1NTM3LCJpYXQiOj
E1MTIwOTkxMzcsImlzcyI6IlZpa2luZ0FwaSIsImp0aSI6Ijk0YTg5NTgzLTIxNmUtNDQwYy1iMTAx
LWM5ZDdjZGI4NWMzNCIsInBlbSI6e30sInN1YiI6IlVzZXI6YWRtaW4iLCJ
0eXAiOiJ0b2tlbiJ9.9DubdgUJ3Y4bezAycIMlXhbaDwVFvZ_gaJYwBv34RMPP8zJz7OApvqkvecDJxBKh_
46SRXlvAJRX436yA5CwhA"}
#

TOKEN="eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJVc2VyOmFkbWluIiwiZXhwI
joxNTEyMTg1NTM3LCJpYXQiOjE1MTIwOTkxMzcsImlzcyI6Il
Zpa2luZ0FwaSIsImp0aSI6Ijk0YTg5NTgzLTIxNmUtNDQwYy1iMTAxLWM5ZD
djZGI4NWMzNCIsInBlbSI6e30sInN1YiI6IlVzZXI6YWRtaW4iLCJ0eXAiOiJ0b2tlbiJ9.
9DubdgUJ3Y4bezAycIMlXhbaDwVFvZ_gaJYwBv
34RMPP8zJz7OApvqkvecDJxBKh_46SRXlvAJRX436yA5CwhA"
```

7 Multiple HAProxy nodes may be set up or configured with a failover configuration.

Making HAProxy highly available using Pacemaker

The following steps are a reference for making HAProxy highly available over a floating IP address.

Table 5-1 HAProxy load balancer configuration

Node	Name	IP
HAProxy Node1	haproxy1	Physical IP: 10.51.148.250
HAProxy Node2	haproxy2	Physical IP: 10.51.148.251

- Phase 1 - install the required Red Hat Enterprise Linux RPMs for clustering
- Phase 2 - set up a two-node cluster using Corosync and Pacemaker

- Phase 3 - set up HAProxy configuration with a floating IP address
- Phase 4 - test failover

Phase 1 - installing the required Red Hat Enterprise Linux RPMs

1. On both the HAProxy nodes (haproxy1 and haproxy2), enable Red Hat Enterprise Linux high availability repository:

   ```
   # yum-config-manager --enable rhel-ha-for-rhel-7-server-rpms
   ```

2. Install pacemaker and the pcs packages.

   Pacemaker is a high-availability cluster resource manager for Corosync, CMAN, and or Linux-HA. Pacemaker Configuration System (pcs) is a corosync and pacemaker configuration tool. You can easily view, modify, and create pacemaker-based clusters.

   ```
   # yum install  pacemaker
   # yum install  pcs
   ```

3. Enable corosync, pacemaker, and pcsd to start at boot on both nodes.

   ```
   # chkconfig corosync on
   ```

   Phase 2 - Setting up a two-node cluster using Corosync and HAProxy

4. Set passwordless ssh between the two nodes.
5. Start the pcsd service and set up authentication for the hacluster user.

   ```
   # systemctl start pcsd.service
   # pcs cluster auth haproxy1 haproxy2 -u
   hacluster -p CHANGEME --force
   haproxy1: Authorized
   haproxy2: Authorized
   ```

6 Set up the cluster between the two nodes, haproxy1 and haproxy2.

```
# pcs cluster setup --force --name haproxy-clus
haproxy1 haproxy2
Destroying cluster on nodes: haproxy1,
haproxy2...
haproxy1: Stopping Cluster (pacemaker)...
haproxy2: Stopping Cluster (pacemaker)...
haproxy1: Successfully destroyed cluster
haproxy2: Successfully destroyed cluster
Sending 'pacemaker_remote authkey' to 'haproxy1',
'haproxy2'
haproxy1: successful distribution of the file 'pacemaker_remote authkey'
haproxy2: successful distribution of the file 'pacemaker_remote authkey'
Sending cluster config files to the nodes...
haproxy1: Succeeded
haproxy2: Succeeded
Synchronizing pcsd certificates on nodes haproxy1,
haproxy2...
haproxy1: Success
haproxy2: Success
Restarting pcsd on the nodes in order to reload the certificates...
haproxy1: Success
haproxy2: Success
```

7 Start the cluster.

```
# pcs cluster start --all
haproxy1: Starting Cluster...
haproxy2: Starting Cluster...
```

8 Disable Stonith on both nodes (required for shared data/quorum).

stonith-enabled parameter indicates that failed nodes and nodes with resources that cannot be stopped should be fenced. Since we are only managing a floating IP address and a service (HAProxy), we disable this parameter on every cluster node. This command needs to be executed on both cluster nodes.

```
# pcs property set stonith-enabled=false
# pcs property set stonith-enabled=false
```

9 Verify Corsync configuration for any errors.

```
# crm_verify -L -V
```

10 Set no-quorum-policy to ignore and migration threshold to 1.

no-quorum policy parameter defines the actions to take when the cluster does not have a quorum. Migration threshold defines how many failures may occur for a resource on a node, before this node is marked ineligible to host this resource.

```
# pcs property set no-quorum-policy=ignore
# pcs resource defaults migration-threshold=1
```

Phase 3 - Set up HAProxy configuration using a floating IP address

11 Install HAProxy on both nodes.

```
# yum install haproxy
Loaded plugins: langpacks, product-id, search-disabled-repos,
subscription-manager
Resolving Dependencies
# yum install haproxy
```

12 On both nodes, set up SELinux to allow the haproxy service to listen on any port.

```
# setsebool -P haproxy_connect_any 1
# setsebool -P haproxy_connect_any 1
```

13 Configure HAProxy to bind on a floating IP address in `haproxy.conf` and add the other haproxy node as a peer.

```
#Add following entries in the default scope in /etc/haproxy/haproxy.cfg:
peers ha-web
  peer haproxy1  10.51.148.250:80
  peer haproxy2  10.51.148.251:80
#Bind to floating IP address:
frontend www1-http
bind 10.51.149.198:9000
backend www1-http
    balance     roundrobin
# round robin balancing between the various backends
#--------------------------------------------------------------
server  veritas-cloud-server-node1  10.51.148.240:80  check
server  veritas-cloud-server-node2  10.51.148.241:80  check
server  veritas-cloud-server-node3  10.51.148.242:80  check
```

14 Add the floating IP address resource for high availability.

```
# pcs resource create haproxy_ip IPaddr2 ip=10.51.149.198
cidr_netmask=22 --group=haproxy_group
Assumed agent name 'ocf:heartbeat:IPaddr2' (deduced from 'IPaddr2')
```

15 Add the HAProxy service resource for high availability. We add both resources (IP and HAProxy service) to the same group.

```
# pcs resource create haproxy_service systemd:haproxy
--group=haproxy_group
```

16 Define constraint to first fail over the IP and then try to start the service.

```
# pcs constraint order start haproxy_ip then haproxy_service
Adding haproxy_ip haproxy_service (kind: Mandatory)
(Options: first-action=start then-action=start)
```

17 View the cluster status and the status of the resources. HAProxy is started by pacemaker.

```
# pcs status
Cluster name: haproxy-clus
Stack: corosync
Current DC: haproxy2 (version 1.1.16-12.el7_4.2-94ff4df) - partition
with quorum
Last updated: Tue Sep 12 02:29:49 2017
Last change: Tue Sep 12 02:29:39 2017 by root via cibadmin on haproxy1
2 nodes configured
2 resources configured
Online: [ haproxy1 haproxy2 ]
Full list of resources:
 Resource Group: haproxy_group
     haproxy_ip (ocf::heartbeat:IPaddr2):  Started haproxy1
     haproxy_service (systemd:haproxy):   Started haproxy1
Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
# systemctl status haproxy
  haproxy.service - Cluster Controlled haproxy
   Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled;
vendor preset: disabled)
  Drop-In: /run/systemd/system/haproxy.service.d
           └─50-pacemaker.conf
   Active: active (running) since Tue 2017-09-12 02:28:25 PDT;
2min 3s ago
 Main PID: 17490 (haproxy-systemd)
   CGroup: /system.slice/haproxy.service
 ├─17490 /usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/haproxy.
cfg -p
 /run/haproxy.pid
 ├─17492 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p
 /run/haproxy.pid -Ds
 └─17493 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p
 /run/haproxy.pid -Ds
Sep 12 02:28:25 haproxy1 systemd[1]: Started Cluster Controlled
haproxy.
Sep 12 02:28:25 haproxy1 systemd[1]: Starting Cluster Controlled
haproxy...
Sep 12 02:28:25 haproxy1 haproxy-systemd-wrapper[17490]:
```

```
haproxy-systemd-wrapper:
executing /usr/sbin/ha...-Ds
Sep 12 02:28:25 haproxy1 haproxy-systemd-wrapper[17490]:
[WARNING] 254/022825 (17492)
: parsing [/etc/ha...ity
Sep 12 02:28:25 haproxy1 haproxy-systemd-wrapper[17490]:
[WARNING] 254/022825 (17492)
: parsing [/etc/ha...ity
Sep 12 02:28:25 haproxy1 haproxy-systemd-wrapper[17490]:
[WARNING] 254/022825 (17492)
: parsing [/etc/ha...d ?
Sep 12 02:28:25 haproxy1 haproxy-systemd-wrapper[17490]:
[WARNING] 254/022825 (17492)
: parsing [/etc/ha...ity
Hint: Some lines were ellipsized, use -l to show in full
# ip a|grep 198
   inet 10.51.149.198/22 brd 10.51.151.255 scope global
   secondary eno16780032
```

Phase 4 - Test failover

18 Verify failover of the HAProxy service (and the floating IP) by killing HAProxy processes on the active node.

```
# kill -9 17492 17493
# ps -ef|grep haproxy
root     18520 15944  0 02:31 pts/0    00:00:00 grep --color=auto
haproxy
# pcs status
Cluster name: haproxy-clus
Stack: corosync
Current DC: haproxy2 (version 1.1.16-12.el7_4.2-94ff4df) - partition
with quorum
Last updated: Tue Sep 12 02:32:55 2017
Last change: Tue Sep 12 02:29:39 2017 by root via cibadmin on haproxy1
2 nodes configured
2 resources configured
Online: [ haproxy1 haproxy2]
Full list of resources:
 Resource Group: haproxy_group
    haproxy_ip (ocf::heartbeat:IPaddr2): Started haproxy2
    haproxy_service (systemd:haproxy):   Started haproxy2
Failed Actions:
* haproxy_service_monitor_60000 on haproxy1'not running' (7): call=149,
status=complete, exitreason='none',
   last-rc-change='Tue Sep 12 02:32:27 2017', queued=0ms, exec=0ms
Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

19 Check the status of the HAProxy service and the floating IP on the redundant node, haproxy2.

```
# systemctl status haproxy
   haproxy.service - Cluster Controlled haproxy
   Loaded: loaded (/usr/lib/systemd/system/haproxy.service; disabled;
   vendor preset: disabled)
  Drop-In: /run/systemd/system/haproxy.service.d
    └─50-pacemaker.conf
   Active: active (running) since Tue 2017-09-12 02:32:34 PDT; 47s ago
 Main PID: 26834 (haproxy-systemd)
   CGroup: /system.slice/haproxy.service
   ├─26834 /usr/sbin/haproxy-systemd-wrapper -f
   /etc/haproxy/haproxy.cfg
   -p /run/haproxy.pid
   ├─26836 /usr/sbin/haproxy -f
   /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds
   └─26837 /usr/sbin/haproxy -f
   /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -D
Sep 12 02:32:34 haproxy2.systemd[1]: Started Cluster Controlled
haproxy.
Sep 12 02:32:34 haproxy2.systemd[1]: Starting Cluster Controlled
haproxy...
Sep 12 02:32:34 haproxy2.haproxy-systemd-wrapper[26834]:
haproxy-systemd-wrapper: executing /usr/sbin/ha...-Ds
Sep 12 02:32:34 haproxy2.haproxy-systemd-wrapper[26834]:
[WARNING] 254/023234 (26836) : parsing [/etc/ha...ity
Sep 12 02:32:34 haproxy2.haproxy-systemd-wrapper[26834]:
[WARNING] 254/023234 (26836) : parsing [/etc/ha...ity
Sep 12 02:32:34 haproxy2.haproxy-systemd-wrapper[26834]:
[WARNING] 254/023234 (26836) : parsing [/etc/ha...d ?
Sep 12 02:32:34 haproxy2.haproxy-systemd-wrapper[26834]:
[WARNING] 254/023234 (26836) : parsing [/etc/ha...ity
Hint: Some lines were ellipsized, use -l to show in full.
```

20 Ensure that pcsd, corosync, and pacemaker services on both nodes are enabled, so that they start after reboot.

```
# systemctl enable corosync
Created symlink from /etc/systemd/system/multi-user.target.wants/
corosync.service
to /usr/lib/systemd/system/corosync.service.
# systemctl enable pacemaker
Created symlink from /etc/systemd/system/multi-user.target.wants/
pacemaker.service
to /usr/lib/systemd/system/pacemaker.service.
# systemctl enable corosync
Created symlink from /etc/systemd/system/multi-user.target.wants/
corosync.service
to /usr/lib/systemd/system/corosync.service.
# systemctl enable pacemaker
Created symlink from /etc/systemd/system/multi-user.target.wants/
pacemaker.service
to /usr/lib/systemd/system/pacemaker.service.
# pcs status|tail -5
Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Section 4

Managing users and roles

Chapter 6

Managing users and permissions

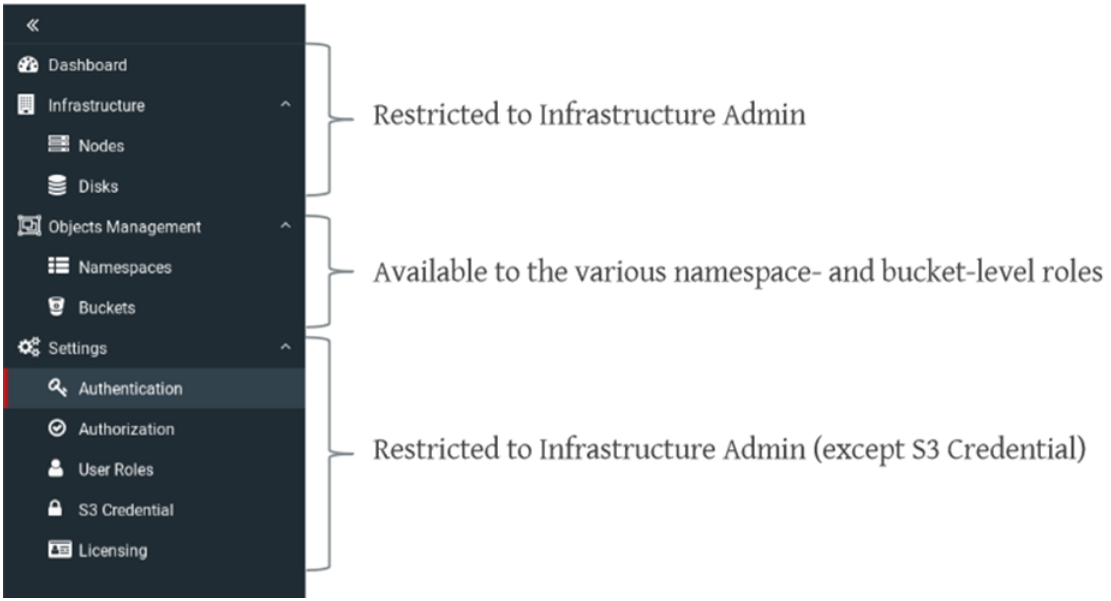
This chapter includes the following topics:

About role-based access controls

Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise.

RBAC is supported for all UI and RESTful API interactions. RBAC is enforced for RESTful APIs, including those that are used by the UI.

UI elements are hidden according to available roles.



Veritas Cloud Storage supports up to six distinct or unique roles.

See "Role assignments" on page 43.

Role assignments

Role assignments require LDAP or Active Directory (AD) configuration. Local users, except for the default admin user, are not supported.

Table 6-1 Role assignments

Role	User
Global roles	Infrastructure administrator Namespace administrator
Namespace-level roles	Namespace owner
Bucket-level roles	Bucket owner Bucket reader Bucket writer

See "About role-based access controls" on page 42.

Section 5

Managing Veritas Cloud Storage capacity

Chapter 7

Optimizing Veritas Cloud Storage capacity

This chapter includes the following topics:

About storage discovery

The storage discovery layer on every node discovers all the important attributes that are associated with the node, such as media type of SSD or HDD devices, disk vendor, disk model, storage capacity, and so on.

The storage tiers that are comprised of SSD and or HDD devices are defined automatically on every node. The storage discovery layer runs as a daemon on every node. It runs the discovery cycle every minute to discover new devices and refresh the state of the identified devices. Currently, only whole disks are supported and any disks with partitions are skipped. Disks that have mounted file systems from other applications are also skipped.

About storage placement policies

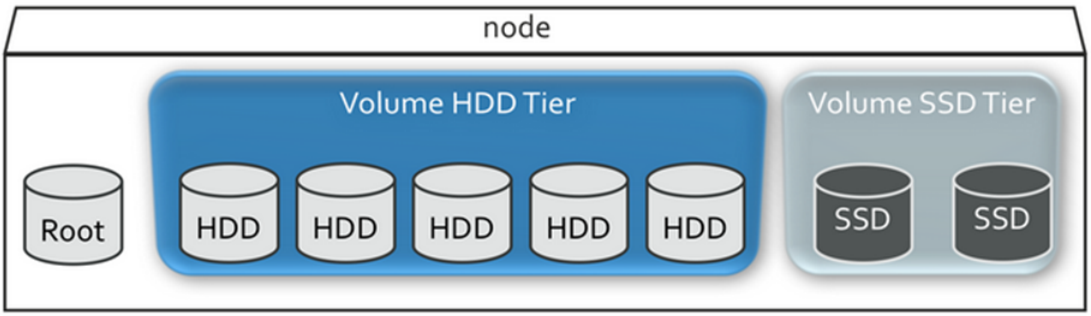
Clusters are formed by combining (or connecting) two or more nodes together. The storage discovery layer discovers the important attributes associated with the node, SSD or HDD, vendor, product line, storage capacity, and so on. Whenever an infrastructure administrator assigns disks to Veritas Cloud Storage, internally the disks get added to the SSD or HDD tier depending on its type.

You can associate a storage tier when creating a namespace. If you have data that needs to be accessed quickly, you can associate the tier according to that data.

For example, you can access data in an SSD tier, which may be comprised of SSDs quickly as compared to other tiers. You can set the storage tiers as a bucket-level policy when creating a bucket. Any PUT operations done on that bucket cause the object to land on the specified tier. You have privileges for placing your data in an efficient manner.

Figure 7-1 shows a node with an HDD tier and an SSD tier. Post-installation as disks are added to the nodes, they are added to the existing file systems or volumes of the relative type.

Figure 7-1 Storage configuration of a node with an HDD tier and an SSD tier



Section 6

Using policies for automating tasks

Chapter 8

Managing policies

This chapter includes the following topics:

About classification of data

Veritas Cloud Storage includes the Veritas Integrated Cognitive Engine (VICE), which enables classification and metadata tagging of objects as they get ingested and stored in Veritas Cloud Storage. Classification, if enabled at the namespace level, can then be selectively enabled or disabled on buckets within that namespace. Classification tags are stored as attributes along with the object.

Objects are classified based on the value (object data) as well as based on the attributes (metadata) that are stored with an object.

You can query classified objects, such as Personal Identifiable Information (PII), intellectual property, regulations, corporate compliance, and so on using the **Metadata Search** option after you have created a bucket.

General Data Protection Regulation (GDPR) is a potential use case for classifying your data.

At the namespace level, the namespace owner can choose policies available for every bucket under that namespace.

At the bucket level, only policies enabled at the namespace level are available for selection.

Each policy has its own set of tags, which are used to query the object metadata. Once you create a namespace, the set of policies applied is locked, and is non-editable. Once you select the policies, the classification process runs in the background as a daemon, and classification of data occurs based on the policy details that you selected.

Veritas Cloud Storage does not currently support the ability to reclassify the objects once they are stored within their respective bucket.

About creating custom metadata tags

You can use Veritas Cloud Storage to separate file metadata from data. You can tag objects contained in a bucket with key-value pairs. This metadata is presented through HTTP headers on requests and responses.

You can add custom metadata tags using either of the following methods:

- Veritas Cloud Storage RESTful APIs
  See "Adding custom metadata tags using the Veritas Cloud Storage RESTful APIs" on page 50.
- Amazon S3cmd, command-line S3 client
  See "Adding custom metadata tags using s3cmd" on page 50.

Table 8-1 Questions and answers regarding custom metadata tags

Question	Answer
Is there any limitation on the name length of custom metadata tags?	There is no limit. It is recommended to use smaller names.
Is there any limitation on the size of the custom metadata tags?	There is a configurable bucket level limit on the size of the custom metadata tags. By default, it is 256 bytes.
How many metadata tags can you add to Veritas Cloud Storage?	There is no limit.
Is there any limitation on the number of metadata tags per bucket or cluster?	There is no limit.
Can you add custom metadata tags using the Veritas Cloud Storage GUI?	No. You cannot add custom metadata tags using the Veritas Cloud Storage GUI at this time.

Adding custom metadata tags using s3cmd

You can use the Amazon s3cmd to add custom metadata tags to objects within your buckets. You can then assign key-value pairs to the object.

See "About creating custom metadata tags" on page 49.

To add custom metadata tags using s3cmd

1. **Prerequisites:**

   You must have installed s3cmd.

   You must have created a bucket.

2. Use the s3cmd `modify` command to modify your metadata.

   ```
   $ s3cmd modify --add-header x-amz-meta-format:MP4 s3://<bucket>/<object>
   modify: 's3://<bucket>/<object>'
   ```

3. You can then use the option `--add-header` to add an HTTP header.

   The headers starting with `x-amz-meta` are seen as metadata.

   If you want to add a metadata `format` with a value of `MP4`, the key is named `x-amz-meta-format`.

   ```
   $ s3cmd modify --add-header x-amz-meta-format:MP4 s3://<bucket>/<object>
   modify: 's3://<bucket>/<object>'
   ```

4. You can also use the s3cmd `put` command to upload a file into a bucket.

   ```
   $ s3cmd put <filename> --add-header x-amz-meta-format:MP4
   s3://<bucket>/<object> put: 's3://<bucket>/<object>'
   ```

Adding custom metadata tags using the Veritas Cloud Storage RESTful APIs

You can use the Veritas Cloud Storage RESTful APIs to add custom metadata tags for objects in buckets.

See "About creating custom metadata tags" on page 49.

To add custom metadata tags using the Veritas Cloud Storage RESTful APIs

1. **Prerequisites:**

   You must have created a bucket.

2. Use the Veritas Cloud Storage namespaces RESTful API.

3. Add the metadata attributes to the .json file.

```
POST :host/api/v1/namespaces/ns1/buckets/bucket1/keys/k1/attrs
Content-Type: application/json
{

 "attribute1": "format",

 "attribute3": "MP4"
}
```

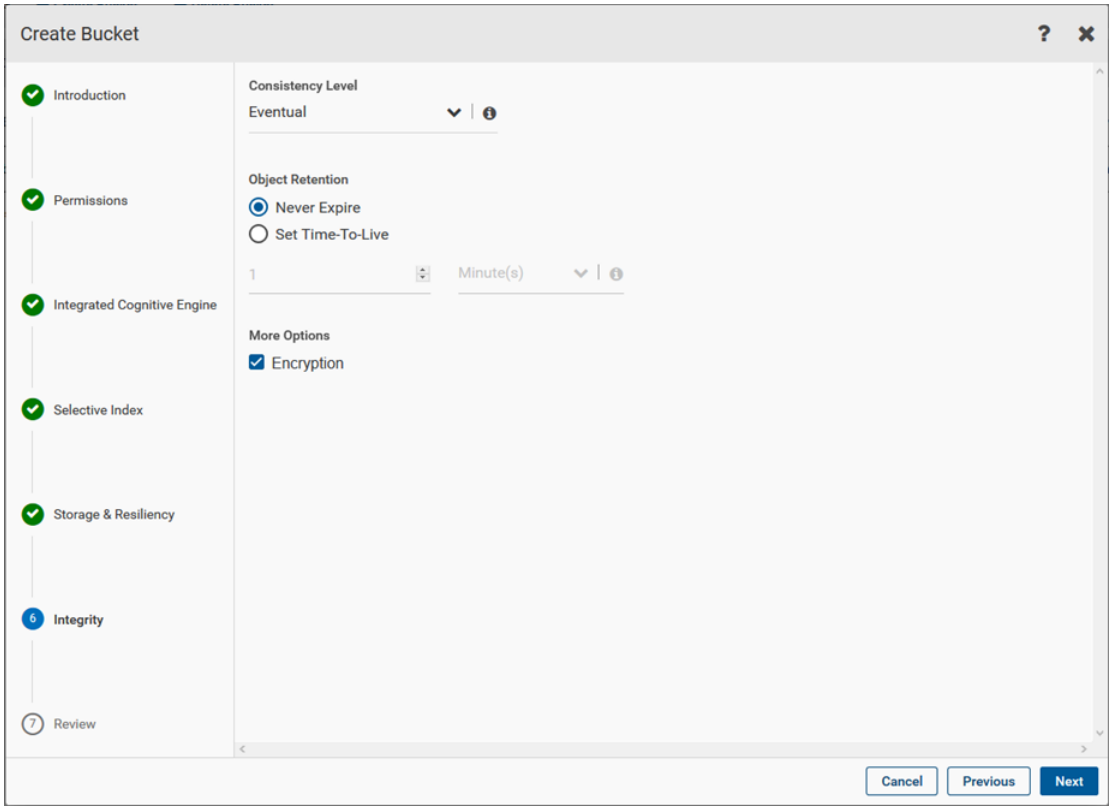
About time to live (TTL)

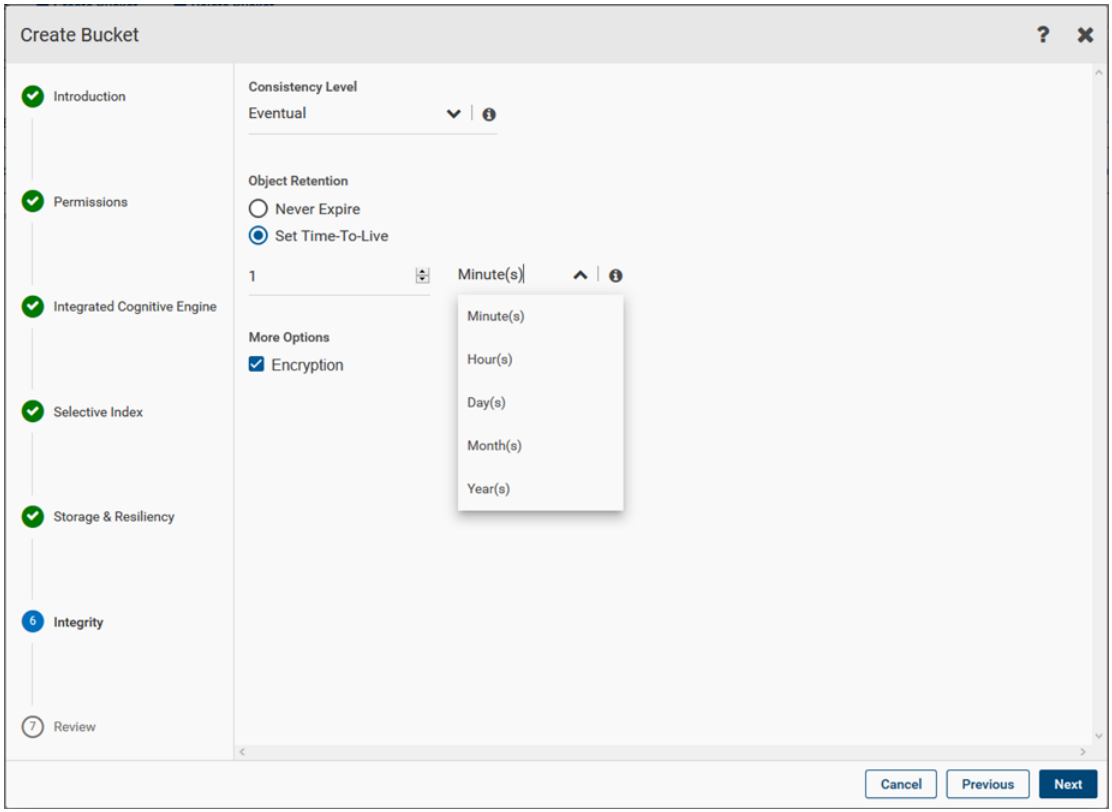
TTL stands for time to live. TTL is an interval of time after which an object in a specific bucket expires. It gets deleted with the next compaction run. Currently, compaction runs at least once in 24 hours, but only if TTL is selected for at least one bucket. If TTL is not selected for any buckets, than compaction does not run.

Using the TTL feature, you can provide TTL intervals through the GUI while creating a bucket. The interval for TTL is set to **Never Expire** by default. Enabling the **Never Expire** option means that the objects are not deleted by the compaction process. You can set the interval in minutes, hours, days, months, and years. Whenever you set TTL, there is a compaction that runs every 24 hours. Having compaction run at least once in 24 hours guarantees that expired records do not linger for more than 24 hours. When the next compaction runs, it flushes all the deleted objects from the bucket and shows the actual available space on the GUI.

If you set TTL to a few minutes, you can remove a lot of expired records before the 24-hour compaction starts.

You can also use a RESTful API to manually trigger compaction.


Create Bucket
Introduction
Permissions
Integrated Cognitive Engine
Selective Index
Storage & Resiliency
6 Integrity
7 Review
Consistency Level
Eventual
Object Retention
Never Expire
Set Time-To-Live
1
Minute(s)
More Options
Encryption
Cancel
Previous
Next


Create Bucket
Introduction
Permissions
Integrated Cognitive Engine
Selective Index
Storage & Resiliency
6
Integrity
7
Review
Consistency Level
Eventual
Object Retention
Never Expire
Set Time-To-Live
1
Minute(s)
Minute(s)
Hour(s)
Day(s)
Month(s)
Year(s)
More Options
Encryption
Cancel
Previous
Next

Section 7

Creating a global namespace for storing your data

Chapter 9

Managing objects

This chapter includes the following topics:

About namespaces, buckets, objects, and attributes

Users with the appropriate permissions can create buckets, and can create objects within the buckets, in the same namespace. A namespace owner can create buckets, but the namespace owner cannot view or list the contents of the objects. A bucket owner can create or update objects.

A namespace provides a mechanism by which objects and buckets are segregated so that an object in one namespace can have the same name as an object in another namespace. A namespace is a logical container for storing buckets.

A bucket is equivalent to a folder in a file system. A bucket belongs to a namespace and objects are stored in the buckets.

An object is represented by a key, value pair where a key uniquely identifies an object in a given bucket. Data stored in the object is represented by the value. You can upload or download objects using the S3 protocol service as an example.

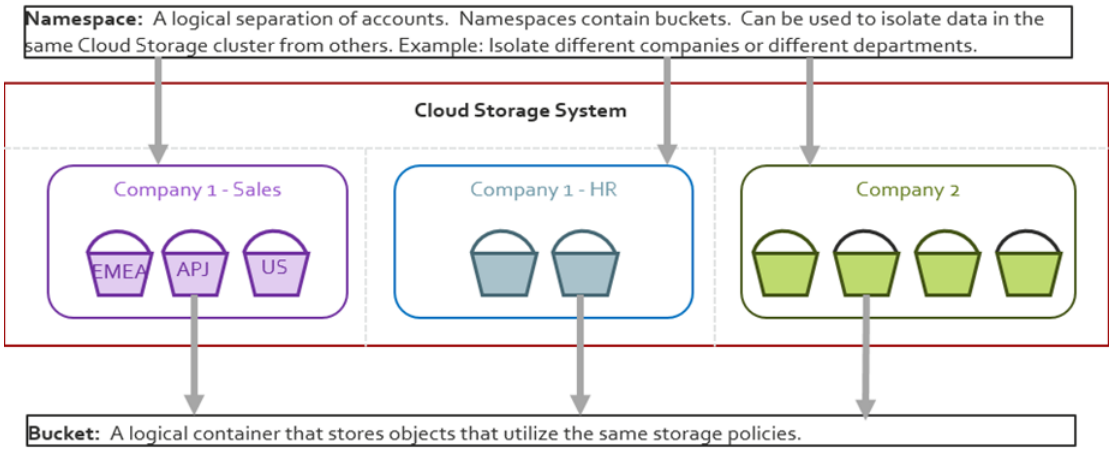
Attributes are key, value pairs that represent metadata associated with an object. There can be multiple attributes associated with any given object. Attributes are indexed so that they can be queried. Attributes can have just keys without any values, for example, tags.

See "About creating custom metadata tags" on page 49.

See "Adding custom metadata tags using s3cmd" on page 50.

See "Adding custom metadata tags using the Veritas Cloud Storage RESTful APIs" on page 50.

Figure 9-1 Example namespace and bucket definitions



Namespaces are where policies are defined and made available to buckets, whereas buckets are where policies are selected and enforced.

About compression

Compression lets you compress infrequently accessed data or data based on access frequency.

Veritas Cloud Storage uses the Snappy algorithm for compression.

Compression is enabled by default within each node when you create a bucket and is performed at the disk-block level.

You cannot configure compression through either the Veritas Cloud Storage GUI or the RESTful APIs.

Table 9-1 Configurable items for compression

Task	What needs to be done

Turn off compression	Make the following edit in the `/etc/viking.conf` file:
	
	`viking.rocksdb.compression = none`
	
	Restart the `vrts-viking` service for the changes to take effect.
	
	`# systemctl restart vrts-viking`
	
	Changes in the configuration file affect only new objects.

Change the compression algorithm	Make the following edit in the `/etc/viking.conf` file: `viking.rocksdb.compression = snappy, zlib, bzip2, lz4, lz4h` This configuration option is set at the global level (node level) and cannot be set or reset at the namespace or bucket level. Changes in the configuration file affect only new objects.

Configuring multiple locations for disaster recovery

What is disaster recovery?

Disaster recovery is necessary to protect your data centers against natural or man-made disaster. You need to understand data and resiliency requirements of your applications and prepare a disaster recovery strategy to preconfigure your data centers across failure domains. Replication of all of your data might not be necessary, however, ensuring availability of critical data is always crucial. Replicating all of your data might multiply your storage costs if data is replicated across data centers when most of the data is not required for an application to survive. Also, it is necessary to perform periodic drills to assess the performance of your disaster-recovery strategy.

Creating a disaster-recovery solution using Veritas Cloud Storage

With Veritas Cloud Storage, you can configure a customized disaster-recovery solution for your data. For any disaster-recovery solution to work, you need to guarantee data placement such that data is always available at multiple data centers. Veritas Cloud Storage uses the node property “location” to identify the data center location of the node. You specify the location once only while adding a new node to a cluster or when creating a new cluster with the first node. The “location” forms a logical grouping of nodes to define the data centers for Veritas Cloud Storage. This is used to create a logical cluster ring, and any data movement due to rebalancing is restricted to nodes within the logical cluster.

When creating a namespace, you can specify the list of locations to which the namespace should be allowed to replicate for disaster recovery. This allowed list of locations is subsequently made available when creating a bucket. Each bucket can have its own disaster plan configured. Additionally, a bucket has configurable resiliency for independent fault tolerance within each location. When multiple locations are configured for a bucket for disaster recovery, the objects created in the bucket are synchronously replicated to all locations.

See “About erasure coding and how it is implemented in Veritas Cloud Storage” on page 59.

Only local reads are allowed for objects stored on a location. In other words, objects replicated to a location can be accessed only from that location. Remote access is not allowed. However, both write (PUT) and list operations are allowed even when the object is not locally available. If all the nodes in the local location are down, you need to connect to the nodes at the remote locations. Application reads are then serviced from the nodes at the remote locations.

Latency and performance considerations

Veritas Cloud Storage supports only synchronous type of replication across locations. Writes to an object that spans multiple locations would have latency resulting from synchronous writes on both local and remote site(s). It is recommended to have a low-latency connection between data centers to avoid network-related issues, such as application timeouts.

See “System requirements” on page 14.

Disaster and recovery

When a data center is down, all the new writes that happen are tracked in a space-efficient manner on all the remaining locations of the bucket. When the data center is brought up, only the objects that got updated are repaired.

Limitations

The location property of a node cannot be changed. The locations for buckets can be specified only during creation of a bucket and cannot be changed. Specifying a location for a node is optional and can be skipped if you do not want to have a disaster-recovery solution between multiple data centers. If the location is left blank, Veritas Cloud Storage assigns a “Default” location to that node. The location “Default” has a special meaning. The “Default” location refers to the global cluster and all the nodes are now considered to be part of that cluster. Buckets created with a “Default” location store objects across all the nodes irrespective of their location. Also, the data movement for rebalancing is across all the nodes in the cluster.

About erasure coding and how it is implemented in Veritas Cloud Storage

Erasure coding

Erasure coding is a resiliency method that provides enhanced data protection from node and disk failures with better storage efficiency compared with other data protection mechanisms. Erasure coding can be enabled for a bucket during creation and cannot be changed later. As with other bucket properties, erasure coding can also be enabled while creating a namespace to make it the default resiliency method for buckets in that namespace.

Erasure coding is implemented using Vandermonde-based Reed-Solomon codes, which are based on traditional Reed-Solomon codes. Erasure coding requires that the object is split into multiple data chunks (ndata) and a specified number of parity chunks (nparity) are calculated using the Reed-Solomon codes. Veritas Cloud Storage implements erasure coding per individual objects. The computation of parity chunks happens synchronously using Intel’s ISA library. Both data chunks and parity chunks are distributed across multiple failure domains using a consistent hashing algorithm. The nparity value determines the fault tolerance of the erasure-coded object and can be specified during creation of a bucket. The ndata value of each object can vary and is dynamically calculated based on object size. There is a maximum limit on the number of data chunks that can be specified on a configuration file. Due to performance implications, if the object size is <= 4k, the object is always replicated. Also, metadata attributes and indexes of objects are always replicated with the configured fault tolerance at the bucket level.

Consistency

Only strong consistency is supported for erasure-coded objects. A PUT request is not considered complete until all the chunks are written. If there is an error on a column during a write, a PUT fails and an object retains the earlier instance of the

object. If parallel PUT requests are made on the same key, content from one of the PUT requests is present after the completion. A parallel GET request during another PUT can either return an earlier instance or the new instance of the object. A GET after a successful PUT always returns the value that was written with the PUT. As explained in the section on fault tolerance, the consistent hashing implementation is always available for PUTs. So irrespective of the number of faults in the system, PUT is guaranteed to succeed if at least one node is available. The first completed PUT request always wins in the event of a parallel number of PUT requests.

Space saving

An erasure-coding scheme in general is more space efficient compared to replication with similar fault tolerance requirements. As the erasure-coding schema changes with object size, space efficiency depends on the object size. Consider an example configuration with a maximum ndata value configured as 10 and nparity value is specified as 3 for a given bucket.

Table 9-2 Space saving compared to replication

Object size	Ndata	Nparity	Space X	Space saving compared to replication
1k	4	0	4x	0 (Object is replicated with four copies to tolerate three faults)
4k	4	0	4x	0 (Objects <= 4k are replicated)
8k	2	3	2.5x	37% less space
16k	4	3	1.7x	56% less space
32k	8	3	1.3x	65% less space
64k	8	3	1.3x	65% less space
128k	8	3	1.3x	65% less space
256k	8	3	1.3x	65% less space

Note: For the specified configuration, optimal storage efficiency for erasure coding is reached beyond 32k. Veritas Cloud Storage needs to keep some additional metadata persistently and the space-saving calculation does not consider that.

Performance considerations

While an erasure-coding system provides space efficiency in general, it requires additional tuning and may be best suited to certain types of workloads. With smaller object sizes, you should weigh performance considerations regarding space-saving compared to the replica method of resiliency.

Multiple factors play a role in determining the performance of an erasure-coded system:

- Increased IOPS required due to chunking
- Bandwidth reduction due to a smaller transfer of data
- Optimal disk I/O size and network packet size
- Computation of parities during writes
- Computation of read data with a faulted erasure-coded chunk

Veritas Cloud Storage provides tunable configurations to improve efficiency of an erasure-coded system. The default configuration parameters are chosen for a typical hardware configuration. Additional tuning of these parameters may be necessary to get the best performance.

Disk I/O performance and fault-tolerance considerations

The data is always protected from node failures using either replication or erasure coding configured at the namespace or bucket level. In case of dense storage attached to individual nodes, you can avoid node faults due to a disk failure. Consider using a protection mechanism such as configuring RAID-5 or RAID 1+0 at the RAID controller level. You should also consider the performance and storage capacity impact of these protection mechanisms. If it is acceptable to have a node fail due to a disk failure (since the data is already protected using either replication or erasure coding), you should consider RAID-0 striping for I/O performance.

Consult your RAID controller vendor's documentation for the appropriate method to configure the RAID levels.

Fault tolerance

The erasure-coded system implemented in Veritas Cloud Storage tolerates faults up to nparity. The fault tolerance works at the object level. Since Veritas Cloud Storage uses consistent hashing for the distribution of chunks, it is not possible to determine which faults are accounted for in calculating the fault tolerance of an

object. Also, this tolerance is guaranteed for objects that already exist in the system. For new objects, the consistent hashing implementation of Veritas Cloud Storage ensures that even on a faulted system, the required fault tolerance is guaranteed for new PUTs.

About Veritas Cloud Storage weighted distribution

Weighted distribution allows nodes having a higher storage capacity to own a larger share of logical partitions of data. This distribution is supported only if resiliency is guaranteed.

To understand how resiliency guarantee is ensured, consider a 4-node Veritas Cloud Storage cluster, with 1 node 20 TB of storage and 3 nodes with 4 TB of storage configured for Veritas Cloud Storage. With 256 virtual partitions, the node with 20- TB storage should get 160 virtual partitions and each node of 4-TB storage should get 32 virtual partitions. But if the node with 20 TB is faulted, 160 out of 256 virtual partitions are affected. In a 3+1 erasure-coded configuration, this results in a loss of more erasure-coded chunks than the erasure-coded configuration can tolerate.

The system with the 3+1 erasure-coded configuration can only tolerate a failure of 25% of the chunks. So the virtual partition distribution is such that a loss of any node should not result in a loss of more than 25% virtual partitions, irrespective of storage capacity.

To successfully demonstrate the effect of weighted distribution, the cluster should at least have 8 nodes with higher storage capacity than the remaining nodes. For example, an 8-node cluster with 4 nodes having a 4-TB capacity and 4 nodes having a 12-TB capacity. In this case, each of the 4 nodes with the 4-TB capacity has 16 virtual partitions and each of the 4 nodes with the 12-TB capacity has 48 virtual partitions.

About slicing and garbage collection

Object slicing

To optimize performance and resiliency when an object exceeds 32 MB, the object is sliced in multiple chunks. Each chunk has an equal size except for the last one. The last chunk might be smaller if the size of the object is not evenly divisible by the chunk size. For example, with a chunk size of 32 MB, an object of 40 MB results in a 32-MB chunk and an 8-MB chunk.

Multipart upload

Multipart upload consists of the following steps:

- Create the multipart.
- Upload individual parts.
- Complete the multipart upload.

Create the multipart

Multipart takes the following as arguments:

- Namespace
- Bucket

It returns an upload ID, which is used to upload the individual parts. Upload is generated using uuid4. The manifest is written with an upload ID as the key.

Upload individual parts

After receiving the upload ID, individual parts can be uploaded using arguments: {{namespace, bucket}, key, upload_id, part_no, value}. The parts can also be overwritten.

It returns md5 of the part uploaded.

Complete the multipart upload

Multipart upload is completed in this step, the manifest is moved from the upload ID to the actual key.

Arguments: {{namespace, bucket}, key, upload_id, list_of_parts}

Where list_of_parts = [[{:etag, <md5>}, {:part, <part no>}], …….]list_of_parts = [[{:etag, <md5>}, {:part, <part no>}], …….]

Garbage collector

The garbage collector performs the following operations:

- Verification
- Deletion

The garbage collector is a background task that is transparent to the user.

Verification

The garbage collector verifies whether the objects in the ‘dirty’ cache have their chunks written. If the object is verified, its entry is removed from cache and the key prefix is removed. Else, prefix ‘cleanup_delete’ is added to the key.

Deletion

Objects with the prefix ‘cleanup_delete’ are deleted. Before deleting the manifest, the chunks are deleted.

Section 8

Managing Veritas Cloud Storage services

Chapter 10

Service support

This chapter includes the following topics:

Supported services

The following are the supported services:

- Message Queing Telemetry Transport (MQTT) - is a client-server publish and subscription messaging transport protocol. It is light-weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and IoT contexts where a small code footprint is required and or when network bandwidth is at a premium.
  See “About MQTT” on page 66.
- S3 - Standard for cloud object storage, originally popularized by Amazon
  See “S3 API command support matrix” on page 68.
  See “ACL support” on page 73.
- REST - Veritas proprietary API that exposes solution functionality

About MQTT

MQTT is a client-server publish and subscription messaging transport protocol. It is lightweight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments, such as for communication in Machine to Machine (M2M) and the Internet of Things (IoT).

Subscribe

Veritas Cloud Storage supports the MQTT protocol. You can connect to any external broker. You need to provide a topic to listen to or subscribe to while connecting to MQTT along with a bucket name. Specify where you want to store the messages of that topic. All the messages published to that topic are then available in your bucket.

Authentication

If the broker follows any authentication mechanism, you can then connect to it by providing a user name and password. It is recommended to make use of the TLS protocol when connecting to the MQTT broker using a user name and password.

Quality of Service

MQTT makes use of Quality of Service (QoS) to ensure delivery of messages even in an unreliable network. In the Veritas Cloud Storage implementation, QoS 2 is used. QoS 2 guarantees that each message is received only once by the client. It is the safest quality of service level.

Use cases

You want to harness the data coming from various devices in the world of IoT and big data. The biggest challenge is how to effectively store and manage this big data. Veritas Cloud Storage provides such a platform by supporting the MQTT protocol. You can make use of analytics to derive insights from the data stored.

The following are a few use-case examples:

- Healthcare solutions that rely on stored streams of data that require real-time analysis.
- Automobile maintenance that requires collection of real-time data from critical components for taking preventive measures based on that data.

Chapter 11

Supported S3 APIs and ACL support

This chapter includes the following topics:

S3 API command support matrix

The following describes the supported S3 APIs with a description and status for each API.

You can obtain more details by accessing the following URLs:

- s3api cli details
  http://docs.aws.amazon.com/cli/latest/reference/s3api/
- s3api details
  https://docs.aws.amazon.com/AmazonS3/latest/API/

Table 11-1 Supported S3 APIs

API	Description	Status
abort-multipart-upload	Aborts a multipart upload	Available in 1.0.2
complete-multipart-upload	Complete a multipart upload	Available in 1.0.2
copy-object	Copy object (across buckets/same bucket)	Available in 1.0.2
create-bucket	Create bucket	Available in 1.0.2

Table 11-1 Supported S3 APIs *(continued)*

API	Description	Status
create-multipart-upload	Start a multipart upload	Available in 1.0.2
delete-bucket	Delete bucket	Available in 1.0.2
delete-bucket-analytics-configuration	Delete analytic using ID	Not available in 1.0.2
delete-bucket-cors	Delete cross-origin resource sharing	Not available in 1.0.2
delete-bucket-inventory-configuration	Delete inventory configuration	Not available in 1.0.2
delete-bucket-lifecycle	Delete bucket lifecycle	Available in 1.0.2 (cannot change, remove, or modify TTL once set)
delete-bucket-metrics-configuration	Delete metric configuration	Not available in 1.0.2
delete-bucket-policy	Use policy to modify access controls on a bucket	Not available in 1.0.2
delete-bucket-replication	Cross region replication configuration update	Not available in 1.0.2
delete-bucket-tagging	Remove bucket tags set via S3 clients	Not available in 1.0.2
delete-object-tagging	Delete object tag(s)	Available in 1.0.2
delete-objects	Delete objects (using prefix)	Available in 1.0.2
get-bucket-accelerate-configuration	Smart transfer	Not available in 1.0.2
get-bucket-acl	Get the ACLs for a bucket	Available in 1.0.2
get-bucket-analytics-configuration	Get analytics configuration for a bucket	Not available in 1.0.2
get-bucket-cors	List cross-origin resource sharing	Not available in 1.0.2
get-bucket-inventory-configuration		Not available in 1.0.2

Table 11-1 Supported S3 APIs *(continued)*

API	Description	Status
get-bucket-lifecycle	Lifecycle, aging, tier etc,	Available in 1.0.2 (shows TTL configuration for a bucket)
delete-bucket-website	Delete bucket specific website	Not available in 1.0.2
delete-object	Delete object	Available in 1.0.2
get-bucket-lifecycle-configuration	List lifecycle details	Available in 1.0.2 (Veritas Cloud Storage ensures that only aging is allowed as a lifecycle configuration)
get-bucket-location	Get location	Available in 1.0.2 (location is always reported as empty)
get-bucket-logging		Not available in 1.0.2
get-bucket-metrics-configuration		Not available in 1.0.2
get-bucket-notification		Not available in 1.0.2
get-bucket-notification-configuration		Not available in 1.0.2
get-bucket-policy		Not available in 1.0.2
get-bucket-replication		Not available in 1.0.2
get-bucket-request-payment		Available in 1.0.2 (always as bucket owner)
get-bucket-tagging	Get bucket tags	Not available in 1.0.2

Table 11-1 Supported S3 APIs *(continued)*

API	Description	Status
get-bucket-versioning	Get bucket versions	Not available in 1.0.2
get-bucket-website	Get bucket specific website	Not available in 1.0.2
get-object	Get object	Available in 1.0.2
get-object-acl	Get bucket ACLs	Available in 1.0.2
get-object-tagging	Get object tags	Available in 1.0.2
get-object-torrent		Not available in 1.0.2
head-bucket	If bucket is present	Available in 1.0.2
head-object	If object is present	Available in 1.0.2
list-bucket-analytics-configurations		Not available in 1.0.2
list-bucket-inventory-configurations		Not available in 1.0.2
list-bucket-metrics-configurations		Not available in 1.0.2
list-buckets	List buckets with some details	Available in 1.0.2
list-multipart-uploads	Show on-going MP upload	Available in 1.0.2
list-object-versions		Not available in 1.0.2
list-objects	List of objects with some attributes	Available in 1.0.2
list-objects-v2	V2 xml response of list objects	Available in 1.0.2
list-parts	Show part of a specific multipart upload	Available in 1.0.2
put-bucket-accelerate-configuration		Not available in 1.0.2

Table 11-1 Supported S3 APIs *(continued)*

API	Description	Status
put-bucket-acl		Not available in 1.0.2 (incompatibility with UI, set bucket-level ACL via UI)
put-bucket-analytics-configuration		Not available in 1.0.2
put-bucket-cors		Not available in 1.0.2
put-bucket-inventory-configuration		Not available in 1.0.2
put-bucket-lifecycle		Available in 1.0.2 (response is MethodNotAllowed) (Limitation for not being able to change, remove, or modify TTL once set.)
put-bucket-lifecycle-configuration		Not available in 1.0.2
put-bucket-logging		Not available in 1.0.2
put-bucket-metrics-configuration		Not available in 1.0.2
put-bucket-notification		Not available in 1.0.2
put-bucket-notification-configuration		Not available in 1.0.2
put-bucket-policy		Not available in 1.0.2
put-bucket-replication		Not available in 1.0.2

Table 11-1 Supported S3 APIs *(continued)*

API	Description	Status
put-bucket-request-payment	Change request payment	Available in 1.0.2 (response is always defaulted to requester)
put-bucket-tagging	Put tags on bucket	Not available in 1.0.2
put-bucket-analytics-configuration		Not available in 1.0.2
put-bucket-versioning		Not available in 1.0.2
put-bucket-website	Not supported	Not available in 1.0.2
put-object	Put object into bucket	Available in 1.0.2
put-object-acl		Available in 1.0.2
put-object-tagging	Set up tags on objects	Available in 1.0.2
restore-object	Restore from Glacier	Not available in 1.0.2
upload-part		Available in 1.0.2
upload-part-copy		Not available in 1.0.2
wait		Not available in 1.0.2

ACL support

You should not use the S3 s3cmd client to set ACLs, as there is a known bug with the S3 s3cmd client that converts the canonical user ID to lowercase.

You can set object-level ACLs only using the S3 interface.

Table 11-2 Bucket ACLs

Amazon S3	Veritas Cloud Storage
FULL_CONTROL	Bucket Owner
READ	Bucket Reader
WRITE	Bucket Writer
READ_ACP	Bucket Owner
WRITE_ACP	Bucket Owner

Table 11-3 Object ACLs

Amazon S3	Veritas Cloud Storage
FULL_CONTROL	Object Owner
READ	Object Reader
READ_ACP	Object Owner
WRITE_ACP	Object Owner

Section 9

Ecosystems for analyzing your data

Chapter 12

Using ecosystem plugins for analyzing your data

This chapter includes the following topics:

About Apache Thrift

Apache™ Thrift is an open source remote procedure call (RPC) framework from the Apache Foundation for cross-language support.

See the following link for more information on Apache Thrift:

https://thrift.apache.org/

About Apache Hadoop and MapReduce

Apache™ Hadoop is a software framework used for distributed computing. Veritas Cloud Storage provides a way to replace Hadoop's file system (HDFS) with Veritas Cloud Storage.

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. It uses the strategy of split-combine-apply to solve big data problems. The default Hadoop file system has drawbacks, such as it is typically deployed on a single location and has a single point of failure. It also lacks performance scalability for large amounts of data.

Veritas Cloud Storage can serve as the source of data to MapReduce, as it stores both "hot" data (from live applications such as MQTT) as well as "cold" historical

data. Rather than moving the Veritas Cloud Storage data to HDFS for MapReduce jobs, such jobs can be executed directly on Veritas Cloud Storage. Veritas Cloud Storage is scalable while also handling large amounts of data. Since the deployment of Veritas Cloud Storage is on multiple nodes based in different geographies, it is very fault-tolerant and suitable storage platform for MapReduce jobs.

See "Configuring Apache Hadoop" on page 77.

See "Supported Apache Hadoop APIs" on page 80.

Configuring Apache Hadoop

To configure Apache Hadoop

1. Copy `hadoop-connector-x.x.x.jar`, `VikingClient-x.x.jar`, `aws-java-sdk-x.x.x.jar`, and `libthrift-x.xx.x.jar` to the Hadoop cluster servers to some path.

   Example:

   `/usr/local/hadoop/`

2. Set the class path to contain the following jar files:

   ```
   export HADOOP_CLASSPATH=/usr/local/hadoop/hadoop-connector-x.x.x.jar:
   /usr/local/hadoop/VikingClient-x.x.jar:
   /usr/local/hadoop/aws-java-sdk-x.x.x.jar
   :/usr/local/hadoop/libthrift-x.xx.x.jar
   ```

3. Edit the `core-site.xml` file to configure the Veritas Cloud Storage endpoint and Viking and S3A file system classes. A sample `core-site.xml` is provided.
4. Edit the `mapred-site.xml` file to configure directories used for MapReduce jobs. A sample `mapred-site.xml` is provided.
5. Create the namespace and bucket for the Hadoop objects in Veritas Cloud Storage from the UI.

   By default, the namespace user is **user** and the bucket used depends on the user name used to run the Hadoop jobs.

Sample XML files

Location: `$HADOOP_PREFIX/etc/hadoop/core-site.xml`

Sample: `core-site.xml`

```
<?xml version="1.0"?>
```

```
-<configuration>
-<property>
<name>fs.defaultFS</name>
<value>viking://172.17.0.1:2112</value>
</property>
-<property>
<name>io.viking.maxpoolsize</name>
<value>10</value>
</property>
-<property>
<name>io.viking.minpoolsize</name>
<value>2</value>
</property>
-<property>
<name>io.viking.datanode</name>
<value>/hadoop/datanodes/</value>
</property>
-<property>
<name>io.viking.machine</name>
<value>172.17.0.1</value>
</property>
-<property>
<name>io.viking.rest.port</name>
<value>4000</value>
</property>
-<property>
<name>io.viking.port</name>
<value>2112</value>
</property>
-<property>
<name>fs.viking.impl</name>
<value>com.veritas.viking.hadoop.fs.custom.VikingFileSystem</value>
<description>The implementation class of the Viking Filesystem</description>
</property>
-<property>
<name>fs.s3a.impl</name>
<value>com.veritas.viking.hadoop.fs.custom.VikingFileSystem</value>
<description>The implementation class of the Viking Filesystem</description>
</property>
-<property>
<name>fs.default.name</name>
<value>viking:///</value>
</property>
```

```
-<property>
<name>fs.viking.buffer.dir</name>
<value>viking_backup_file_dir</value>
</property>
-<property>
<name>fs.local.block.size</name>
<value>5000</value>
</property>
-<property>
<name>test.fs.viking.name</name>
<value>viking://172.17.0.1:2112</value>
</property>
-<property>
<name>io.file.buffer.size</name>
<value>10000</value>
</property>
-<property>
<name>viking.part.size.threshold</name>
<value>0</value>
</property>
</configuration>
```

Location: `$HADOOP_PREFIX/etc/hadoop/mapred-site.xml`

Sample: `mapred-site.xml`

```
<?xml version="1.0"?>
-<configuration>
-<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>file:///tmp/hadoop-yarn/staging/mapred/.staging</value>
</property>
-<property>
<name>mapred.healthChecker.script.path</name>
<value>viking:///mapred/jobstatus</value>
</property>
-<property>
<name>mapred.job.tracker.history.completed.location</name>
<value>viking:///mapred/history/done</value>
</property>
-<property>
<name>mapred.system.dir</name>
<value>file:///mapred/system</value>
```

```
</property>
-<property>
<name>mapreduce.jobhistory.done-dir</name>
<value>viking:///job-history/done</value>
</property>
-<property>
<name>mapreduce.jobhistory.intermediate-done-dir</name>
<value>viking:///job-history/intermediate-done</value>
</property>
-<property>
<name>mapreduce.jobtracker.staging.root.dir</name>
<value>file:///user</value>
</property>
-<property>
<name>mapreduce.framework.name</name>
<value>local</value>
</property>
</configuration>
```

`log4j.properties` (only needs to be updated to switch off the logs or redirect the logs to the file)

Filename: `$HADOOP_PREFIX/etc/hadoop/log4j.properties`

```
#Root logger option
log4j.rootLogger=INFO, file
# Direct log messages to a log file
log4j.appender.file=org.apache.log4j.RollingFileAppender
#Redirect to Tomcat logs folder
#log4j.appender.file.File=${catalina.home}/logs/logging.log
log4j.appender.file.File=$HADOOP_PREFIX/logigng.log
log4j.appender.file.MaxFileSize=10MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n
```

See “About Apache Hadoop and MapReduce” on page 76.

See “Supported Apache Hadoop APIs” on page 80.

Supported Apache Hadoop APIs

The following are the Apache Hadoop APIs supported by Veritas Cloud Storage:

- Make directory (`mkdir`).

- Put or copyFromLocal.
- List the status (`ls`).
- Get.
- Remove file (`rm`).
- Remove directory (`rmr`).

Testing the Apache Hadoop APIs

MapReduce Java file: `org.myorg.WordCount.java`

Table 12-1 Test commands

Task	Location
Create a directory in Veritas Cloud Storage using `mkdir`.	`bin/hadoop fs -mkdir viking://172.22.29.158:2112/input4`
Move the test file to Veritas Cloud Storage using a PUT command.	`bin/hadoop fs -put julius_caesar.txt viking://172.22.29.158:2112/input4`
List the status of the input file.	`bin/hadoop fs -ls viking://172.22.29.158:2112/input4`
Get the file from Veritas Cloud Storage to your local computer.	`bin/hadoop fs -get viking://172.22.29.158:2112/input4/julius_caesar.txt test_get.txt`
Run the MapReduce on the file present in Veritas Cloud Storage at the input directory and store the results in the output4 folder.	`bin/hadoop org.myorg.WordCount viking://172.22.29.158:2112/input4/julius_caesar.txt viking://172.22.29.158:2112/output4`
List the status of the output file.	`bin/hadoop fs -ls viking://172.22.29.158:2112/output4`
Copy the results from Veritas Cloud Storage to your local computer.	`bin/hadoop fs -get viking://172.22.29.158:2112/output4/part-r-00000`
Delete the output file.	`bin/hadoop fs -rm s3a://192.168.225.200:2112/output4/part-r-00000`

Table 12-1 Test commands *(continued)*

Task	Location
Delete the output folder.	`bin/hadoop fs -rmr s3a://192.168.225.200:2112/output4`

See “About Apache Hadoop and MapReduce” on page 76.

See “Configuring Apache Hadoop” on page 77.

Section 10

Security

Chapter 13

Using encryption for securing your data

This chapter includes the following topics:

About encryption at rest

You can protect your data from being read in case of a disk theft using the encryption at rest feature of Veritas Cloud Storage. The encryption at rest feature does not protect or control access of data for Veritas Cloud Storage users.

Usage

You have the ability to create buckets with encrypted properties. A bucket once created as encrypted cannot be changed and made un-encrypted. Like all other bucket properties, encryption can also be set for the namespace, and can then be overridden for an individual bucket within that namespace.

You can choose an encryption algorithm (AES-256/AES-128) in the `/etc/viking.conf` file of a node. AES-256/AES-128 is considered the industry standard for protecting data at rest. Choose this option before initializing the databases. You cannot change the encryption option once the database is initialized, since this makes the existing databases, which have been encrypted with a different method unreadable.

All the data that is sent to the encrypted bucket is stored encrypted (keys, values, and attributes). Because the encryption key is linked to the administrator's password, the encrypted data cannot be read until the system gets the admin password. Until the password is entered, the operations on the encrypted buckets error out. Since

the MQTT implementation also uses encrypted buckets to store passwords, MQTT does not start until the administrator password is available.

Key management

The AES algorithm requires an encryption key. This key is generated at random when the node is initialized. To secure this key, the key is stored as encrypted. The key is encrypted using another key (key-encryption-key), which is generated from the administrator password using PBKDF2, which is a key derivation function (see https://en.wikipedia.org/wiki/PBKDF2). When Veritas Cloud Storage first starts, the encrypted data is not available until the administrator password is entered on the GUI log on screen. Entering passwords on any of the nodes in the cluster is sufficient to load encrypted data on all the nodes of the cluster. Veritas Cloud Storage also takes care of all other cases like node restarts and node joins, which do not require any separate intervention from you for encrypted data to become available. In case of a password change, Veritas Cloud Storage generates a new key-encryption-key from the new password, and encrypts the actual encryption key with this new key.

About encryption on the wire

Cluster SSL (or Intracluster SSL)

The Cluster SSL feature encrypts data communication taking place between nodes (port 6000) and data handoff taking place through port (8098).

You can enable or disable Cluster SSL by configuring the `/etc/default/vikings` file. Set the `RELEASE_DIST_SSL` option to `true`.

Cluster SSL uses the protocol `inet_tls`. As a consequence, all the nodes in a cluster should have same the Cluster SSL configuration. A mismatch causes communication and handoff errors. You can also update the certificate and key required for Cluster SSL. Restart the Veritas Cloud Storage service after updating the Cluster SSL configuration or certificate and key.

Veritas Cloud Storage uses a self-signed certificate.

Phoenix HTTPS (or web server HTTPS)

Phoenix HTTPS encrypts data requests from the user to Veritas Cloud Storage. HTTPS is on by default and the default port is 443 for HTTPS.

You can enable or disable HTTPS by configuring the `/etc/Viking.conf` file. Set `Viking.https` to `true`.

You can update the certificate and the key for HTTPS. Configuration changes such as updating the certificate and key, and enabling or disabling HTTPS require a restart of the Veritas Cloud Storage service.

Use the following command to restart the Veritas Cloud Storage service:

```
# systemctl restart vrts-viking
```

Veritas Cloud Storage uses a self-signed certificate.

Self-signed certificate

A self-signed certificate and key are generated post-installation using OpenSSL. The certificates are signed using host name as the `localhost`. The private key is RSA 2048 bit. The validity for the certificate is 365 days. You need to update your certificate while it is still valid to avoid getting any client errors.

The certificate and the key are placed in the directory `/opt/VRTScloudstorage/etc`.

While using the self-signed certificate and key, the check for CA certificate needs to be disabled.

Section 11

Applying the latest updates

Chapter 14

Updating Veritas Cloud Storage

This chapter includes the following topics:

Getting the latest updates

If you have the 1.0.1 version of Veritas Cloud Storage installed, you need to do the following:

- Clean the existing system.
- Uninstall the 1.0.1 version of Veritas Cloud Storage.
- Perform a fresh installation of Veritas Cloud Storage 1.0.2.

See the *Veritas Cloud Storage Quick Start Guide* or the deployment steps within this guide to perform a fresh installation of Veritas Cloud Storage 1.0.2.

Section 12

Troubleshooting and maintenance

Chapter 15

Troubleshooting

This chapter includes the following topics:

Viewing the Veritas Cloud Storage log files

You can access the Veritas Cloud Storage logs using the following command:

```
# journalctl -u vrts-viking
```

How to clean up from a failed node

You may need to clean up a failed node. Currently there is no automated way to clean up a server for a failed node.

To clean up a failed node

1. Log on to the node that you want to remove from the cluster.
2. Run the `storage_teardown` tool.

```
# /opt/VRTScloudstorage/bin/storage_teardown
WARNING: This will clean up the Veritas Cloud storage data and cluster
metadata, continue [y/n] ? y
Stopping Veritas Cloud storage service
Redirecting to /bin/systemctl stop vrts-viking.service
Cleaning up storage
Logical volume "VK_HDD_LV" successfully removed
Volume group "VK_HDD_VG" successfully removed
Labels on physical volume "/dev/sda" successfully wiped
Logical volume "VK_SSD_LV" successfully removed
Volume group "VK_SSD_VG" successfully removed
Labels on physical volume "/dev/fioa" successfully wiped
Cleaning up cluster metadata
```

3. Uninstall the package.

```
# yum remove VRTScloudstorage
```

4. Clean up any stale directories, if any.

```
# rm -rf /etc/viking.rpmsave
# rm -rf /opt/VRTS*
# rm -rf /var/viking/
# rm -rf /var/log/viking
# rm -rf /etc/vx/licenses/lic
```

Section 13

Reference

Appendix A

This appendix includes the following topics:

Using the Veritas Cloud Storage documentation

The latest version of the Veritas Cloud Storage product documentation is available on the Veritas Services and Operations Readiness Tools (SORT) website.

https://sort.veritas.com/documents

You need to specify the product and the platform and apply other filters for finding the appropriate document.

Make sure that you are using the current version of documentation. The document version appears on page 2 of each guide. The publication date appears on the title page of each document. The documents are updated periodically for errors or corrections.

The following documents are available on the SORT website:

- *Veritas Cloud Storage Deployment Guide*
- *Veritas Cloud Storage Quick Start Guide*
- *Veritas Cloud Storage Release Notes*
- *Veritas Cloud Storage Third-Party License Agreements*

See “About Veritas Cloud Storage” on page 9.

Index

R

S

T

U

V

W