HADOOP TIERED STORAGE WITH DELL EMC ISILON AND DELL EMC ECS CLUSTERS

March 2021

Abstract

This solution guide describes how to easily expand storage to existing DAS Hadoop clusters with Dell EMC Isilon and Dell EMC ECS systems to provide immediate capacity, better storage efficiency, and reduced total cost of ownership.

H16659.3

SOLUTION GUIDE

Copyright

Published in the USA 03/21 Solution Guide H16659.3.

Contents

Chapter 1 Executive Summary

This chapter presents the following topics:

Business case

Enterprises implementing digital transformation initiatives and data-driven decision-making often must deal with exponential data growth that is not provided for in their IT budgets. For most enterprises, most of this data growth is “cold data,” which is historical in nature and does not require frequent or low-latency access. The remainder of the data growth is in “hot data,” which is recently generated data that requires frequent and low-latency access.

A Hadoop solution that consists of a hot tier and a cold tier enables the enterprise to store hot data in a high-throughput, low-latency cluster with low cost per MB/s and cold data in a capacity-dense cluster with low cost per TB.

Solution overview

This Hadoop tiered storage solution provides an architecture that can support cross-namespace analytics. With this solution, you can use both direct-attached storage (DAS) and an alternate storage media such as Dell EMC Isilon and Dell EMC Elastic Cloud Storage (ECS) storage, and run analytics jobs and toolsets across data that spans these storage tiers.

The Hadoop tiered storage solution from Dell EMC enables:

- Cold data storage in a shared storage cluster that is based on the Isilon or ECS system, providing outstanding capacity density and low cost per TB.
- Hot data storage in a DAS cluster that is based on the Dell EMC PowerEdge server, which delivers high performance and low cost per MB/s.
- Processing of data by Yarn- or Mesos-based Hadoop applications across both clusters, which are subject to data governance, risk management, and compliance management. The DAS and Isilon clusters represent separate namespaces, so Hadoop applications and governance run on the federated namespace.

Deployment options are as follows:

- Customers who have existing Hadoop clusters running DAS and who need to expand their Hadoop clusters to hundreds of TBs or PBs can add an Isilon or ECS cluster to their existing Hadoop cluster to handle the high volume of data growth.
- Customers who plan to deploy a large Hadoop data lake can build the Hadoop tiered storage solution with DAS and Isilon or ECS clusters.

Key results

Dell EMC and Hortonworks have validated multiple configurations for Hadoop tiered storage with a logical Hadoop cluster (DAS storage) and an infrastructure cluster (Isilon or ECS system) that meet or exceed the functional objectives of this solution. You can match most needs with an approved configuration. By combining the Hortonworks Data Platform (HDP) cluster (logical Hadoop cluster) with the flexibility of an Isilon or ECS infrastructure cluster, you can scale the solution to handle future requirements without extensive upgrades or expensive replatforming.

Document purpose

This solution guide provides detailed information for evaluating the applicability of Hadoop tiered storage for your environment. The guide provides solution validation, including results of rigorous testing of the major components of the Hadoop cluster and their functionality in the tiered storage environment.

Audience

This guide is for IT administrators, storage administrators, virtualization administrators, system administrators, IT managers, and those who evaluate, acquire, manage, maintain, or operate Hadoop cluster environments.

We value your feedback

Dell EMC and the author of this document welcome your feedback on the Ready Stack and the Ready Stack documentation. Contact the Dell EMC Solutions team by email or provide your comments by completing our documentation survey.

Authors: Boni Bruno, Kirankumar Bhusanurmath, Tao Guo, Eric Wang, Karen Johnson

Chapter 2 Technology Overview

This chapter presents the following topics:

Reference architecture

Figure 1 shows the reference architecture of Hadoop tiered storage with an Isilon or ECS system. This reference architecture provides for hot-tier data in high-throughput, low-latency local storage and cold-tier data in capacity-dense remote storage. You can deploy the Hadoop cluster on physical hardware servers or a virtualization platform.

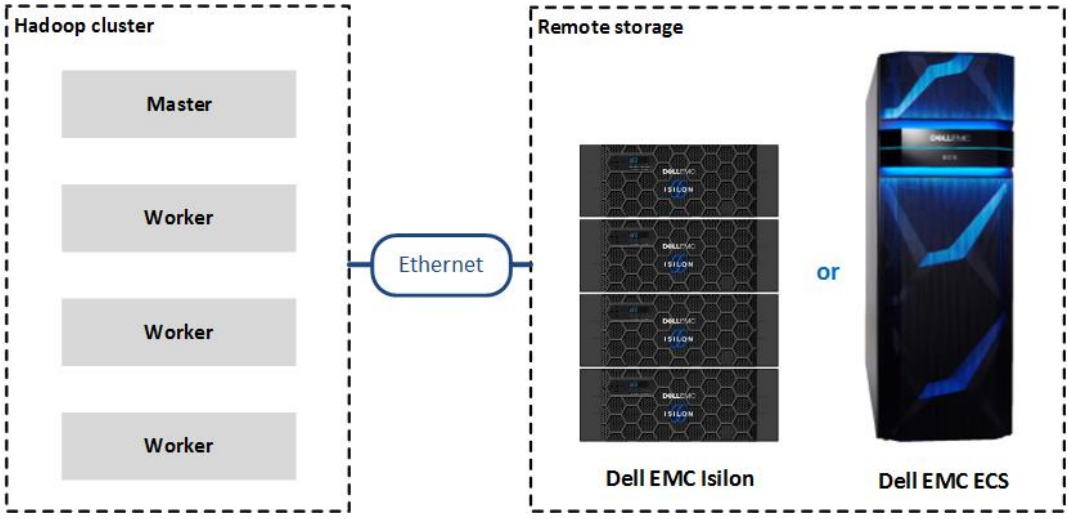


Figure 1. Reference architecture of Hadoop tiered storage with an Isilon or ECS system

Figure 2 shows the high-level reference architecture of Hadoop tiered storage with an Isilon cluster.

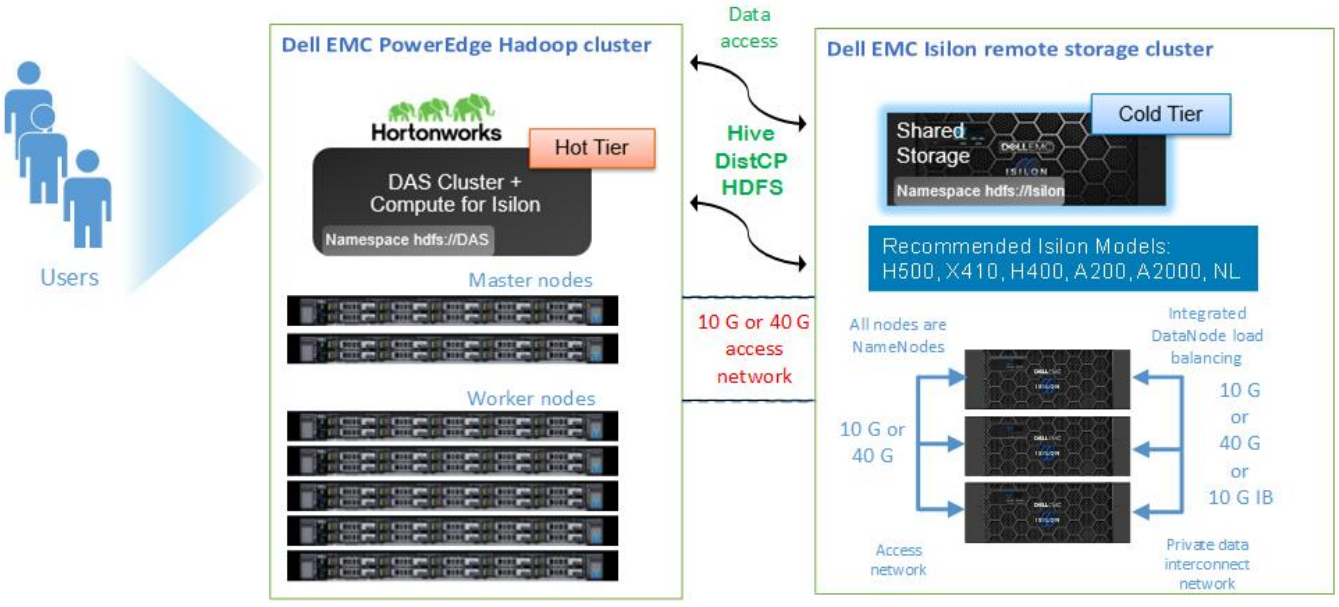


Figure 2. Reference architecture of Hadoop tiered storage with an Isilon cluster

Key components

Dell EMC Isilon

The Dell EMC Isilon scale-out network-attached storage (NAS) platform provides Hadoop clients with direct access to Big Data through a Hadoop File System (HDFS) interface. Powered by the distributed Dell EMC Isilon OneFS operating system, an Isilon cluster delivers a scalable pool of storage with a global namespace. The distributed OneFS operating system combines the memory, I/O, CPUs, and disks of the nodes into a cohesive storage unit to present a global namespace as a single file system.

Hadoop compute clients access the data that is stored in an Isilon cluster by using the HDFS protocol. Every node in the cluster can act as a NameNode and a DataNode. Each node boosts performance and expands the cluster's capacity. For Hadoop analytics, the Isilon scale-out distributed architecture minimizes bottlenecks, rapidly serves big data, and optimizes performance for analytics jobs. The NameNode daemon is a distributed process that runs on all the nodes in the cluster. A compute client can connect to any node in the cluster to access NameNode services. The nodes work together as peers in a shared-nothing hardware architecture with no single point of failure.

An Isilon cluster is platform agnostic for compute. You can run most of the common Hadoop distributions with an Isilon cluster. Clients running different Hadoop distributions or versions can simultaneously connect to the cluster.

Dell EMC ECS

The Dell EMC ECS platform is a complete software-defined cloud storage system that supports the storage, manipulation, and analysis of unstructured data on a massive scale on commodity hardware. You can deploy the ECS platform as a turnkey storage appliance or as a software product on a set of qualified commodity servers and disks. The ECS platform offers the cost advantages of a commodity infrastructure and the enterprise reliability, availability, and serviceability of traditional arrays.

The ECS scalable architecture includes multiple nodes and attached storage devices. The nodes and storage devices are commodity components, similar to devices that are generally available, and are housed in one or more racks.

An ECS appliance consists of a rack, rack components, and preinstalled software that are supplied by Dell EMC. An ECS software-only solution uses a rack and commodity nodes that are not supplied by Dell EMC. A cluster consists of multiple racks.

ECS HDFS is a Hadoop Compatible File System (HCFS) that enables you to run Hadoop 2.*x* applications on top of your ECS infrastructure. You can configure your Hadoop distribution to run against the built-in Hadoop file system, ECS HDFS, or any combination of HDFS, ECS HDFS, or other HCFSs available in your environment.

Hortonworks Data Platform

HDP is an enterprise-level, hardened Hadoop distribution that combines the most useful and stable versions of Apache Hadoop and its related projects into a single tested and certified package. HDP enables Enterprise Hadoop by providing a complete set of essential Hadoop capabilities. It delivers the core elements of Hadoop—scalable storage and distributed computing—as well as all of the necessary enterprise capabilities such as security, high availability, and integration with a broad range of hardware and software solutions.

Ambari

Apache Ambari is a utility that provides installation, monitoring, and management capabilities for an HDP cluster. The Ambari web client and REST APIs are used to deploy, operate, manage, and monitor the HDP cluster.

Kerberos

Kerberos is a network authentication protocol designed to provide strong authentication for client/server applications by using secret-key cryptography in most distributed systems, including HDP. Kerberos provides secure and reliable authentication to multiple applications. Isilon and ECS systems support the Kerberos authentication feature using Kerberos Key Distribution Center (KDC) services.

Ranger

Apache Ranger is a centralized management console that enables you to monitor and manage data security across the Hortonworks Hadoop distribution system. A Ranger administrator can define and apply authorization policies across Hadoop components including HDFS. Isilon OneFS 8.0.1.0 and later releases support Ranger HDFS policies. In an Isilon OneFS cluster with Hadoop deployment, Ranger authorization policies serve as a filter before the application of native file access control.

Software resources

Table 1 lists the solution software resources.

Table 1. Software resources

Software	Version
Red Hat Enterprise Linux 64-bit	7.2
Apache Ambari	2.6.0.0
Hortonworks Data Platform	2.6.3.0
MIT Kerberos	5
Dell EMC OneFS	8.0.1.1
Dell EMC ECS HDFS Client	3.0.0.0

Chapter 3 Solution Design

This chapter presents the following topics:

Deployment best practices

Spreading data to an Isilon or ECS cluster

Spread data to an Isilon or ECS cluster when cold data grows beyond 75 TB but is below 64 PB.

For 1 PB of usable data:

- The acquisition cost of a Hadoop cluster with an Isilon or ECS cluster equals 60 percent of DAS.
- The rack space of a Hadoop cluster with an Isilon or ECS cluster equals 40 percent of DAS.

For more than 1 PB of usable data:

- The acquisition cost of a Hadoop cluster with an Isilon or ECS cluster could be more than 60 percent of DAS.

Partitioning large tables

Partition tables if you collect time series data or logs that accumulate over time and you only need to query parts of the data. You can store the data in a subdirectory tree such as year/month/day, continent/country/region/city, and so on, enabling your query to skip the irrelevant data.

ORCFile format for Hive tables

Hive supports ORCFile, a new table storage format that provides significantly increased speed through techniques such as predicate push-down, compression, and more. Using ORCFile for every Hive table provides fast response times for your Hive queries.

Directory structure considerations

You can use directory structures to organize data by department, business unit, lifecycle stage (new versus old, hot versus cold, raw versus derived), or other business concerns. Access control is an important consideration as well, especially in multitenant environments.

Unlike more advanced traditional DBMS access-control models where you can carve up access based on metadata, HDFS is a distributed filesystem; directories can represent your metadata.

Tools like Hive understand partition pruning during query execution. Each partition is simply a directory with a special naming convention that indicates the range of the table to which the contained data belongs (at least in range-based partitioning). Tools other than Hive can have similar partition pruning simply by including only the directories that are known to contain data of interest.

Key directories to be aware of include:

- /user/<username>—Home directories/scratch pads for users
- /tmp—Sticky-bit set scratch for tools and users (no guarantee on longevity)
- /data—Canonical, raw data sets ingested from other systems/applications

For example:

`/data/<dataset name>/<optional partitions>`

where <*dataset name*> is the equivalent of a table name in an RDBMS. Optionally, data sets can be partitioned by *n* columns, depending on the use case.

Partitioned security log data by day example:

```
/data/seclogs/date=20170101/{x.avro,y.avro,z.avro}
/data/seclogs/date=20170102/{x.avro,y.avro,z.avro}
```

ETL directory example:

`/etl/<group>/<application>/<process>/{incoming,working,complete,failed}`

where <*group*> is the line of business/group (research, search quality, fraud analysis), <*application*> is the name of the application the process supports, and <*process*> is for applications that have multiple processing stages. Each process "queue" could have four state directories. For example:

- `incoming`—Newly arriving files drop off here. A process automatically renames them into a temp directory under `working` to indicate that they are in progress.
- `working`—This directory contains a timestamped directory for each attempt at processing the files. Files in these directories that are older than *x* require human intervention.
- `complete`—After an ETL process finishes processing a file in `working`, this is where it could land.
- `failed`: If an ETL process permanently rejects a file, it moves the file here. If the directory contains > 0 files, it requires human intervention.

This example of an ETL directory structure shows four scenarios only. You could extend the structure for your particular use cases.

The general idea is to develop a directory structure to support a data lifecycle that can be controlled by directories for partitions, ETL processes, user data, and the like.

You can apply access control to individual processes, groups, applications, or data sets. Even partitions can be separately controlled in terms of access (on user type or line of business for data sets, for example).

Directory structure design is a complex topic. Dell EMC offers professional services to assist in directory structure design as well as other Hadoop-related services.

Hadoop tiered storage with an Isilon or ECS cluster

Overview

The solution architectures of Hadoop with Isilon and Hadoop with ECS enable you to run analytics jobs and toolsets on data that is spread across both DAS and Isilon or ECS storage tiers.

Figure 3 shows the Hadoop with Isilon solution architecture. Figure 4 shows the Hadoop with ECS solution architecture.

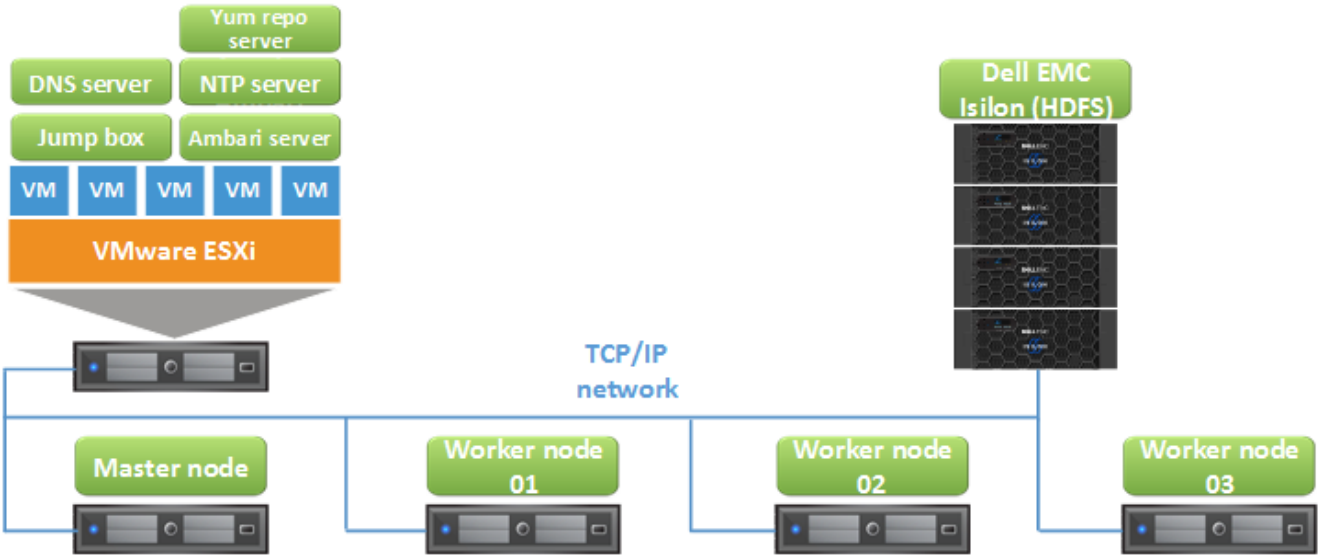


Figure 3. Hadoop tiered storage with Isilon solution architecture

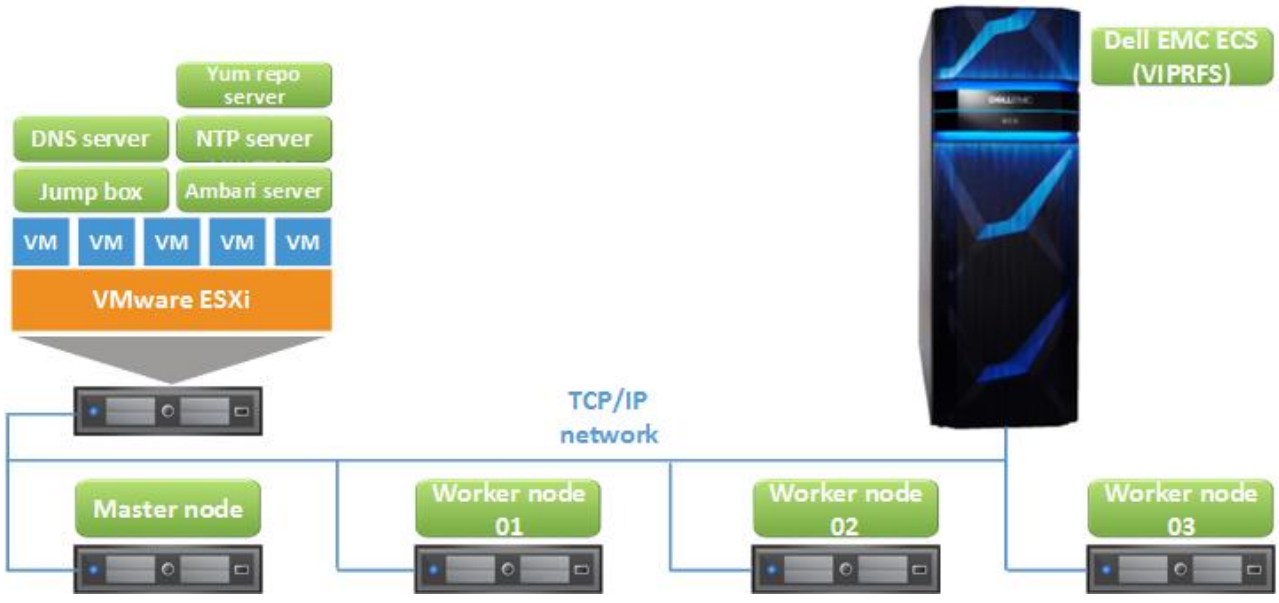


Figure 4. Hadoop tiered storage with ECS solution architecture

Hadoop cluster design

Table 2 describes the Hadoop cluster nodes, their roles, and the services running on them.

Table 2. Hadoop cluster services and instance roles

Host	Instance role	Services on the node
hdp-ambari.bigdata.emc.local	Ambari Server	Hortonworks SmartSense Tool (HST) Agent
		Kerberos Client
		Metrics Monitor
		SNameNode
hdp-master.bigdata.emc.local	HDP Master/HDP Client	Activity Analyzer
		Activity Explorer
		App Timeline Server
		HCat Client
		HDFS Client
		History Server
		Hive Client
		Hive Metastore
		HiveServer2
		HST Agent
		HST Server
		Infra Solr Client
		Infra Solr Instance
		Kerberos Client
		MapReduce2 Client
		Metrics Collector
		Grafana
		Metrics Monitor
		NameNode
		Pig Client
		Ranger Admin
		Ranger Tagsync
		Ranger Usersync
		ResourceManager
		Slider Client
		Spark2 Client

Host	Instance role	Services on the node
		Spark2 History Server
		Spark2 Thrift Server
		Tez Client
		WebHCat Server
		YARN Client
		ZooKeeper Client
hdp-worker01.bigdata.emc.local	Worker node 01	DataNode
		HCat Client
		HDFS Client
		Hive Client
		HST Agent
		Infra Solr Client
		Kerberos Client
		MapReduce2 Client
		Metrics Monitor
		NodeManager
		Pig Client
		Slider Client
		Spark2 Client
		Tez Client
		YARN Client
		ZooKeeper Client
		ZooKeeper Server
hdp-worker02.bigdata.emc.local	Worker node 02	DataNode
		HCat Client
		HDFS Client
		Hive Client
		HST Agent
		Infra Solr Client
		Kerberos Client
		MapReduce2 Client
		Metrics Monitor
		NodeManager

Host	Instance role	Services on the node
		Pig Client
		Slider Client
		Spark2 Client
		Tez Client
		YARN Client
		ZooKeeper Client
		ZooKeeper Server
hdp-worker03.bigdata.emc.local	Worker node 03	DataNode
		HCat Client
		HDFS Client
		Hive Client
		HST Agent
		Infra Solr Client
		Kerberos Client
		MapReduce2 Client
		Metrics Monitor
		NodeManager
		Pig Client
		Slider Client
		Spark2 Client
		Tez Client
		YARN Client
		ZooKeeper Client
		ZooKeeper Server

Chapter 4 Hadoop Cluster Deployment and Integration with Isilon Cluster

This chapter presents the following topics:

Overview

Table 3 lists the process flow for the Hadoop cluster deployment with an Isilon cluster.

Table 3. Hadoop cluster deployment and integration with Isilon cluster

Step	Action
1	Set up the HDP cluster
2	Set up the Isilon cluster
3	Create an Isilon access zone
4	Enable Kerberos on the HDP cluster
5	Enable Kerberos on the Isilon cluster
6	Enable Ranger and set policies
7	Validate HDP deployment and Isilon integration

Setting up the HDP cluster

Installing Ambari Server

Ambari Server automates the installation and configuration of HDP regardless of scale or deployment environment. It also helps to manage and monitor the Apache Hadoop cluster and provides an intuitive Hadoop management web UI.

Before you begin HDP cluster deployment, set up Ambari Server. For this solution, we set up Ambari Server using a virtual machine on one shared ESXi host.

The following steps provide instructions for setting up Ambari Server. For more details about the installation process, see Apache Ambari Installation on the Hortonworks website.

1. Find an available physical server or virtual machine to host Ambari Server.
2. Install RHEL 7.2 or later using the default installation option, **Minimal Install**.
3. Set up the IP address, netmask, and hostname.
4. Log in to the server using the root account.
5. Create an Ambari Server local repository configuration file (`/etc/yum.repos.d/ambari.repo`).

Note: We created `http://public-repo-1.hortonworks.com/HDP/centos7/2.x/updates/2.6.3.0/hdp.repo` and `http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.6.0.0/ambari.repo` as the local repository for the Ambari Server 2.6.0.0 and HDP 2.6.3.0 packages obtained from the Hortonworks public repository.

6. Generate SSH key pairs without passwords:

```
clear &&
ssh-keygen &&
cd /root/.ssh &&
cat id_rsa.pub >> authorized_keys &&
chmod 600 /root/.ssh/authorized_keys &&
ll /root/.ssh/authorized_keys &&
echo "done"
```

7. Copy the passwordless SSH key pairs to all Hadoop nodes:

```
clear &&
ssh root@hdp-master01 "mkdir -p /root/.ssh && chmod 700 /root/.ssh"
&& scp /root/.ssh/authorized_keys root@hdp-master01:/root/.ssh/ &&
ssh root@hdp-worker01 "mkdir -p /root/.ssh && chmod 700 /root/.ssh"
&& scp /root/.ssh/authorized_keys root@hdp-worker01:/root/.ssh/ &&
ssh root@hdp-worker02 "mkdir -p /root/.ssh && chmod 700 /root/.ssh"
&& scp /root/.ssh/authorized_keys root@hdp-worker02:/root/.ssh/ &&
ssh root@hdp-worker03 "mkdir -p /root/.ssh && chmod 700 /root/.ssh"
&& scp /root/.ssh/authorized_keys root@hdp-worker03:/root/.ssh/ &&
echo "done"
```

8. Test the passwordless SSH key pairs and ensure that no errors occur:

```
clear &&
ssh root@hdp-master01 "hostname" &&
ssh root@hdp-worker01 "hostname" &&
ssh root@hdp-worker02 "hostname" &&
ssh root@hdp-worker03 "hostname" &&
echo "done"
```

9. Make a copy of the passwordless SSH private key, which is used later during HDP deployment.

10. Install the Ambari bits:

```
yum install ambari-server
```

The installation also installs the default PostgreSQL Ambari database.

11. Set up Ambari Server:

```
ambari-server setup
```

12. Start Ambari Server, check its status, and then stop it:

```
ambari-server start
ambari-server status
ambari-server stop
```

Installing HDP

After you complete the Ambari Server installation, install HDP.

The following steps provide instructions for installing HDP 2.6.3.0. For more details about the installation process, see Apache Ambari Installation on the Hortonworks website.

1. Start Ambari Server:

```
ambari-server start
```

2. Log in to Apache Ambari and click **Launch Install Wizard**.

3. At the prompt, type a name for the cluster.

4. Specify the following:
   - **HDP-2.6—HDP-2.6.3.0**
   - **Use Local Repository**
   - **OS—redhat7**
   - **HDP-2.6**—*<your local repository for the HDP packages>*
5. On the **Install Options** page, as shown in Figure 5, type the requested information.

   In the **Target Hosts** text box, type the Fully Qualified Domain Name (FQDN) of each of your hosts. The wizard also needs to access the private key file you created when you set up password-less SSH. Using the host names and key file information, the wizard can locate, access, and interact securely with all hosts in the cluster.

Figure 5. Install Options page of the Cluster Install Wizard

6. On the **Confirm Hosts** page, confirm that Ambari has located the correct hosts for your cluster and that they have the correct directories, packages, and processes required to continue the installation.

   If you previously selected any hosts in error, remove them by selecting the corresponding checkbox and clicking the grey **Remove Selected** button. To remove a single host, click the white **Remove** button in the **Action** column.
7. On the **Choose Services** page, as shown in Figure 6, select the services you want to install.

   The wizard presents a list of services that you can install in the cluster, based on the selected stack. You can choose to install any available services now, or you can add services later. The wizard selects all available services for installation by default.

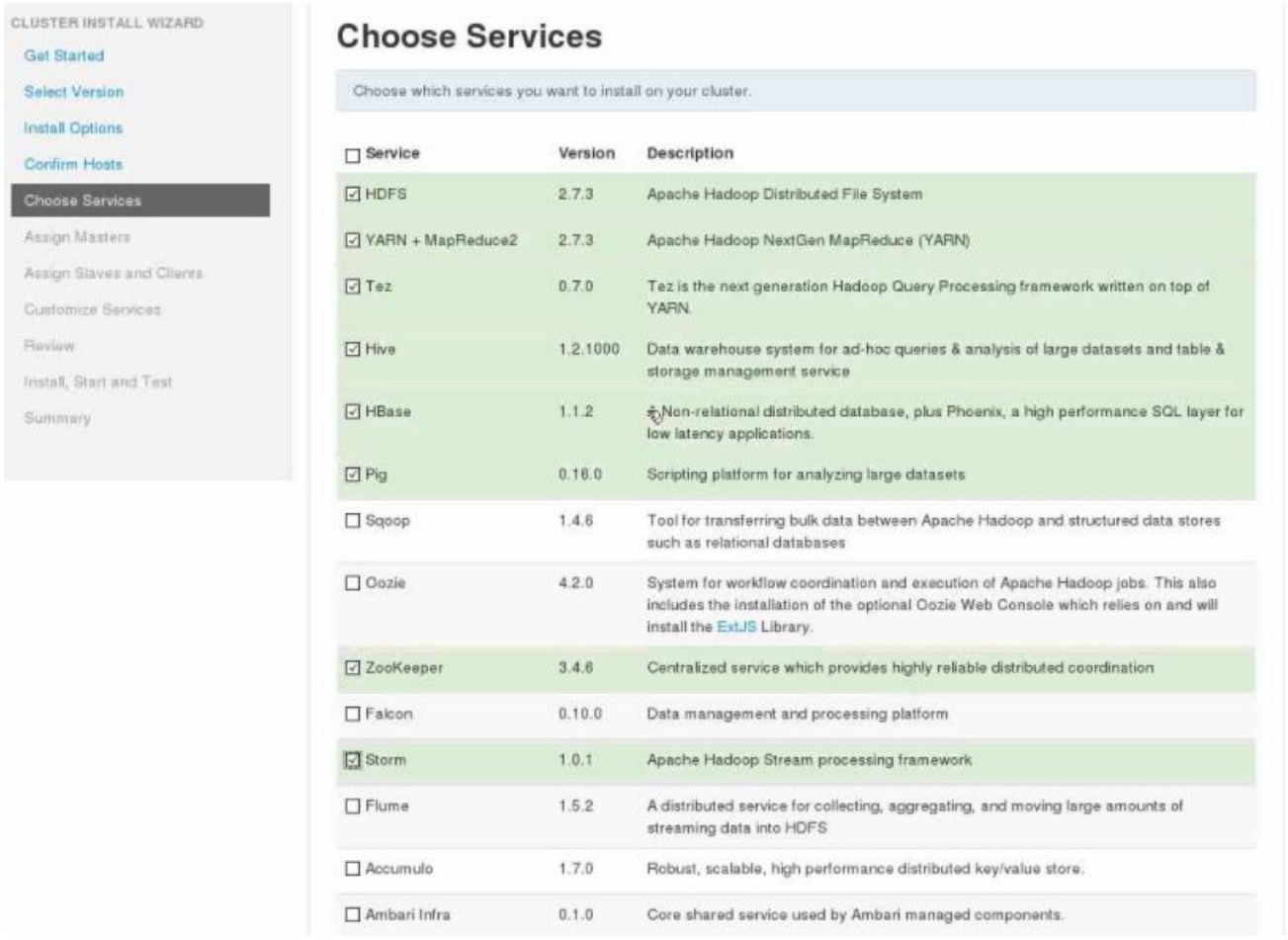


Figure 6. Choose Services page of Cluster Install Wizard

8. On the **Assign Masters** page, verify the host assignments, make changes as needed, and then click **Next**.

   The wizard assigns the master components for selected services to appropriate hosts in your cluster and displays the assignments on this page. The column on the left shows services and current hosts. The column on the right shows current master component assignments by host, indicating the number of CPU cores and amount of RAM installed on each host.

   To change the host assignment for a service, select a host name from the list box for the service.

9. On the **Assign Slaves and Clients** page, verify the assignments, make changes as needed, and then click **Next**.

   The wizard assigns the slave components, such as DataNodes, NodeManagers, and RegionServers, to appropriate hosts in your cluster. It also attempts to select hosts on which to install the set of clients.

10. On the **Customize Services** page, review the cluster setup, modify it as needed, and then click **Next**.

    The wizard displays tabs that let you review and modify your cluster setup. The wizard attempts to set reasonable defaults for each of the options.

11. On the **Review** page, verify that the displayed information is correct, and then click **Next**.

    To make changes, use the navigation bar on the left to return to a previous screen.

12. On the **Install, Start and Test** page, click **Next** when the installation is complete.

    Ambari installs, starts, and runs a simple test on each component. The wizard displays the overall status of the process in the progress bar at the top of the page, and it displays host-by-host status in the main section of the page.

13. On the **Summary** page, click **Complete**.

    The Ambari web console opens in your web browser.

Setting up the Isilon cluster

To set up the Isilon cluster infrastructure, after setting up the HDP cluster, contact your Dell EMC or partner representative.

Creating Isilon access zones

On one of the Isilon OneFS cluster nodes, define access zones and enable the Hadoop node to connect to them:

1. On a node in the Isilon OneFS cluster, create two Hadoop access zones—hdfs1 and hdfs2.

   ```
   isi zone zones create --name=hdfs1 --path=/ifs/data/hdfs1 --
   create-path
   isi zone zones create --name=hdfs2 --path=/ifs/data/hdfs2 --
   create-path
   ```

2. Verify that the access zones are set up correctly:

   ```
   isi zone zones view hdfs1
   isi zone zones view hdfs2
   ```

3. Create the HDFS root directory within the access zones that you created:

   ```
   mkdir -p /ifs/data/hdfs1/hdfs
   mkdir -p /ifs/data/hdfs2/hdfs
   ```

4. List the contents of the Hadoop access zone root directory:

```
ls -al /ifs/data/hdfs1
ls -al /ifs/data/hdfs2
```

5. Create an access zone:

```
isi network pools create groupnet0:subnet0:pool1 --
ranges=172.16.1.241-172.16.1.250 --access-zone=hdfs1 --
alloc-method=static --ifaces=1-3:ext-1 --sc-subnet=subnet0 -
-sc-dns-zone=isi-cluster-hdfs1.bigdata.emc.local --
description="hdfs1 hdfs access zone"
isi network pools create groupnet0:subnet0:pool2 --
ranges=172.16.1.211-172.16.1.220 --access-zone=hdfs2 --
alloc-method=static --ifaces=1-3:ext-1 --sc-subnet=subnet0 -
-sc-dns-zone=isi-cluster-hdfs2.bigdata.emc.local --
description="hdfs2 hdfs access zone"
```

6. View the properties of the existing pool:

```
isi network pools view groupnet0.subnet0.pool1
isi network pools view groupnet0.subnet0.pool2
```

Configuring the Isilon HDFS for OneFS 8.1.2 and previous versions

In OneFS 8.1.2, the HDFS user must be mapped to root and you must modify the access control list (ACL).

On a node in the Isilon OneFS cluster, create and configure the HDFS root directory:

1. Set the HDFS root directory for the access zones:

```
isi hdfs settings modify --zone=hdfs1 --root-
directory=/ifs/data/hdfs1/hdfs
isi hdfs settings modify --zone=hdfs2 --root-
directory=/ifs/data/hdfs2/hdfs
```

2. View the HDFS service settings:

```
isi hdfs settings view --zone=hdfs1
isi hdfs settings view --zone=hdfs2
```

3. Map the HDFS user to Isilon root. Create a user mapping rule to map the HDFS user to the OneFS root account.

   This mapping enables the services from the Hadoop cluster to communicate with the OneFS cluster using the correct credentials.

```
isi zone zones modify --user-mapping-rules="hdfs=>root" --
zone hdfs1
isi zone zones modify --user-mapping-rules="hdfs=>root" --
zone hdfs2
```

4. Modify the ACL settings for the OneFS cluster.

   Before you create directories or files, type the following commands on a node in the Isilon OneFS cluster to modify ACL settings. This modification creates the correct permission behavior on the cluster for HDFS.

   Note: Because ACL policies are cluster-wide, ensure that you understand this change before performing it on production clusters.

```
isi auth settings acls modify --group-owner-
inheritance=parent
isi auth settings acls view
```

Configuring Isilon HDFS for OneFS 8.2.0 and later

In OneFS 8.2.0, the HDFS user no longer has to be mapped to root. Instead, a new role with backup and restore privileges must be assigned.

On a node in the Isilon OneFS 8.2 cluster, create a role and configure the backup and restore privileges to the HDFS user as follows:

1. Set the HDFS root directory for the access zone:

```
isi hdfs settings modify --zone=hdfs1 --root-
directory=/ifs/data/hdfs1/hdfs

isi hdfs settings modify --zone=hdfs2 --root-
directory=/ifs/data/hdfs2/hdfs
```

2. Create a new role for the Hadoop access zone:

```
isi auth roles create --name=<role_name> --
description=<role_description> --zone=<access_zone>
```

   For example:

```
isi auth roles create --name=HdfsAccess --
description="Bypass FS permissions" --zone=hdfs1

isi auth roles create --name=HdfsAccess --
description="Bypass FS permissions" --zone=hdfs2
```

3. Add restore privileges to the new HdfsAccess role:

```
isi auth roles modify <role_name> --add-
priv=ISI_PRIV_IFS_RESTORE --zone=<access_zone>
```

   For example:

```
isi auth roles modify HdfsAccess --add-
priv=ISI_PRIV_IFS_RESTORE --zone=hdfs1

isi auth roles modify HdfsAccess --add-
priv=ISI_PRIV_IFS_RESTORE --zone=hdfs2
```

4. Add backup privileges to the new HdfsAccess role:

```
isi auth roles modify <role_name> --add-
priv=ISI_PRIV_IFS_BACKUP --zone=<access_zone>
```

For example:

```
isi auth roles modify HdfsAccess --add-
priv=ISI_PRIV_IFS_BACKUP --zone=hdfs1
isi auth roles modify HdfsAccess --add-
priv=ISI_PRIV_IFS_BACKUP --zone=hdfs2
```

5. Add user hdfs to the new HdfsAccess role:

```
isi auth roles modify <role_name> --add-user=hdfs --
zone=<access_zone>
```

For example:

```
isi auth roles modify HdfsAccess --add-user=hdfs --
zone=hdfs1
isi auth roles modify HdfsAccess --add-user=hdfs --
zone=hdfs2
```

6. Verify the role setup, backup/restore privileges, and HDFS user setup:

```
isi auth roles view <role_name> --zone=<access_zone>
```

For example:

```
isi auth roles view HdfsAccess --zone=hdfs1
Name: HdfsAccess
Description: Bypass FS permissions
Members: - hdfs
Privileges
ID: ISI_PRIV_IFS_BACKUP
Read Only: True
ID: ISI_PRIV_IFS_RESTORE
Read Only: True
```

7. (Optional) Flush auth mapping and auth cache to make the HDFS user take immediate effect as the HdfsAccess role:

```
isi_for_array "isi auth mapping flush --all"
isi_for_array "isi auth cache flush --all"
```

Note: For OneFS 8.2 and later, ACL policies do not have to be modified. The HDFS protocol acts the same as other HDFS implementations for file system group owner inheritance.

Creating HDFS users and groups

This methodology achieves UID/GID parity by executing user creation in the following sequence:

1. Create local users and groups on Isilon OneFS.
2. Collect the UIDs and GIDs of the users.
3. Create local users and groups on all HDP hosts to be deployed.

Create HDFS users and groups as follows:

1. On a node in the Isilon OneFS cluster, create scripts directories:

   ```
   mkdir -p /ifs/data/hdfs1/scripts

   mkdir -p /ifs/data/hdfs2/scripts
   ```

   You will extract the scripts to this directory.

2. Clone or download the latest version of the Isilon Hadoop tools as a Zip file from https://github.com/Isilon/isilon_hadoop_tools to the `/ifs/zone1/hdp/scripts` directory.

3. Unzip and upload `isilon_create_users.sh` and `isilon_create_directories.sh`.:

   ```
   /ifs/data/hdfs1/scripts

   /ifs/data/hdfs2/scripts
   ```

4. Run the following script to create all required local users and groups on your Isilon OneFS cluster for the Hadoop services and applications:

   ```
   bash /ifs/data/hdfs1/scripts/isilon_create_users.sh --dist
   hwx --startuid 1000 --startgid 1000 --zone hdfs1

   bash /ifs/data/hdfs2/scripts/isilon_create_users.sh --dist
   hwx --startuid 1000 --startgid 1000 --zone hdfs2
   ```

5. To create directories to map to the Hadoop users with appropriate ownership and permissions, download `isilon_create_directories.sh` from https://github.com/Isilon/isilon_hadoop_tools and run the following script:

   ```
   bash /ifs/data/hdfs1/scripts/isilon_create_directories.sh --
   dist hwx --zone hdfs1 –fixperm

   bash /ifs/data/hdfs2/scripts/isilon_create_directories.sh --
   dist hwx --zone hdfs2 --fixperm
   ```

6. Test accessibility from the primary cluster:

   ```
   hadoop fs -ls hdfs://isi-cluster-
   hdfs1.bigdata.emc.local:8020/

   hadoop fs -ls hdfs://isi-cluster-
   hdfs2.bigdata.emc.local:8020/
   ```

Enabling Kerberos on the HDP cluster

Ambari provides a wizard to help enable Kerberos on the cluster. This section provides information about preparing Ambari before running the wizard and the steps to run the wizard.

Preparing Ambari

Prepare Ambari as follows:

1. Ensure that you are running Ambari 2.0 or later.
2. If you are using an existing MIT KDC installation, ensure that MIT KDC is running.
3. Ensure that forward and reverse DNS lookups are enabled on all the hosts:
   - All the compute hosts must have forward DNS lookup resolved correctly for all the hosts.
   - Isilon SmartConnect zonename lookups must resolve correctly.
   - Reverse PTR records for all IP addresses in the SmartConnect pool must exist.
   - Isilon OneFS must be able to resolve all the hosts, KDCs, and Active Directory servers as needed.
4. Test and validate all the host names and IP lookups before Kerberization:
   - Ambari must be able to manage and deploy `keytab` and `krb5.conf` files.
   - All the services must be running on the Ambari dashboard.
5. Do the following and restart all the services:
   a. Click **HDFS** > **Advanced** > **Custom core-site**. In the **Add Property** dialog box, create the key `hadoop.security.token.service.use_ip`, and set the value to `false`.
   b. Click **MapReduce2** > **Advanced** > **Advanced mapred-site** and add `hadoop classpath:` at the beginning of path in the **mapreduce.application.classpath** field.
6. Save and restart the service.

Enabling Ambari-automated Kerberos

Enable Ambari-automated Kerberos on an HDP cluster using MIT KDC as follows:

1. Log in to the Ambari web console and select **Admin** > **Kerberos**.
2. Click **Enable Kerberos**.

   The following warning message might appear, depending on your configuration settings:

   ```
   YARN log and local dir will be deleted and ResourceManager
   will be formatted as part of Enabling/Disabling Kerberos.
   ```

   Ignore this message and go to the next step.
3. On the **Get Started** page of the Enable Kerberos Wizard, as shown in Figure 7:
   a. Select **Existing MIT KDC**.
   b. In the **Existing MIT KDC** area of the page, select all the checkboxes to confirm that you have addressed all the prerequisites.

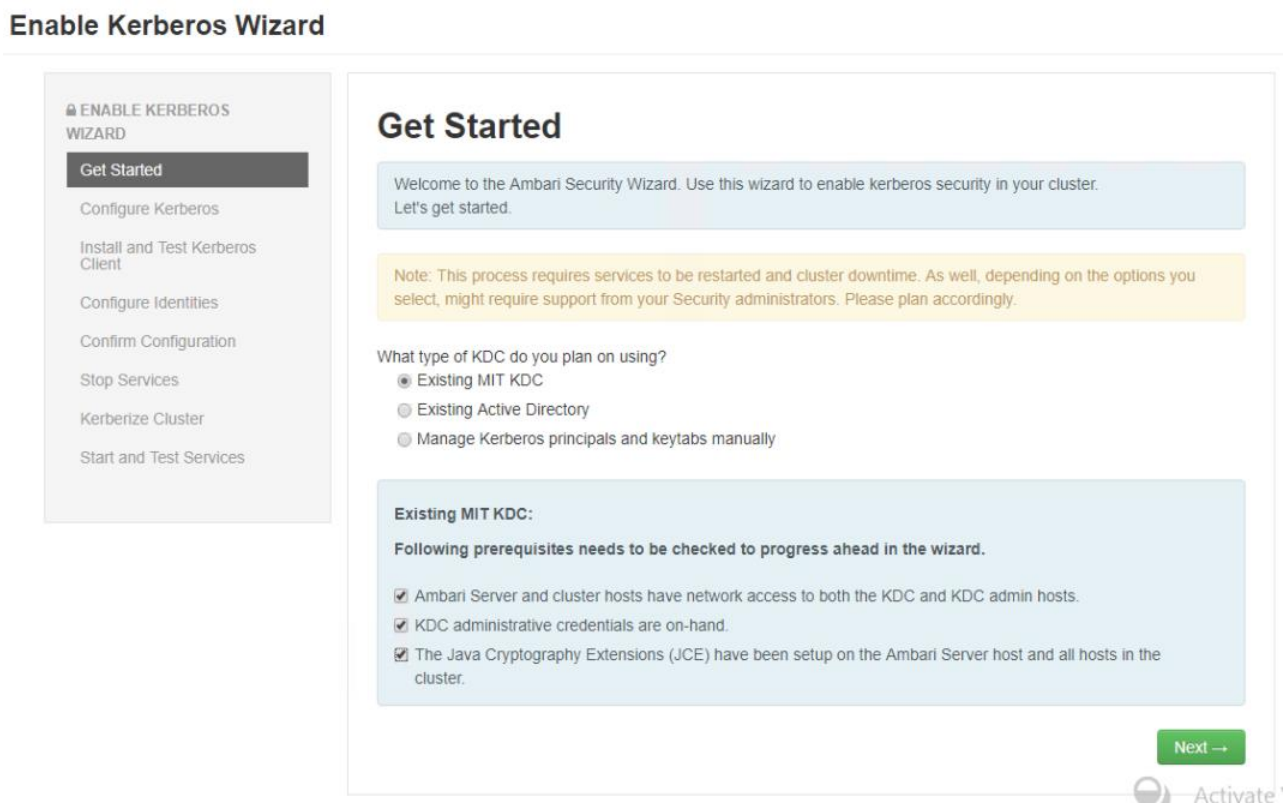


Figure 7. KDC settings on Get Started page of Enable Kerberos Wizard

For more information about setting up and meeting the requirements and prerequisites, see the *Ambari Security Guide* on the Hortonworks website.

4. On the **Configure Kerberos** page, specify the KDC and admin server information, as shown in the example in Figure 8.

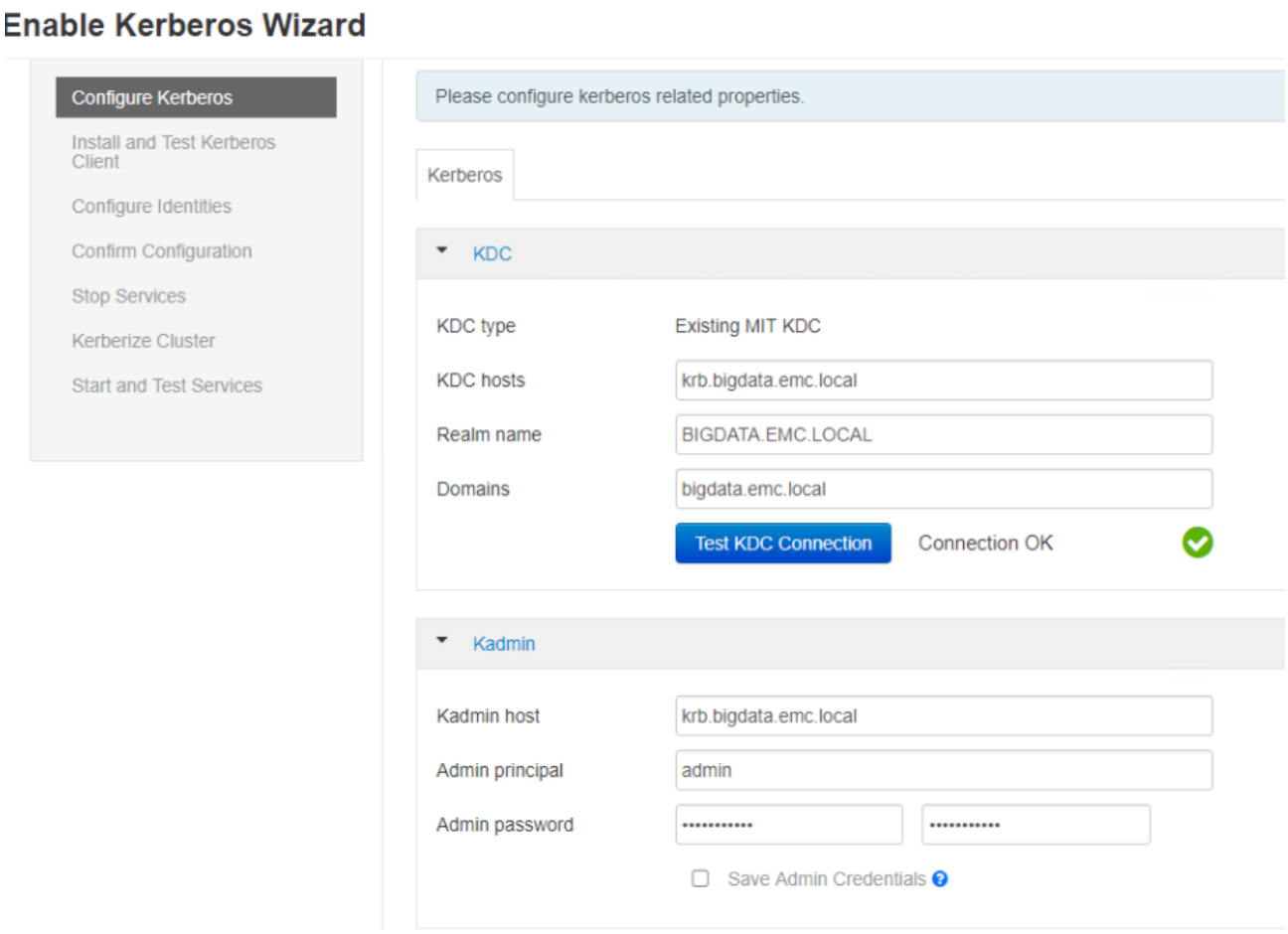


Figure 8. Configure Kerberos page of Enable Kerberos Wizard

5. On the **Install and Test Kerberos Client** page, run a test that deploys and configures Kerberos clients on all the hosts.

   Ambari Server performs a smoke test to ensure that you have configured Kerberos correctly. The wizard reports the status, as shown in Figure 9.

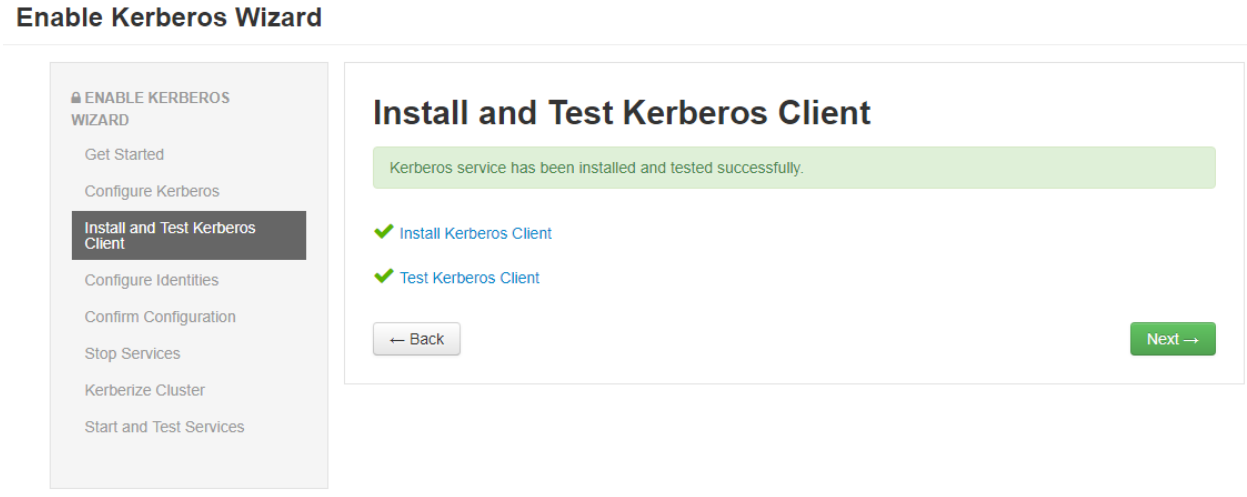


Figure 9. Results of test of Kerberos client configuration

6. On the **Configure Identities** page, specify mapping rules that are specific to configuring Kerberos for an Isilon OneFS cluster:

   a. Click the **General** tab and configure the Apache Ambari user principals as shown in Figure 10.

      Remove `-${cluster-name}` from the default value for all the **Ambari Principals**.

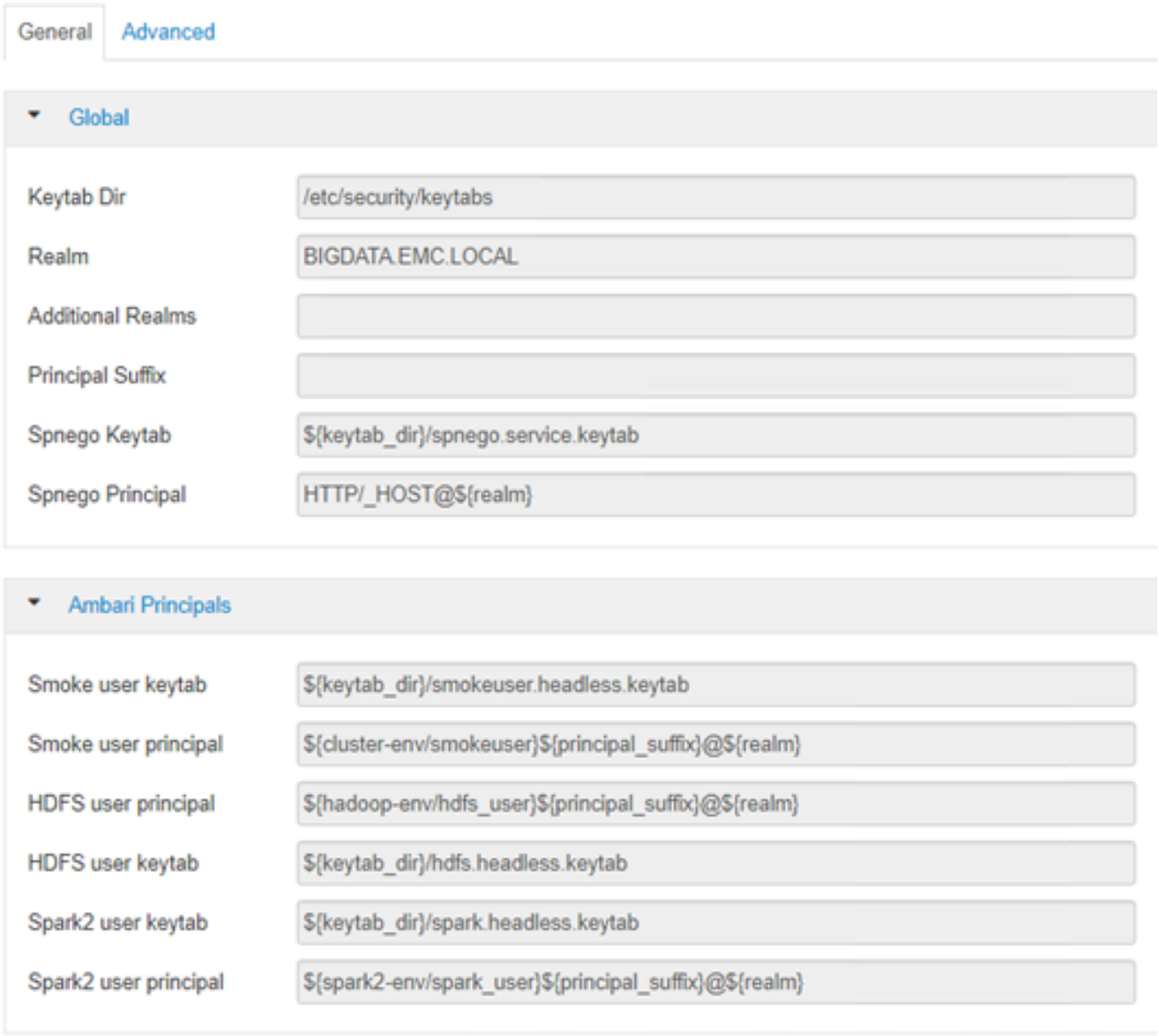


Figure 10. General tab of Configure Identities page

b. On the **Advanced** tab, update the properties as specified in Table 4.

Table 4. Ambari configuration properties and required values

Hadoop service component	Ambari configuration property name	Default value	Required value
HDFS	dfs.secondary.namenode. kerberos.principal	nn/_HOST@${realm}	hdfs/_HOST@$ {realm}
HDFS	dfs.secondary.namenode. keytab.file	${keytab_dir}/nn. service.keytab	${keytab_dir}/ hdfs.service. keytab
HDFS	dfs.datanode.kerberos. principal	dn/_HOST@${realm}	hdfs/_HOST@$ {realm}
HDFS	dfs.datanode.keytab.file	${keytab_dir}/dn. service.keytab	${keytab_dir}/ hdfs.service. keytab
HDFS	dfs.namenode.kerberos. principal	nn/_HOST@${realm}	hdfs/_HOST@$ {realm}
HDFS	dfs.namenode.keytab.file	${keytab_dir}/nn. service.keytab	${keytab_dir}/ hdfs.service. keytab
HDFS	dfs.secondary.namenode. kerberos.principal	nn/_HOST@${realm}	hdfs/_HOST@$ {realm}
YARN	yarn.nodemanager. principal	nm/_HOST@${realm}	yarn/_HOST@${realm}
YARN	yarn.nodemanager. keytab	${keytab_dir}/nm. service.keytab	${keytab_dir}/ yarn.service. keytab
YARN	yarn.resourcemanager. principal	rm/_HOST@${realm}	yarn/_HOST@$ {realm}
YARN	yarn.resourcemanager. keytab	${keytab_dir}/rm. service.keytab	${keytab_dir}/ yarn.service. keytab
MapReduce2	mapreduce.jobhistory. principal	jhs/_HOST@$ {realm}	mapred/_HOST@$ {realm}
MapReduce2	mapreduce.jobhistory. keytab	${keytab_dir}/jhs. service.keytab	${keytab_dir}/ mapred.service. keytab

7. Click **Next** on the **Configure Identities** page.
8. On the **Confirm Configuration** page, review the settings and click **Next** to accept them.
9. Wait until the **Stop Services** page indicates that all the servers are stopped, as shown in Figure 11, and then click **Next**.

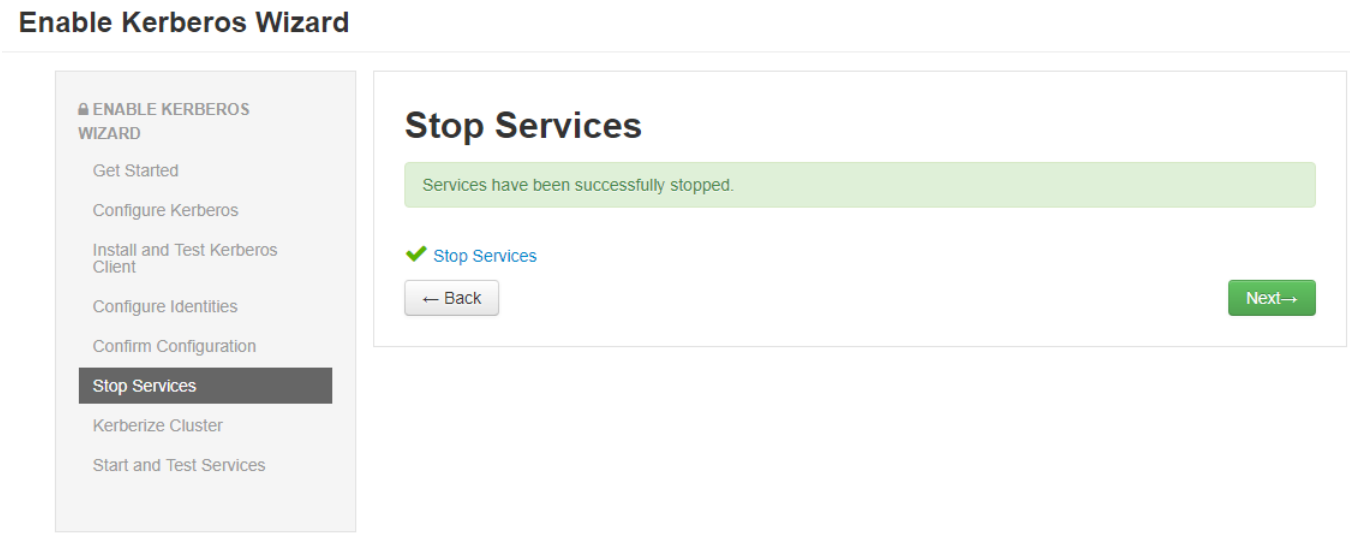


Figure 11. Message indicating that services have been stopped

The Kerberization process is automatically initialized. The Ambari services have been Kerberized, user principals have been created, and keytabs have been distributed.

10. Wait until you receive a message that Kerberos has been enabled on the cluster, and then click **Next**.

Warning: Do *not* click **Next** on the **Kerberize Cluste**r page until you see the message that Kerberos has been successfully enabled on the cluster, as shown in Figure 12.

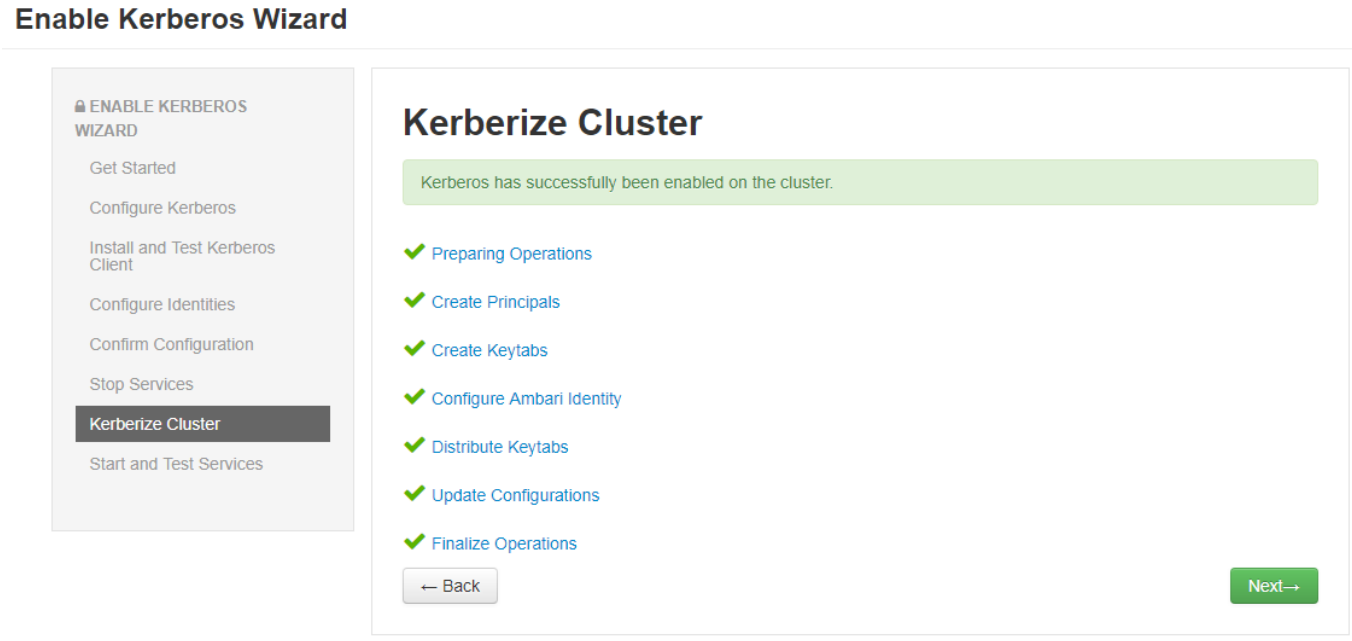


Figure 12. Message indicating that Kerberos has been enabled on the cluster

The **Start and Test Services** page displays the status of the testing, as shown in Figure 13.

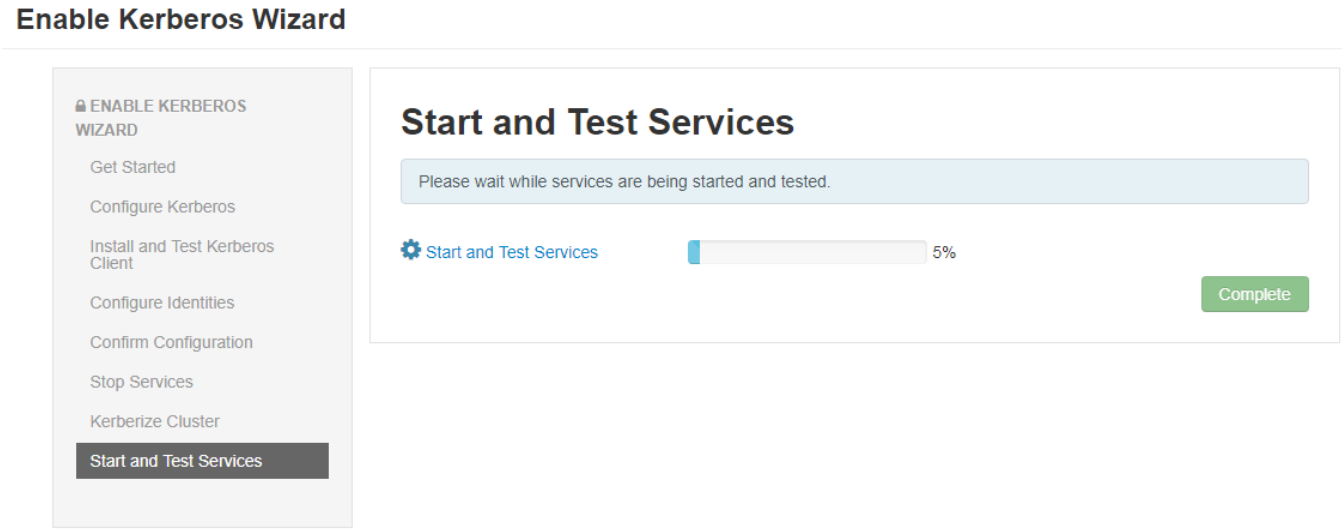


Figure 13. Start and Test Services page

11. When the status bar indicates that testing is finished, click **Complete**.

Enabling Kerberos on the Isilon cluster

Kerberize the Isilon cluster and synchronize it to the HDP cluster as follows:

1. Ensure that your access zone is configured to use MIT KDC. If it is not, follow these steps:
   a. Connect to an Isilon OneFS cluster and specify MIT KDC as an Isilon authentication provider.
   b. Configure your access zone to use MIT KDC by either using the OneFS web administration interface or by running the following commands through an SSH client:

```
isi auth krb5 create --realm=BIGDATA.EMC.LOCAL --admin-
server=krb.bigdata.emc.local --kdc=krb.bigdata.emc.local
--user=root/admin --password=Password01!

isi zone zones modify --zone=hdfs1 --add-auth-
provider=krb5:BIGDATA.EMC.LOCAL

isi zone zones modify --zone=hdfs2 --add-auth-
provider=krb5:BIGDATA.EMC.LOCAL
```

2. Create service principal names for HDFS and HTTP (for WebHDFS) by either using the OneFS web administration interface or by running the following commands through an SSH client:

```
isi auth krb5 spn create --provider-name=BIGDATA.EMC.LOCAL --spn=hdfs/isi-cluster-hdfs1.bigdata.emc.local@BIGDATA.EMC.LOCAL --user=root/admin --password=Password01!

isi auth krb5 spn create --provider-name=BIGDATA.EMC.LOCAL --spn=HTTP/isi-cluster-hdfs1.bigdata.emc.local@BIGDATA.EMC.LOCAL --user=root/admin --password=Password01!

isi auth krb5 spn create --provider-name=BIGDATA.EMC.LOCAL --spn=hdfs/isi-cluster-hdfs2.bigdata.emc.local@BIGDATA.EMC.LOCAL --user=root/admin --password=Password01!

isi auth krb5 spn create --provider-name=BIGDATA.EMC.LOCAL --spn=HTTP/isi-cluster-hdfs2.bigdata.emc.local@BIGDATA.EMC.LOCAL --user=root/admin --password=Password01!
```

3. In the Isilon OneFS web administration interface, under **Data Protection** > **Authentication**, enable Kerberos and provide the required information, as shown in Figure 14 and Figure 15.

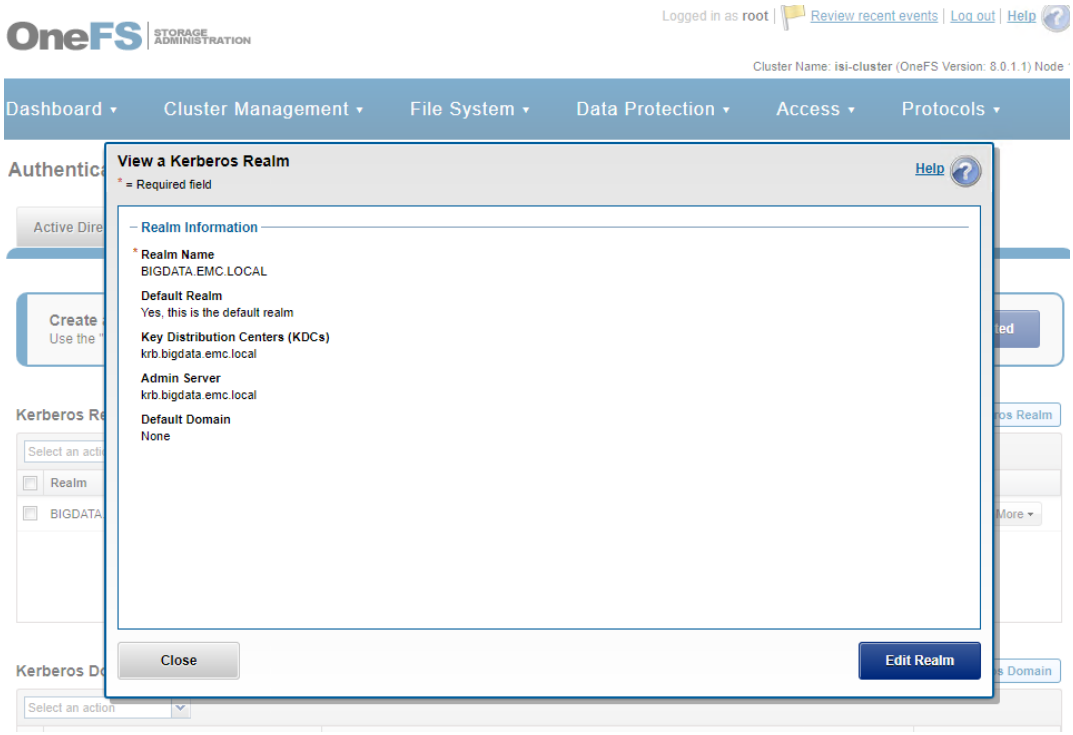


Figure 14. Enabling Kerberos authentication in the OneFS web administration interface

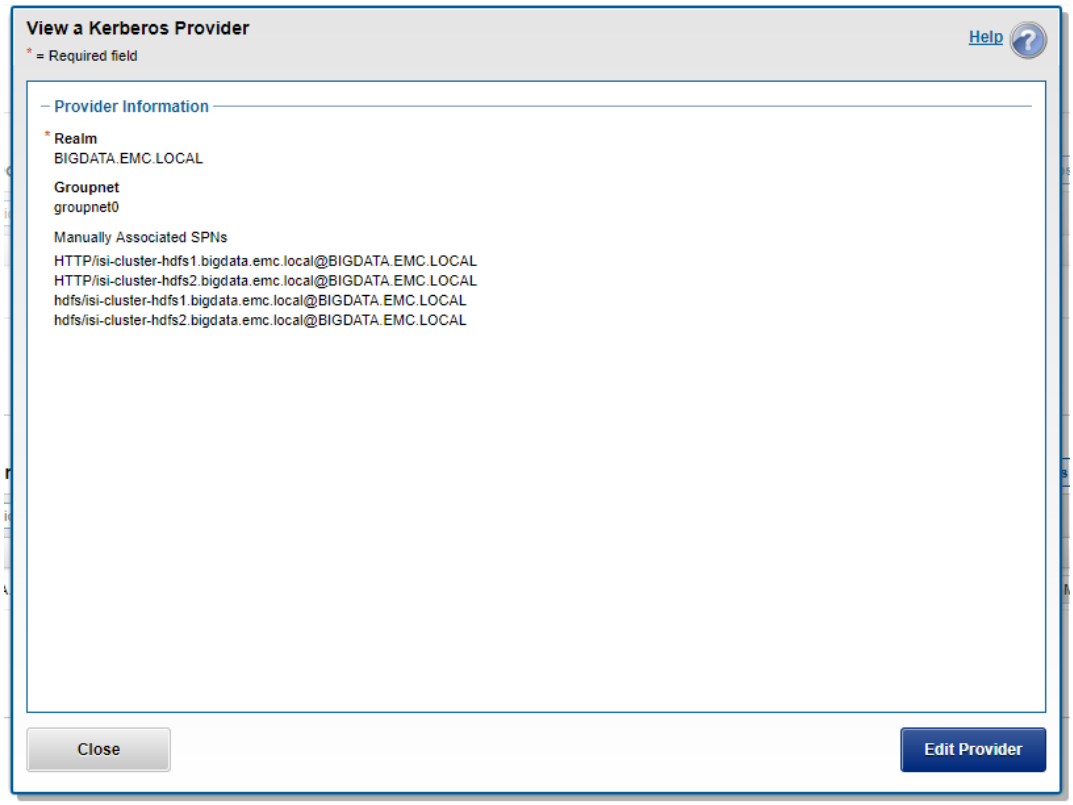


Figure 15. List of Kerberos providers in the OneFS web administration interface

4. Disable simple authentication by either using the OneFS web administration interface or by running the following command through an SSH client:

   ```
   isi hdfs settings modify --zone=$isilon_zone --
   authentication-mode=kerberos_only
   ```

   This action also ensures that WebHDFS uses only Kerberos for authentication.

5. Follow the steps in Enabling Ambari-automated Kerberos on page 29.

Enabling Ranger and setting policies

This section describes how to install Ranger services on the HDP and Isilon clusters and how to set up access policies.

Enabling access policies on HDP and Isilon clusters

1. In the Ambari Server web UI, select **Actions** > **+ Add Service**.
2. Add the Ranger service, as shown in Figure 16, and set up the Ranger admin and database host.

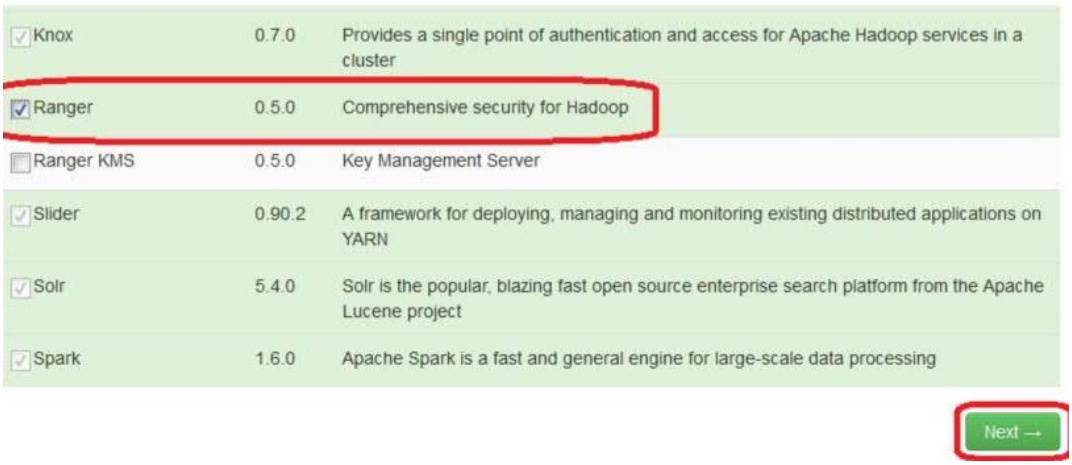


Figure 16. Selecting Ranger service

3. Start the Ranger service and verify the all the Ranger services are running as expected, as shown in Figure 17.

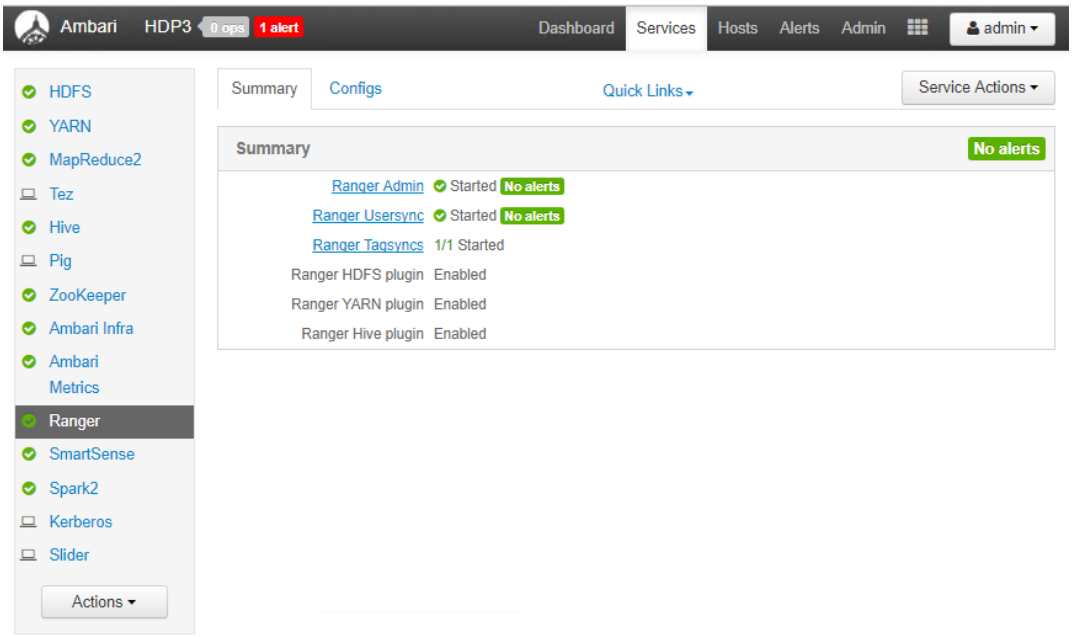


Figure 17. Ranger service in Ambari

4. Ensure that the Ranger admin and database hosts are provided, as shown in Figure 18.

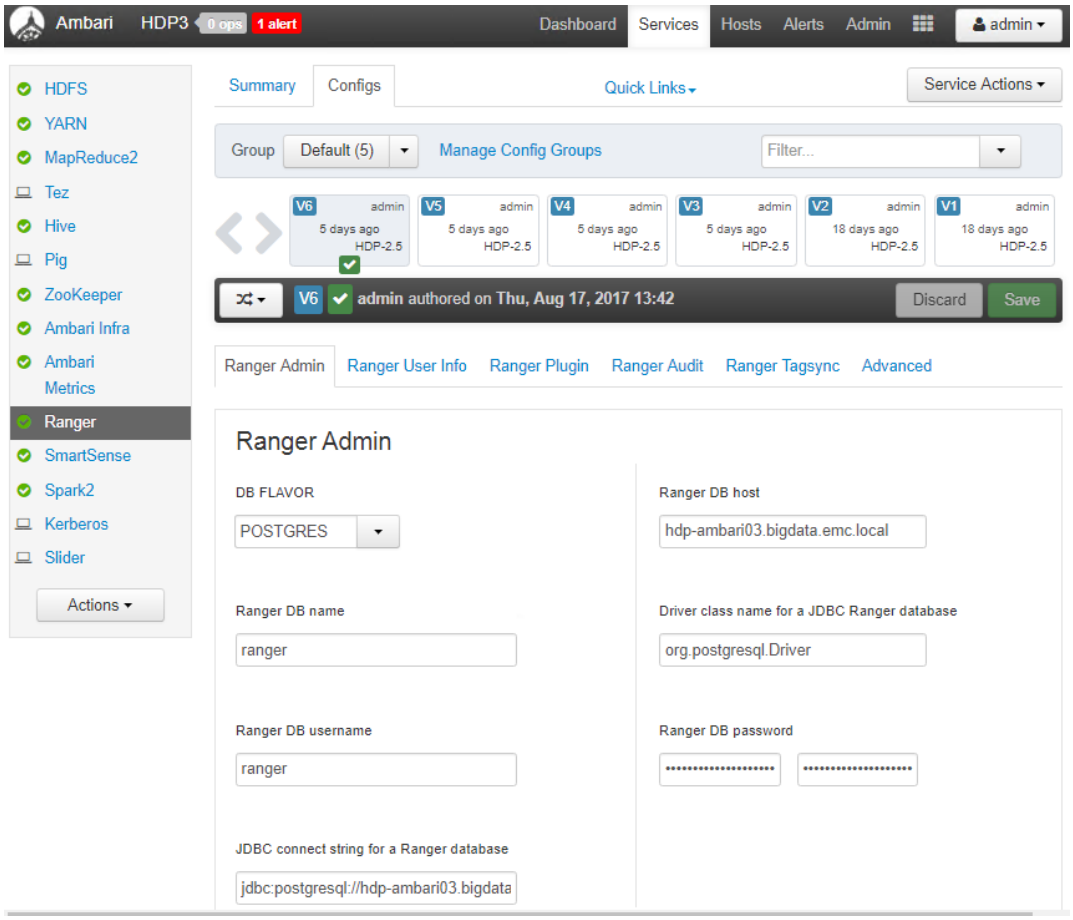


Figure 18. Ranger configuration

5. Under **Ranger Plugin**, enable the HDFS, YARN, and Hive Ranger plug-ins, as shown in Figure 19.

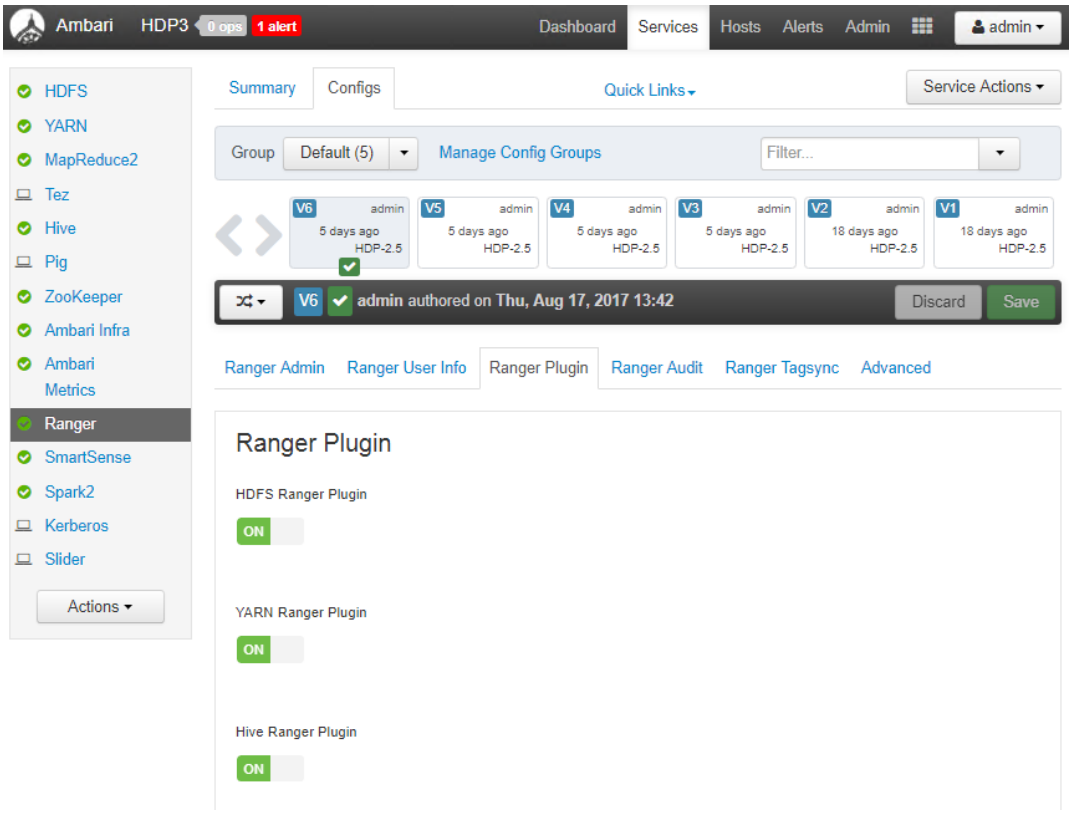


Figure 19. Enabling Ranger plug-ins

6. Log in to the Ranger admin panel from the web UI, and check the Service Manager to ensure that HDFS, Yarn, and Hive policies are enabled, as shown in Figure 20.

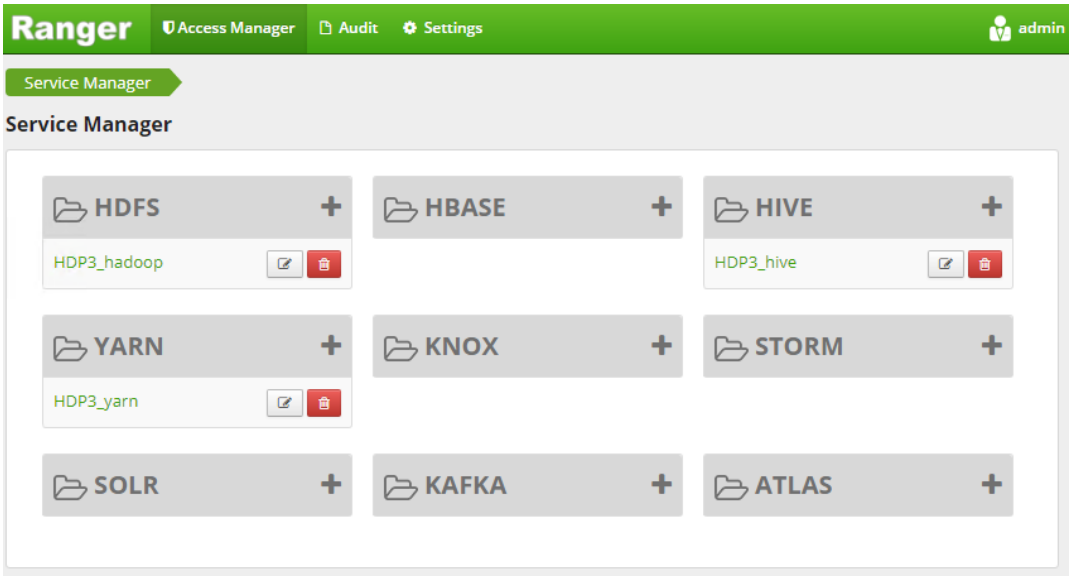


Figure 20. Service Manager on Ranger admin panel

7. Log in to the Isilon OneFS web UI.

8. On the **Ranger Plugin Settings** tab, as shown in Figure 21:
   a. Select **Enable Ranger Plugin**.
   b. In the **Policy manager URL** text box, type the URL for the policy manager.
   c. In the **Repository name** text box, type the repository name.

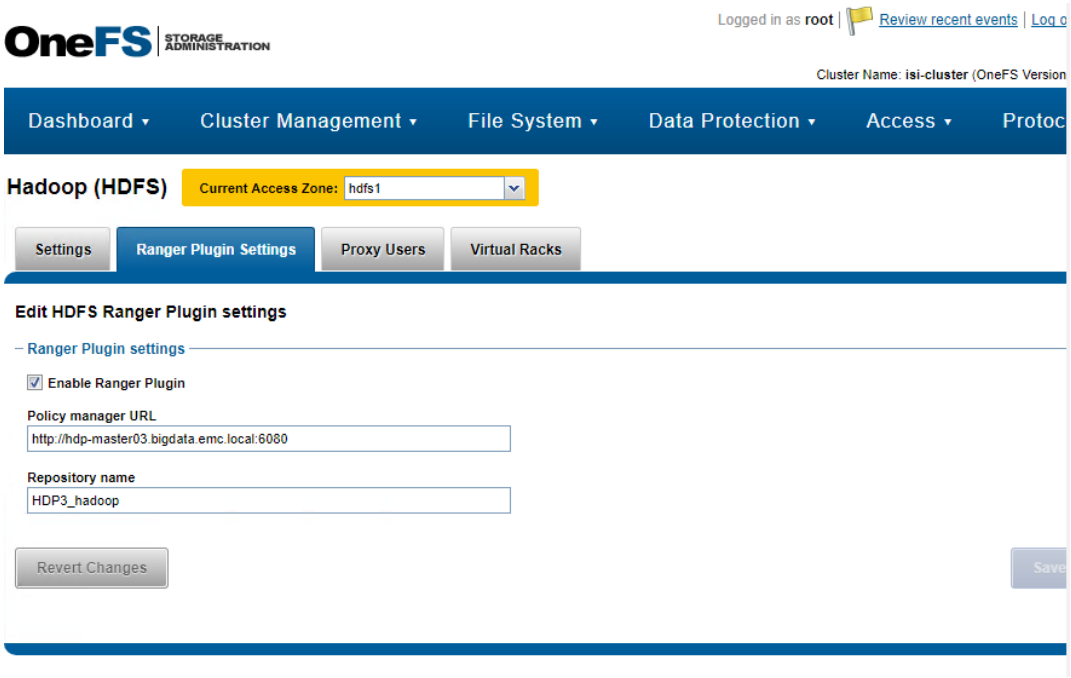


Figure 21. Ranger Plugin Settings tab

Creating and assigning new access policies

1. Create sample directories such as `GRANT_ACCESS` and `RESTRICT_ACCESS` on the Isilon HDFS cluster.
2. Create hdp-user1 on all the nodes of the HDP cluster and Isilon cluster.
3. In the Ranger UI under HDP3_hadoop Service Manager, assign Read/Write/Execute (RWX) access for the hdp-user1 on `GRANT_ACCESS`, as shown in Figure 22 and Figure 23.

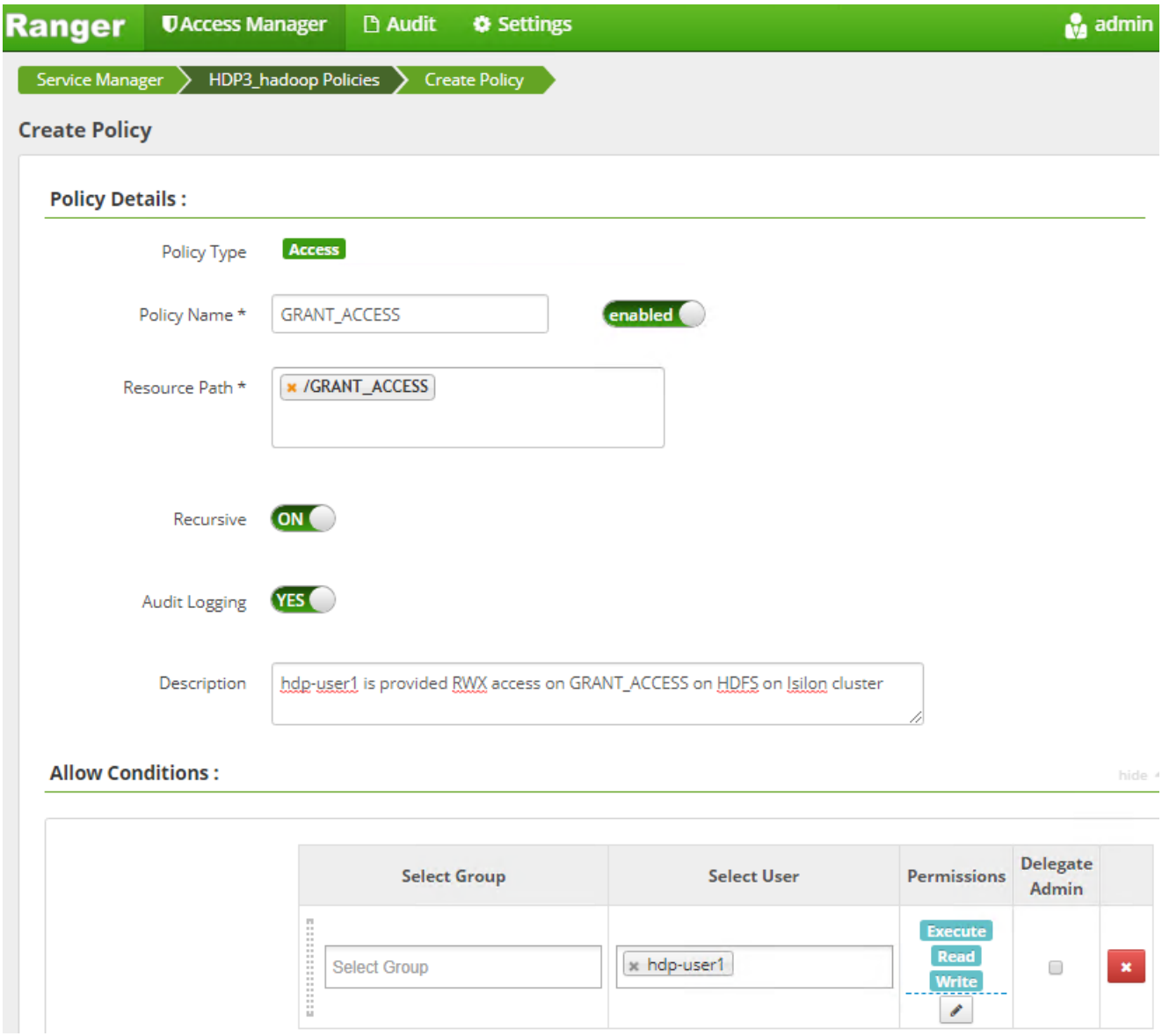


Figure 22. Creating Ranger policy GRANT_ACCESS and providing access

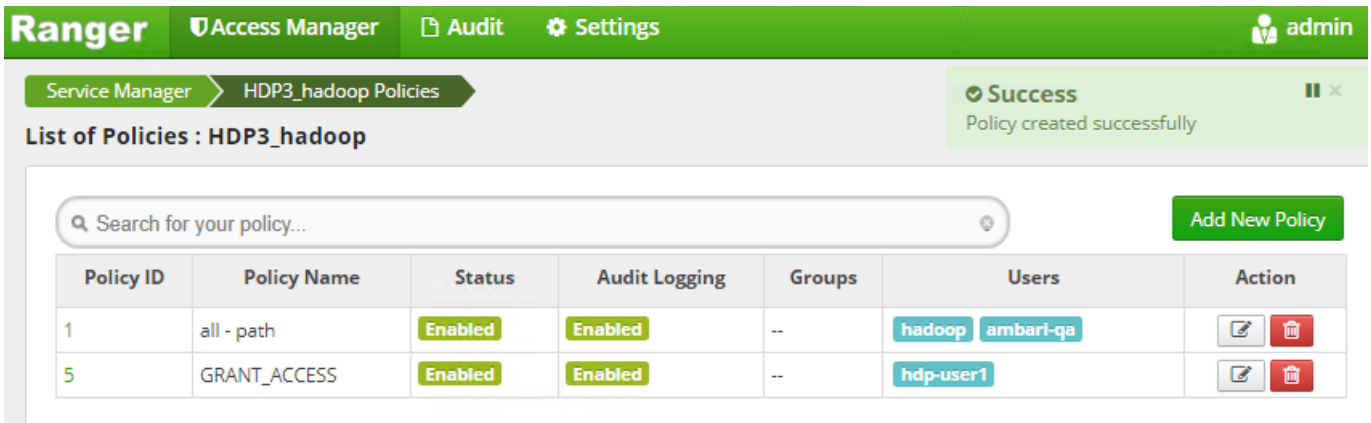


Figure 23. New Ranger policy listed in the HDFS policies

4. Similarly, create a `RESTRICT_ACCESS` policy for the hdp-user1, and add the user under **Deny Conditions**, as shown in Figure 24.

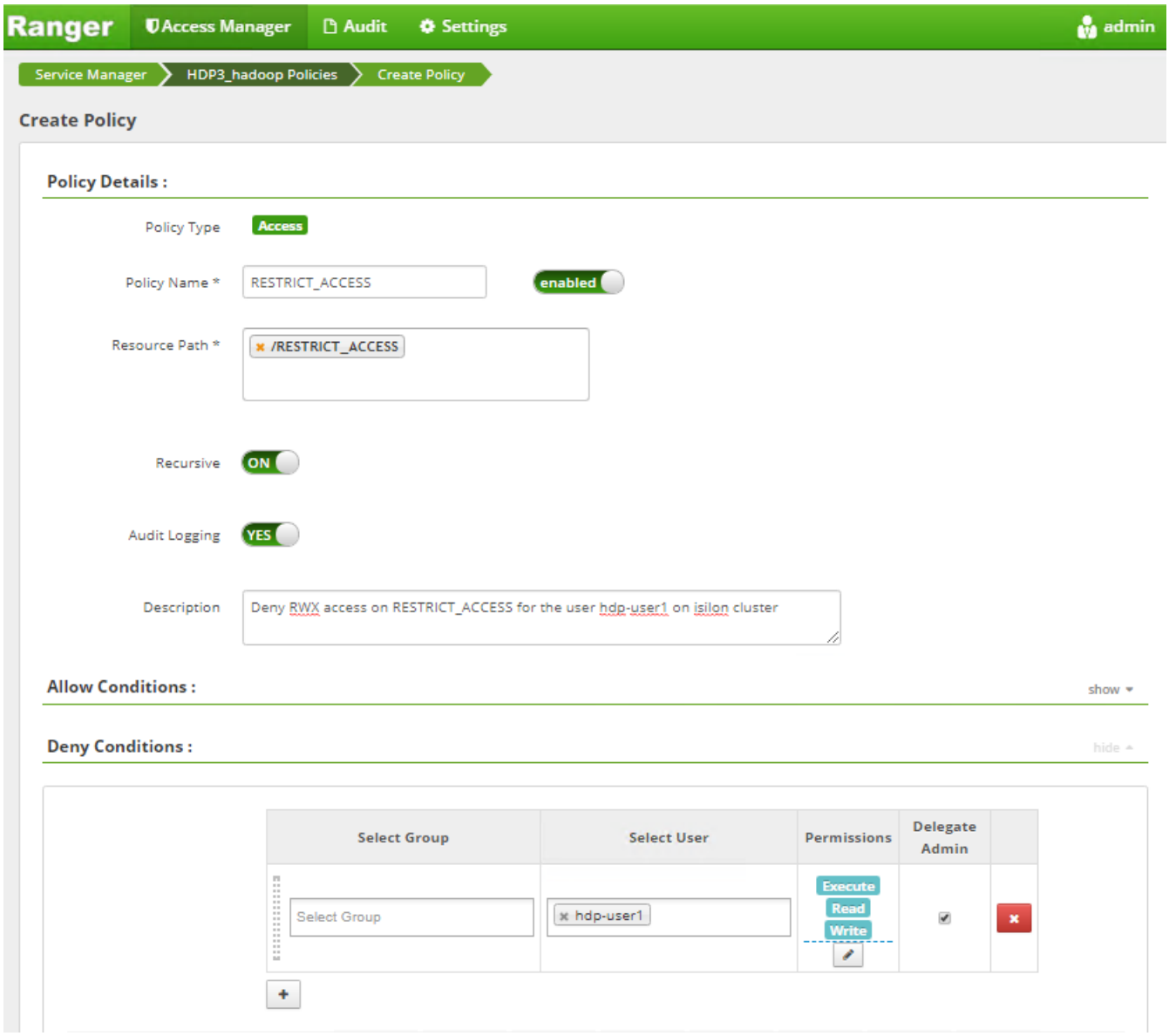


Figure 24. Restrict access policy for selected user

Setting up Ranger policy for Hive data warehouse

1. In the Ranger UI, create a Hive data warehouse policy and assign RWX permissions on the `/user/hive` directory for hdp-user1 on the HDP and Isilon clusters, as shown in the example in Figure 25.

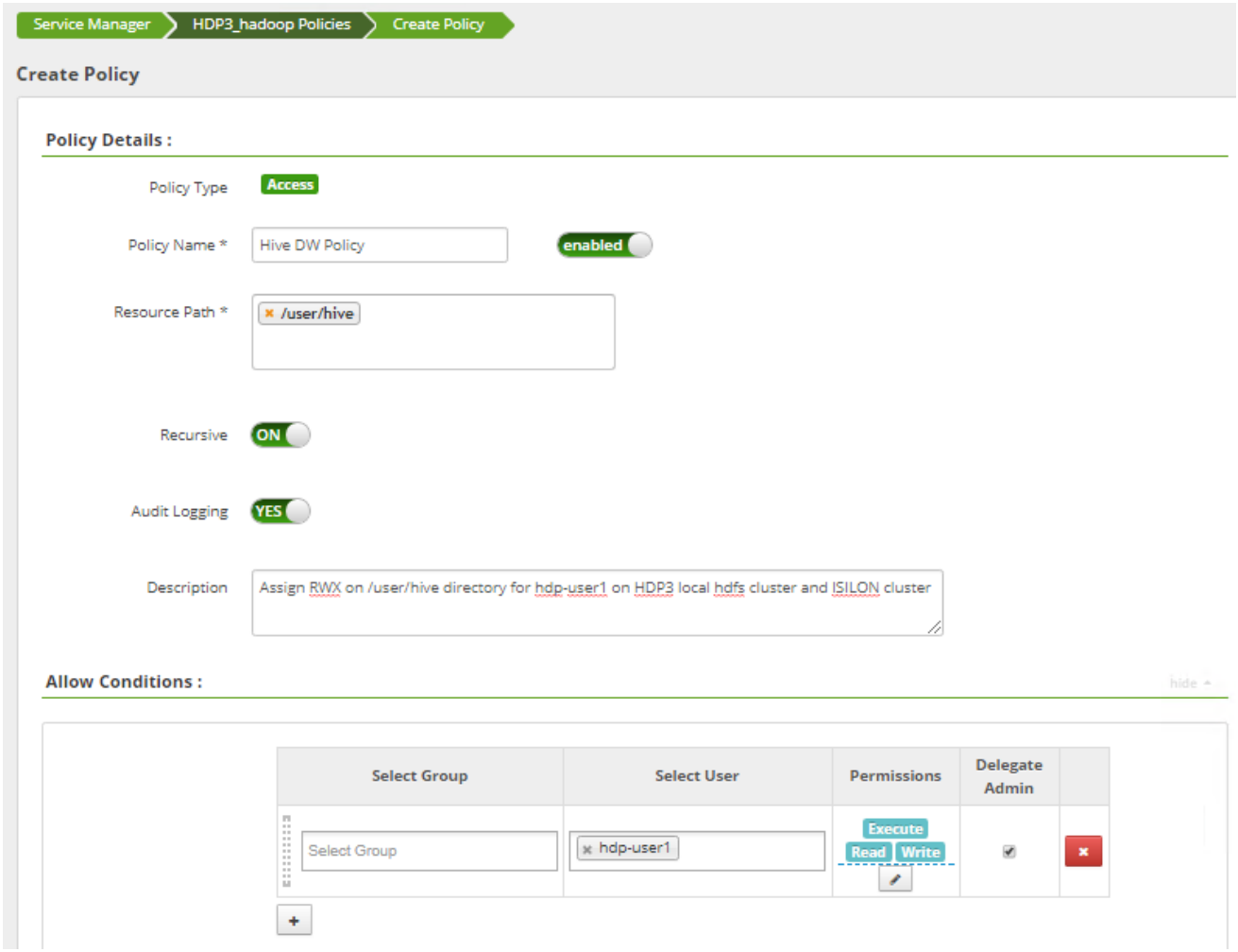


Figure 25. Creating Hive data warehouse policy and assigning permissions

Figure 26 shows the assigned policies.

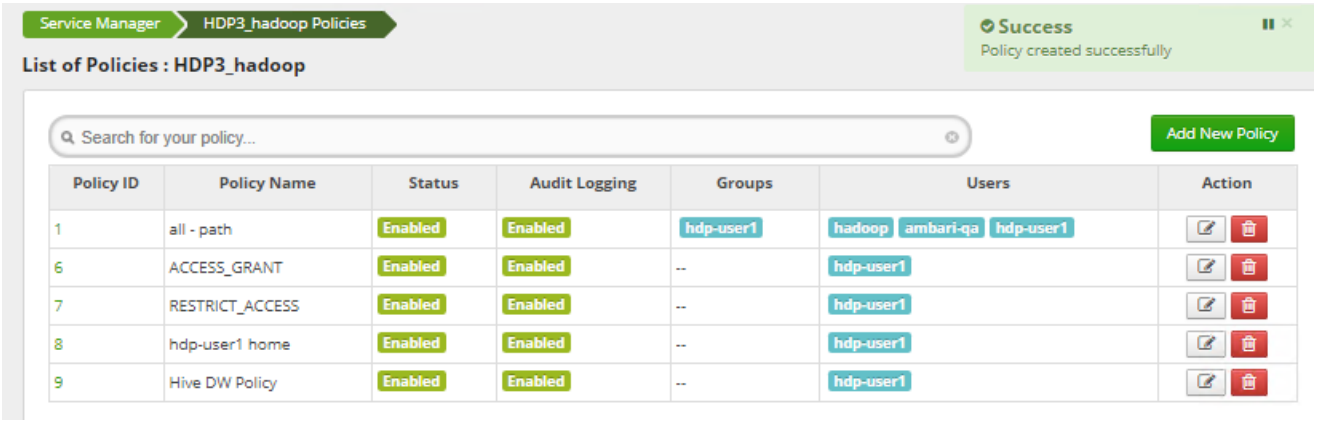


Figure 26. Policy list in Ranger UI

2. In the OneFS web UI, enable the Ranger plug-in and specify the policy manager URL and repository name, as shown in Figure 27.

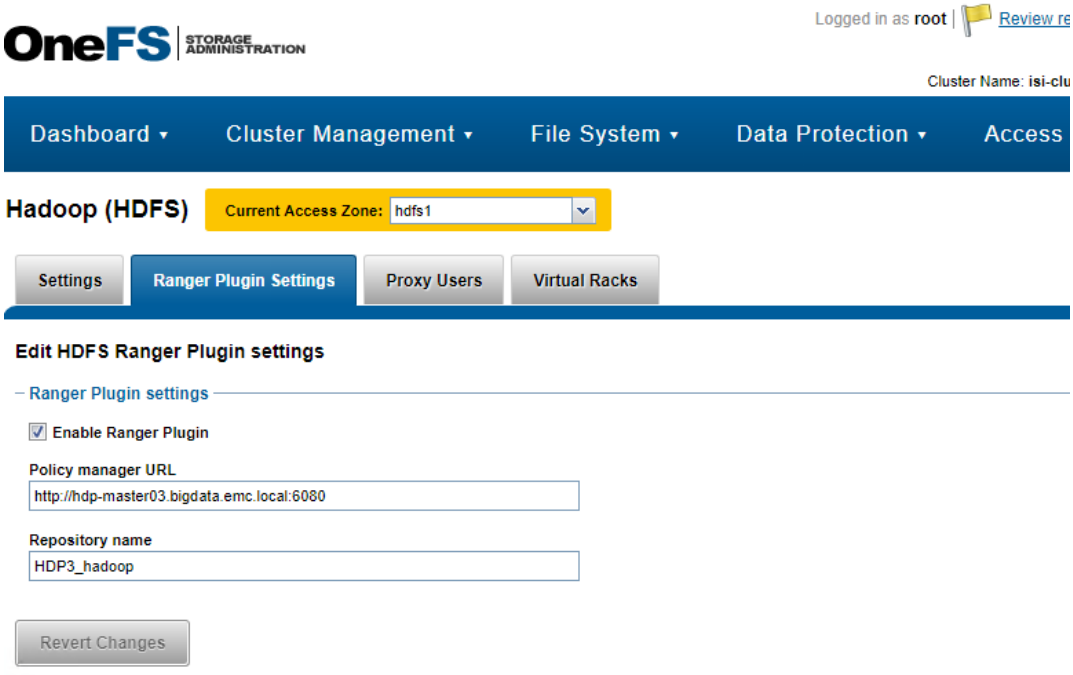


Figure 27. OneFS Ranger Plugin Settings tab

3. In the OneFS web UI, set up hdp-user1 as a proxy user for Hive in the Isilon cluster, as shown in Figure 28.

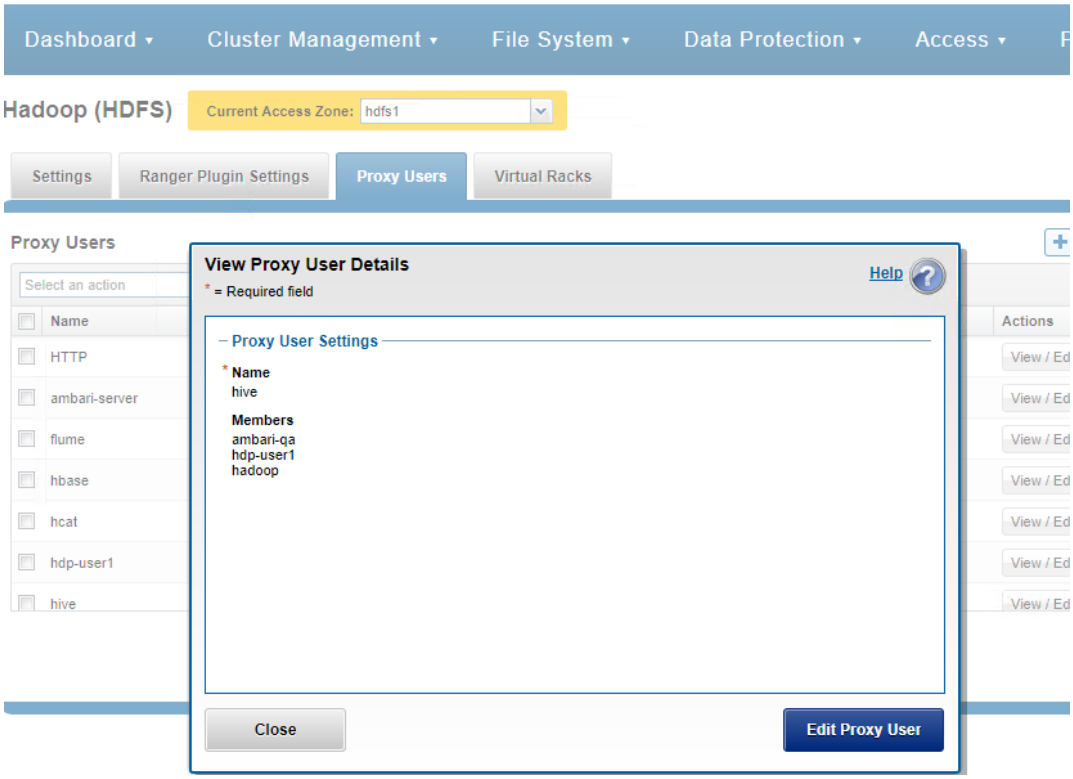


Figure 28. Editing proxy user in OneFS web UI

Validating HDP deployment and Isilon integration

Overview

Table 5 lists the validation procedures for the deployment of Hadoop tiered storage and integration with an Isilon cluster.

Table 5. Validating HDF cluster deployment and integration with Isilon cluster

Step	Category	Validation procedures
1	Ambari	Install/configure validation and GUI functionality testing
2	Kerberos and Ranger environment configuration	Kerberos and Ranger functionality testing
3	Kerberos security	Kerberos user and non-Kerberos user testing on Kerberized Hadoop and Isilon clusters
4	Ranger policy	GRANT_ACCESS and RESTRICT_ACCESS functionality test
5	Ranger policy with Kerberos security	• MapReduce—WordCount job run for all permutations of default and nondefault file systems as input and output directories • Spark—WordCount and LineCount job run for all permutations of default and nondefault file systems as input and output directories
6	Ranger policy with Kerberos security on Hive data warehouse	1. Data Definition Language (DDL) operations, LOAD data local inpath, INSERT into table, INSERT Overwrite table 2. Data Manipulation Language (DML) operations 3. JOIN tables in and between local DAS HDFS and remote Isilon tier HDFS 4. Temp table, import and export operations 5. Table-level and column-level statistics
7	DistCp in Kerberized and non-Kerberized clusters	DistCp operation testing in and between local DAS HDFS and remote Isilon tier HDFS

Validation procedures

Ambari

The steps for performing an Ambari smoke test are as follows.

Note: For screenshots of our Ambari smoke test, see Appendix A on page 81. For a list of specific test steps, see Appendix B on page 87.

4. Set up a five-node HDP cluster for local DAS HDFS and a three-node Isilon cluster as the remote HDFS cluster.
5. In the Ambari Server web UI, run all the service checks, and stop and restart all the services.

Kerberos and Ranger environment configuration

Table 6 outlines the validation procedures for the Kerberos and Ranger environments.

Table 6. Validation of Kerberos and Ranger environment configuration

Test case name	Step	Description
Kerberos setup	1	Set up the Kerberos KDC server
	2	Create the admin principal
	3	Add the Ambari host to the HDP cluster to install Ambari agent
	4	Kerberize the HDP (local DAS HDFS) and Isilon cluster
	5	Verify that keytabs were generated for all the HDP service accounts
Ranger setup	1	Create a database and user, and assign a role for Ranger in PostgreSQL
	2	Add the new Ranger service into the cluster
	3	Add the Ranger plug-in for HDFS, YARN, and Hive

Kerberos security

We validated Kerberos security as follows.

Note: For a list of specific test steps, see Appendix B on page 87.

1. Kerberize the local DAS HDFS and remote Isilon HDFS clusters and create hdp-user1 on all the nodes of the local DAS HDFS cluster.

   Do not add the hdp-user1 principal to the Kerberos KDC server and try to access the local DAS HDFS cluster. Permission will be denied.

```
[hdp-user2@hdp-master03 ~]$ clear
[hdp-user2@hdp-master03 ~]$ whoami
hdp-user2
[hdp-user2@hdp-master03 ~]$ hadoop fs -ls /
17/08/23 18:17:09 WARN ipc.Client: Exception encountered while connecting to the server :
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided (Mecha
nism level: Failed to find any Kerberos tgt)]
        at com.sun.security.sasl.gsskerb.GssKrb5Client.evaluateChallenge(GssKrb5Client.java:211)
        at org.apache.hadoop.security.SaslRpcClient.saslConnect(SaslRpcClient.java:413)
        at org.apache.hadoop.ipc.Client$Connection.setupSaslConnection(Client.java:595)
        at sun.security.jgss.GSSContextImpl.initSecContext(GSSContextImpl.java:212)
        at sun.security.jgss.GSSContextImpl.initSecContext(GSSContextImpl.java:179)
        at com.sun.security.sasl.gsskerb.GssKrb5Client.evaluateChallenge(GssKrb5Client.java:192)
        ... 41 more
ls: Failed on local exception: java.io.IOException: javax.security.sasl.SaslException: GSS initiate failed [Caused b
y GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]; Host Details : lo
cal host is: "hdp-master03.bigdata.emc.local/172.16.1.62"; destination host is: "hdp-master03.bigdata.emc.local":802
0;
[hdp-user2@hdp-master03 ~]$
```

2. Add the user hdp-user1 principal to the Kerberos KDC server, assign a password, and access the local DAS HDFS cluster:

```
sudo -u hdp-user1 hdfs dfs -ls hdfs://hdp-
master03.bigdata.emc.local:8020/user/hdp-user1/mr/
Found 1 items
-rw-r--r--   3 hdp-user1 hdfs         52 2017-08-24 13:13
hdfs://hdp-master03.bigdata.emc.local:8020/user/hdp-
user1/mr/redhat-release
```

3. Access the remote Isilon cluster as the hdp-user1:

```
sudo -u hdp-user1 hdfs dfs -ls hdfs://isi-cluster-
hdfs1.bigdata.emc.local:8020/user/hdp-user1/mr/
Found 1 items
-rw-r--r--   3 hdp-user1 hdpuser         52 2017-08-24 13:13
hdfs://isi-cluster-hdfs1.bigdata.emc.local:8020/user/hdp-
user1/mr/redhat-release
```

Ranger policies

The steps for validating Ranger policies are as follows.

Note: For a list of specific test steps, see Appendix B on page 87.

1. Assign RWX access to hdp-user1 in the `GRANT_ACCESS` directory and deny RWX access in the `RESTRICT_ACCESS` directory.
2. Access the `GRANT_ACCESS` directory:

```
hadoop fs -ls hdfs://isi-cluster-
hdfs1.bigdata.emc.local:8020/GRANT_ACCESS
hadoop fs -put /etc/redhat-release hdfs://isi-cluster-
hdfs1.bigdata.emc.local:8020/GRANT_ACCESS/
hadoop fs -ls hdfs://isi-cluster-
hdfs1.bigdata.emc.local:8020/GRANT_ACCESS
Found 1 items
-rw-r--r--   3 hdp-user1 hadoop         52 2017-08-24 12:12
hdfs://isi-cluster-
hdfs1.bigdata.emc.local:8020/GRANT_ACCESS/redhat-release
```

3. Add the user hdp-user1 to the `RESTRICT_ACCESS` directory on the remote Isilon HDFS:

```
hadoop fs -put /etc/redhat-release hdfs://isi-cluster-
hdfs1.bigdata.emc.local:8020/RESTRICT_ACCESS/
17/08/24 12:18:34 WARN retry.RetryInvocationHandler:
Exception while invoking
ClientNamenodeProtocolTranslatorPB.getFileInfo over null.
Not retrying because try once and fail.
org.apache.hadoop.ipc.RemoteException(org.apache.ranger.auth
orization.hadoop.exceptions.RangerAccessControlException):
Permission denied: user=hdp-user1@BIGDATA.EMC.LOCAL,
access=EXECUTE, path="/RESTRICT_ACCESS"
        at
org.apache.hadoop.ipc.Client.getRpcResponse(Client.java:1552
)
        at
org.apache.hadoop.ipc.Client.call(Client.java:1496)
        at
org.apache.hadoop.ipc.Client.call(Client.java:1396)
```

```
                .
                .
                at
org.apache.hadoop.fs.FsShell.main(FsShell.java:350)
put: Permission denied: user=hdp-user1@BIGDATA.EMC.LOCAL,
access=EXECUTE, path="/RESTRICT_ACCESS"
```

Ranger policies with Kerberos security

Table 7 outlines the procedures for validating Ranger policies with Kerberos security.

Table 7. Validation of Ranger policies with Kerberos security

Test case name	Step	Description
MapReduce (word count)	1	Create an hdp-user1 home directory on the HDP3 (local HDFS) and Isilon clusters
	2	In the Ranger UI, assign RWX on the `/user/hdp-user1` directory for hdp-user1 on the HDP3 (local HDFS) and Isilon cluster
	3	Put local file `/etc/redhat-release` on the HDP3 (local HDFS) file system
	4	Put local file `/etc/redhat-release` on the Isilon HDFS
	5	Run MapReduce WordCount job on input from HDP3 (local HDFS) with output to the Isilon HDFS
	6	Run MapReduce WordCount job on input from Isilon HDFS1, with output to HDP3 (local HDFS)
Spark (line count and word count)	1	Put local file `/etc/passwd` on the HDP3 (local HDFS) file system
	2	Put local file `/etc/passwd` on the Isilon HDFS
	3	Run a Spark LineCount/WordCount job on input from the primary HDP3 (local HDFS), with output to the secondary Isilon HDFS
	4	Run a Spark LineCount/WordCount job on input from the secondary Isilon HDFS, with output to the primary HDP3 (local HDFS)

Ranger policies with Kerberos security on Hive data warehouse

The steps for validating Ranger policies with Kerberos security on the Hive data warehouse are as follows.

Note: For a list of specific test steps, see Appendix B on page 87.

1. In the Ranger UI, assign RWX on the `/user/hive` directory for hdp-user1 on HDP (local DAS HDFS) and Isilon clusters.
2. Ensure that the local DAS HDFS and remote Isilon HDFS clusters are Kerberized and the necessary Ranger policies for user hdp-user1 RWX access are provided.

3. Switch to hdp-user1 and run the Hive CLI to create a remote database location on the remote Isilon HDFS cluster:

```
CREATE database remote_DB COMMENT 'Holds all the tables data
in remote location Hadoop cluster' LOCATION 'hdfs://isi-
cluster-hdfs1.bigdata.emc.local:8020/user/hive/remote_DB'
OK
Time taken: 0.045 seconds
```

4. Create an internal nonpartitioned table and load data using local inpath:

```
USE remote_DB
OK
Time taken: 0.036 seconds
```

```
CREATE TABLE passwd_int_nonpart (user_name STRING, password
STRING, user_id STRING, group_id STRING, user_id_info
STRING, home_dir STRING, shell STRING) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ':'
OK
Time taken: 0.211 seconds
```

```
LOAD data local inpath '/etc/passwd'  into TABLE
passwd_int_nonpart
Loading data to table remote_db.passwd_int_nonpart
Table remote_db.passwd_int_nonpart stats: [numFiles=1,
numRows=0, totalSize=1808, rawDataSize=0]
OK
Time taken: 0.261 seconds
```

DistCp in Kerberized and non-Kerberized clusters

The steps for validating DistCp in Kerberized and non-Kerberized clusters are as follows:

1. Run DistCp to copy a sample file from the local DAS HDFS to the remote Isilon HDFS in a non-Kerberized cluster:

```
sudo -u hdfs hadoop distcp -skipcrccheck -update
/tmp/redhat-release hdfs://isilon.solarch.lab.emc.com/tmp/
sudo -u hdfs hdfs dfs -ls -R
hdfs://isilon.solarch.lab.emc.com:8020/tmp/redhat-release
-rw-r--r--   3 root hdfs         27 2017-09-07 13:26
hdfs://isilon.solarch.lab.emc.com:8020/tmp/redhat-release
```

2. Run DistCp to copy a sample file from the remote isilon HDFS to the local DAS HDFS in a Kerberized cluster:

```
sudo -u hdfs hadoop distcp -pc
hdfs://isilon.solarch.lab.emc.com:8020/tmp/redhat-release
hdfs://hdp-master.bigdata.emc.local:8020/tmp/
sudo -u hdfs hdfs dfs -ls -R hdfs://hdp-
master.bigdata.emc.local:8020/tmp/redhat-release
-rw-r--r--   3 hdfs hdfs         27 2017-09-07 13:26
hdfs://hdp-master.bigdata.emc.local:8020/tmp/redhat-release
```

Validation results

This section describes detailed results of the validation procedures as PASS/FAIL.

Table 8. Test cases, expected results, and status

Category	Test case name	Expected test results	Status
Ambari	Validation of install/configuration	Deployment is completed successfully	PASS
	Usability and functionality test of GUI	Service check runs successfully	PASS
MapReduce	Word count	Command runs successfully	PASS
Spark	Line count and word count	Command runs successfully	PASS
Hive/Tez and Hive/MapReduce	DDL operations 1. LOAD data local inpath 2. INSERT into table 3. INSERT Overwrite TABLE	Command runs successfully	PASS
	DML operations 1. Query local database tables 2. Query remote database tables	Command runs successfully	PASS
	JOIN tables in local database	Command runs successfully	PASS
	JOIN tables in remote database	Command runs successfully	PASS
	JOIN tables between local_db and remote_db	Command runs successfully	PASS
	Local temporary table from remote_db table	Command runs successfully	PASS
	IMPORT and EXPORT operations	Command runs successfully	PASS
	Table-level and column-level statistics	Command runs successfully	PASS
Hive TPC-DS	Hive-testbench preparation	Command runs successfully	PASS
	Database creation on remote Hadoop cluster, and data load	Command runs successfully	PASS
	TPC-DS benchmarking	Command runs successfully	PASS
Kerberos and Ranger environment configuration	Kerberos setup	Command runs successfully	PASS
	Ranger setup	Command runs successfully	PASS
Kerberos users	Kerberos user setup and testing	Command runs successfully	PASS
	Non-Kerberos user setup and testing	Command runs successfully	PASS
Ranger policy	User access and restriction policy	Command runs successfully	PASS
Ranger policy and Kerberos security	MapReduce > WordCount	Command runs successfully	PASS
	Spark > LineCount & WordCount	Command runs successfully	PASS

Category	Test case name	Expected test results	Status
Ranger policy with Kerberos security on Hive warehouse	Ranger policy setup for Hive data warehouse	Command runs successfully	PASS
	DDL operations 1. LOAD data local inpath 2. INSERT into table 3. INSERT Overwrite TABLE	Command runs successfully	PASS
	DML operations 1. Query local database tables 2. Query remote database tables	Command runs successfully	PASS
	JOIN tables in local database	Command runs successfully	PASS
	JOIN tables in remote database	Command runs successfully	PASS
	JOIN tables between local_db and remote_db	Command runs successfully	PASS
	Local temporary table from remote_db table	Command runs successfully	PASS
	IMPORT and EXPORT operations	Command runs successfully	PASS
	Table-level and column-level statistics	Command runs successfully	PASS
DistCp in Kerberized and non-Kerberized cluster	DisctCp and script	Command runs successfully	PASS

Chapter 5 Hadoop Cluster Deployment and Integration with ECS Cluster

This chapter presents the following topics:

Overview

Table 9 lists the process flow for the Hadoop cluster deployment with an ECS cluster.

Table 9. HDP and ECS cluster deployment and integration

Step	Action
1	Set up the HDP cluster
2	Set up the ECS cluster
3	Create ECS buckets
4	Install ECS HDFS Client software
5	Enable Kerberos on the HDP cluster
6	Enable Kerberos on the ECS cluster
7	Validate HDP deployment and ECS integration

Setting up the HDP and ECS clusters

To set up the HDP cluster, see Setting up the HDP cluster on page 20.

After you set up the HDP cluster, contact your Dell EMC or partner representative to set up the four-node ECS appliance cluster infrastructure.

Creating ECS buckets

After you set up the HDP and ECS clusters, create buckets on the ECS cluster and prepare them for use, as follows.

For more details, see the instructions for creating a bucket for the HDFS filesystem in the *Elastic Cloud Storage Data Access Guide*.

1. Log in to the ECS web console and go to **Manage** > **Namespace** > **New Namespace**.
2. Create namespace ns01, as shown in Figure 29.

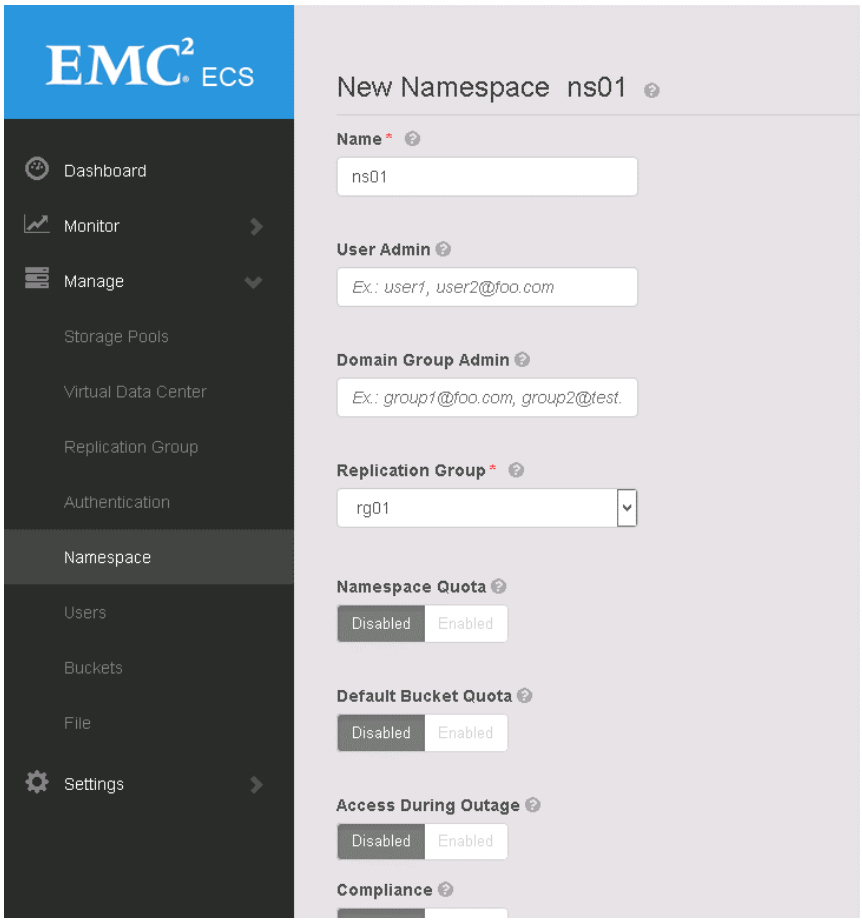


Figure 29. Creating ECS namespace ns01

3. Go to **Manage** > **Users** > **New Object User**.
4. Create user hdfs, as shown in Figure 30.

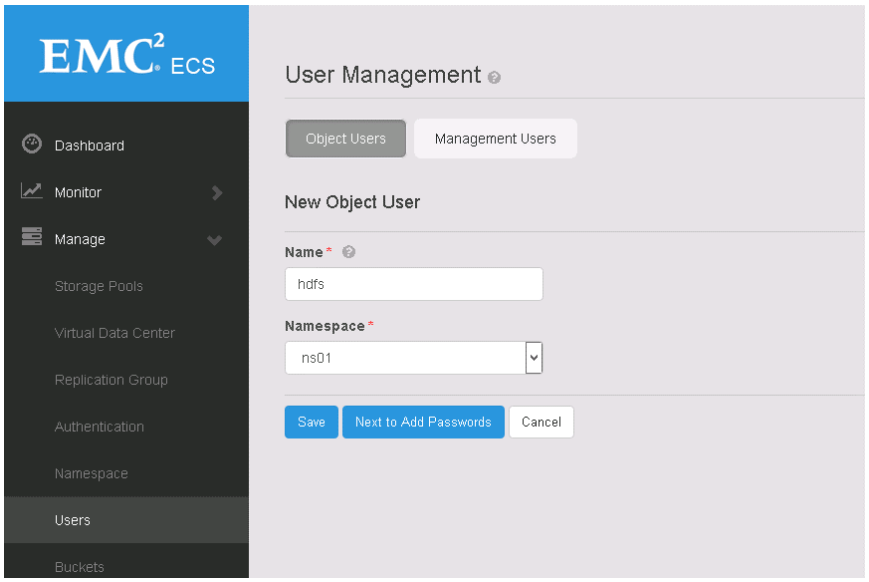


Figure 30. Creating user hdfs on ECS

5. Go to **Manage** > **Buckets** > **New Bucket**.

6. Create buckets hdp01 and hdp02.

   Figure 31 shows an example for hdp01. The hdp02 bucket uses the same configuration.

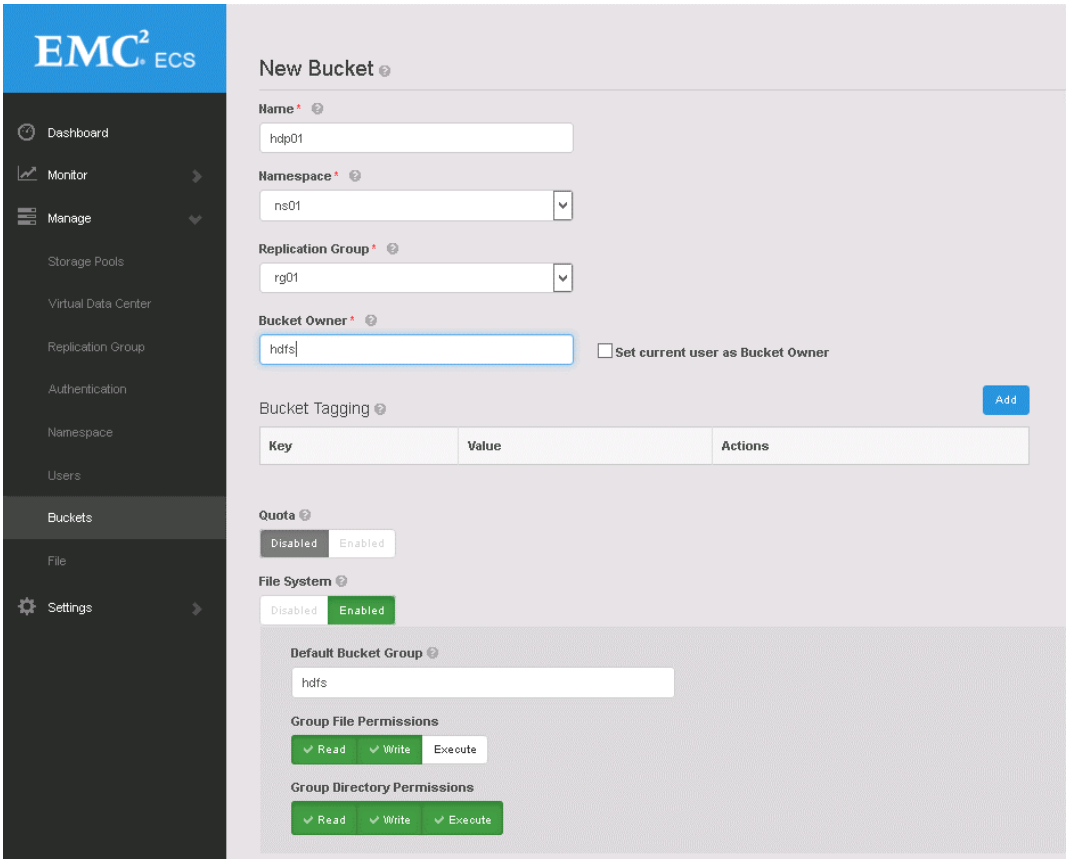


Figure 31. Creating ECS bucket hdp01

7. Set bucket access control lists (ACLs) for hdp01, as shown in Figure 32 through Figure 34.

Figure 32. User ACLs for bucket hdp01

Figure 33. Group ACLs for bucket hdp01

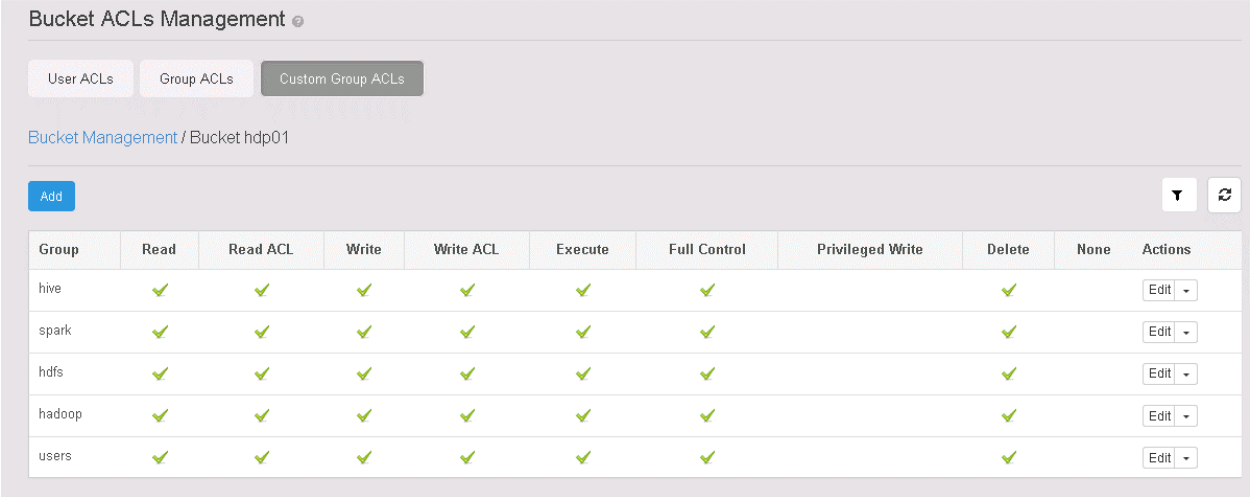


Figure 34. Custom Group ACLs for bucket hdp01

8. Repeat step 7, setting bucket ACLs for hdp02.

Installing ECS HDFS Client software

After you create the ECS buckets, install ECS HDFS Client on each HDP node and use Ambari to configure HDP to work with the ECS buckets, as follows.

For more details, see the instructions for configuring ECS HDFS integration with a simple Hadoop cluster in the *Elastic Cloud Storage Data Access Guide*.

1. Download the ECS HDFS Client Zip file from the following location:

   ```
   https://download.emc.com/downloads/DL85199_ECS_HDFS_Client_3.0.0.0.86889.0a0ee19.zip
   ```

2. Unzip the file to get the following ECS Client Library (viprfs) JAR file:

   ```
   /viprfs-client-3.0.0.0.86889.0a0ee19/client/viprfs-client-3.0.0.0-hadoop-2.7.jar
   ```

3. Create a directory on each HDP node, including the master node and worker nodes:

   ```
   /usr/lib/hadoop/lib
   ```

4. Upload the ECS Client Library (viprfs) JAR file to the directory on each of the HDP nodes.
5. Log in to the Ambari web console.
6. Set the configuration properties that are required by ECS HDFS Client, as shown in Table 10.

Table 10. Hadoop configuration to enable ECS access

Location	Property	Value
HDFS > Configs > Advanced > Custom core-site	fs.viprfs.impl	com.emc.hadoop.fs.vipr.ViPRFileSystem
	fs.AbstractFileSystem.viprfs.impl	com.emc.hadoop.fs.vipr.ViPRAbstractFileSystem
	fs.permissions.umask-mode	022
	fs.viprfs.auth.identity_translation	NONE
	fs.viprfs.auth.anonymous_translation	LOCAL_USER
	fs.vipr.installations	federation1 (this value can be any name and is referred to as $FEDERATION) If you have multiple independent ECS federations, type multiple values separated by commas.
	fs.vipr.installation.$FEDERATION.hosts	<comma-separated list of FQDN or IP address of each ECS host in the local site>
	fs.vipr.installation.$FEDERATION.hosts.resolution	dynamic
	fs.vipr.installation.$FEDERATION.resolution.dynamic.time_to_live_ms	900000
YARN > Configs > Advanced > Advanced yarn-site	yarn.application.classpath	Append the following: /usr/lib/hadoop/lib/*
MapReduce2 > Configs > Advanced > Advanced mapred-site	mapreduce.application.classpath	Append the following: /usr/lib/hadoop/lib/*
Tez > Configs > Advanced tez-site	tez.cluster.additional.classpath.prefix	Append the following: /usr/lib/hadoop/lib/*
HDFS > Configs > Advanced > Advanced hadoop-env	hadoop-env template	Append the following: export HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:/usr/lib/hadoop/lib/*
Spark > Configs > Advanced spark-env	spark-env template	Append the following: export SPARK_DIST_CLASSPATH="${SPARK_DIST_CLASSPATH}:/usr/lib/hadoop/lib/*:/usr/hdp/current/hadoop-client/client/guava.jar"
Hive > Configs > Advanced > Custom hive-env	fs.trash.interval	0
	hive.exim.uri.scheme.whitelist	hdfs,pfile,viprfs
Hive > Configs > Settings > ACID Transactions	ACID Transactions	On

7. In the Ambari web console, select **Actions** > **Restart All Required**.

Enabling Kerberos on the HDP cluster

After you install ECS HDFS Client software, enable Kerberos on the HDP cluster. See Enabling Kerberos on the HDP cluster on page 28.

Enabling Kerberos on the ECS cluster

After you enable Kerberos on HDP, configure ECS buckets to work with Kerberos, as follows.

For more details, see the instructions for configuring ECS HDFS integration with a secure (Kerberized) Hadoop cluster in the *Elastic Cloud Storage Data Access Guide*.

1. Ensure that the ECS nodes use the same DNS resolution as used by the HDP cluster.
2. Copy the `hdfsclient-3.0.0.0.86889.0a0ee19.zip` to the following directory on ECS node 1:

   ```
   /home/admin/ansible
   ```

3. Unzip the Zip file, and edit `inventory.txt` in the `playbooks/samples` directory to refer to the ECS nodes and KDC server:

   ```
   [data_nodes]
   <FQDN of ECS nodes>

   [kdc]
   <IP of KDC server>
   ```

4. If you are using strong encryption, download `UnlimitedJCEPolicyJDK7.zip` and extract it to an `UnlimitedJCEPolicy` directory in `playbooks/samples`.

   Note: Perform this step only if you are using strong encryption.

5. Start the utility container on ECS node 1 and make the Ansible playbooks available to the container:

   a. Load the utility container image:

   ```
   sudo docker load -i
   /opt/emc/caspian/checker/docker/images/utilities.txz
   ```

   b. Get the identity of the docker image:

   ```
   sudo docker images
   REPOSITORY                  TAG                         IMAGE
   ID            CREATED              VIRTUAL SIZE
   utilities                   1.5.0.0-432.2ae21ed
   be5eee0f7494        11 months ago       740.7 MB
   ```

c. Start and enter the utilities image:

```
sudo docker run -v
/opt/emc/caspian/fabric/agent/services/object/main/log:/
opt/storageos/logs -v /home/admin/ansible/viprfs-client-
3.0.0.0.86889.0a0ee19/playbooks:/ansible --name=ecs-
tools -i -t --privileged --net=host be5eee0f7494
/bin/bash
```

6. Change to the working directory in the container:

```
cd /ansible
```

7. Copy the `krb5.conf` file from the KDC to the working directory.

   If the DNS resolution is not working correctly, change the `kdc` and `admin_server` in the `krb5.conf` from an FQDN to an IP address.

8. Install the supplied Ansible roles:

```
ansible-galaxy install -r requirements.txt –f
```

9. Edit the `generate-vipr-keytabs.yml` file as necessary and set the domain name:

```
cat generate-vipr-keytabs.yml
---
###
# Generates keytabs for ViPR/ECS data nodes.
###

- hosts: data_nodes
  serial: 1

  roles:
    - role: vipr_kerberos_principal
      kdc: "{{ groups.kdc | first }}"
      principals:
        - name:  vipr/_HOST@BIGDATA.EMC.LOCAL
          keytab: keytabs/_HOST@BIGDATA.EMC.LOCAL.keytab
hop-u300-02-pub-01:/ansible # cat setup-vipr-kerberos.yml
```

Note: In this example, the default value (`vipr/_HOST@EXAMPLE.COM`) has been replaced with `vipr/_HOST@BIGDATA.EMC.LOCAL`, and the domain is `BIGDATA.EMC.LOCAL`.

10. Type the following command:

```
export ANSIBLE_HOST_KEY_CHECKING=False
```

11. Run the Ansible playbook to generate keytabs:

```
ansible-playbook -v -k -i inventory.txt --user admin –b --
become-user=root generate-vipr-keytabs.yml
```

12. Edit the `setup-vipr-kerberos.yml` file as necessary:

```
cat setup-vipr-kerberos.yml
###
# Configures ViPR/ECS for Kerberos authentication.
# - Configures krb5 client
# - Installs keytabs
# - Installs JCE policy
###

 - hosts: data_nodes

   roles:
     - role: vipr_kerberos_config
       krb5:
         config_file: krb5.conf
       service_principal:
         name: vipr/_HOST@BIGDATA.EMC.LOCAL
         keytab: keytabs/_HOST@BIGDATA.EMC.LOCAL.keytab

     - role: vipr_jce_config
       jce_policy:
         name: unlimited
         src: UnlimitedJCEPolicy/
```

Note: In this example, the default value (`vipr/_HOST@EXAMPLE.COM`) has been replaced with `vipr/_HOST@BIGDATA.EMC.LOCAL` and the domain is `BIGDATA.EMC.LOCAL`.

Note: Remove the `vipr_jce_config` role if you are not using strong encryption.

13. Run the Ansible playbook to configure the DataNodes with the ECS service principal:

```
ansible-playbook -v -k -i inventory.txt --user admin –b --
become-user=root setup-vipr-kerberos.yml
```

14. Verify that the correct ECS service principal, one per DataNode, has been created from the KDC:

```
kadmin.local -q "list_principals" | grep vipr
vipr/hop-u300-02-pub-
01.solarch.lab.emc.com@BIGDATA.EMC.LOCAL
vipr/hop-u300-02-pub-
02.solarch.lab.emc.com@BIGDATA.EMC.LOCAL
vipr/hop-u300-02-pub-
03.solarch.lab.emc.com@BIGDATA.EMC.LOCAL
vipr/hop-u300-02-pub-04.solarch.lab.emc.com@BIGDATA.EMC.LOCAL
```

15. Log in to ECS node 1 using the admin credentials.

16. Create a JSON file that contains metadata as described in the instructions for securing the ECS bucket by using metadata in the *Elastic Cloud Storage Data Access Guide*.

```
cat hdp.ns01.json
{
        "head_type": "hdfs",
        "metadata": [
                {
                        "name":
"internal.kerberos.user.hdfs.name",
                        "value": "hdfs-
hdp4@BIGDATA.EMC.LOCAL"
                },
                {
                        "name":
"internal.kerberos.user.hdfs.shortname",
                        "value": "hdfs"
                },
/                {
                        "name":
"internal.kerberos.user.hdfs.groups",
                        "value": "hadoop,hdfs"
                },
                {
                        "name":
"dfs.permissions.supergroup",
                        "value": "hdfs"
                },
{
 "name": "hadoop.proxyuser.hive.hosts",
 "value": "*"
},
{
 "name": "hadoop.proxyuser.hive.groups",
 "value": "*"
}
        ]
}
```

Note: Information about every Kerberos user who requires Hadoop access to a bucket must be uploaded to ECS.

17. Run the ECS Management REST API command to deploy the metadata for ECS bucket hdp01:

```
CLUSTER_NAME=hdp01
ECS_HOST=<FQDN of ECS node 1>
ECS_NS=ns01
ECS_BUCKET=hdp01
ECS_LOGIN=<user>:<password>
```

```
ECS_WS_URL=https://$ECS_HOST:4443
ECS_FED=federation1
FS=viprfs://$ECS_BUCKET.$ECS_NS.$ECS_FED/
ECS_TOKEN=$(curl -s -k -u $ECS_LOGIN -D - -o /dev/null
$ECS_WS_URL/login | grep X-SDS-AUTH-TOKEN | tr -cd '\40-
\176')
curl -s -k -X PUT -H "$ECS_TOKEN" -H "Accept:
application/json" -H "Content-Type: application/json"  -T
hdp.ns01.json
$ECS_WS_URL/object/bucket/$ECS_BUCKET/metadata?namespace=$ECS_
NS
```

18. Run the ECS Management REST API command to deploy the metadata for ECS bucket hdp02:

```
CLUSTER_NAME=hdp02
ECS_HOST=<FQDN of ECS node 1>
ECS_NS=ns01
ECS_BUCKET=hdp02
ECS_LOGIN=<user>:<password>
ECS_WS_URL=https://$ECS_HOST:4443
ECS_FED=federation1
FS=viprfs://$ECS_BUCKET.$ECS_NS.$ECS_FED/
ECS_TOKEN=$(curl -s -k -u $ECS_LOGIN -D - -o /dev/null
$ECS_WS_URL/login | grep X-SDS-AUTH-TOKEN | tr -cd '\40-
\176')
curl -s -k -X PUT -H "$ECS_TOKEN" -H "Accept:
application/json" -H "Content-Type: application/json"  -T
hdp.ns01.json
$ECS_WS_URL/object/bucket/$ECS_BUCKET/metadata?namespace=$EC
S_NS
```

19. Log in to the Ambari web console, and set the configuration properties that ECS HDFS Client requires, as shown in Table 11.

Table 11. Hadoop Configuration to Enable Kerberos with ECS bucket

Location	Property	Value
HDFS > Configs > Advanced > Custom core-site	fs.viprfs.auth.identity_translation	CURRENT_USER_REALM
	hadoop.security.kerberos.ticket.cache.path	/tmp/krb5cc_1007 Obtain the value from the output of the `klist` command.
	hadoop.proxyuser.hive.groups	*
	hadoop.proxyuser.hive.hosts	*

20. In the Ambari web console, select **Actions** > **Restart All Required**.

Validating HDP deployment and ECS integration

Overview

Table 12 lists the validation procedures for the deployment of Hadoop tiered storage and integration with an ECS cluster.

Table 12. Validating HDP cluster deployment and integration with ECS cluster

Step	Category	Validation procedures
1	Ambari	Install/configure validation and GUI functionality testing
2	Kerberos security	Kerberos and non-Kerberos user testing on Kerberized Hadoop and Isilon clusters
3	DistCp in Kerberized and non-Kerberized clusters	DistCp operation testing in and between local DAS HDFS and remote ECS tier HDFS

Validation procedures

Ambari

The steps for performing an Ambari smoke test are as follows.

Note: For screenshots of our Ambari smoke test, see Appendix A on page 81. For a list of specific test steps, see Appendix C on page 98.

1. Set up a five-node HDP cluster for local DAS HDFS and a four-node ECS cluster as the remote HDFS cluster.
2. In the Ambari Server web UI, run all the service checks, and stop and restart all the services.

Note: ECS does not support Ranger access policies.

Kerberos security

The steps for performing Kerberos security testing are as follows.

Note: For a list of specific test steps, see Appendix C on page 98.

1. Kerberize the local DAS HDFS cluster and remote ECS HDFS cluster, and create hdp_user2 on all the nodes of the local DAS HDFS cluster.

   Do not add the hdp_user2 principal to the Kerberos KDC server and try to access the local DAS HDFS cluster. Permission will be denied.

```
[root@hdp-worker12 ~]# su hdp_user2
[hdp_user2@hdp-worker12 root]$ kinit

kinit: Client 'hdp_user2@BIGDATA.EMC.LOCAL' not found in
Kerberos database while getting initial credentials

[hdp_user2@hdp-worker12 root]$ klist
klist: Credentials cache file '/tmp/krb5cc_1013' not found

[hdp_user2@hdp-worker12 root]$ hadoop fs -ls /
```

```
17/08/25 16:52:51 WARN ipc.Client: Exception encountered
while connecting to the server :
javax.security.sasl.SaslException: GSS initiate failed
[Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
        at
com.sun.security.sasl.gsskerb.GssKrb5Client.evaluateChalleng
e(GssKrb5Client.java:211)
            .
            .
        at
com.sun.security.sasl.gsskerb.GssKrb5Client.evaluateChalleng
e(GssKrb5Client.java:192)
        ... 41 more
ls: Failed on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed
[Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]; Host
Details : local host is: "hdp-
worker12.bigdata.emc.local/172.16.1.69"; destination host
is: "hdp-master04.bigdata.emc.local":8020;
```

2. Create and add the user hdp_user1 principal to the Kerberos KDC server, assign a password, and access the local DAS HDFS cluster:

```
[hdfs@hdp-master04 root]$ hadoop fs -ls /user/
Found 7 items
drwxrwx---   - ambari-qa hdfs          0 2017-08-22 14:03
/user/ambari-qa
drwxr-xr-x   - hcat      hdfs          0 2017-07-19 16:06
/user/hcat
drwxr-xr-x   - hdfs      hdfs          0 2017-08-08 15:58
/user/hdfs
drwxr-xr-x   - hdp_user1 hdfs          0 2017-08-23 16:26
/user/hdp_user1
drwxr-xr-x   - hive      hdfs          0 2017-08-22 10:28
/user/hive
drwxrwxr-x   - livy      hdfs          0 2017-07-19 16:04
/user/livy
drwxrwxr-x   - spark     hdfs          0 2017-07-19 16:04
/user/spark
```

3. Access the remote ECS cluster using the hdp-user1:

```
[hdfs@hdp-worker12 root]$ fs=viprfs://hdp01.ns01.federation1
[hdfs@hdp-worker12 root]$ hadoop fs -ls $fs/
Found 2 items
drwxr-xr-x   - hdfs hdfs          0 2017-08-22 10:18
viprfs://hdp01.ns01.federation1/tmp
drwxrwxrwx   - hdfs hdfs          0 2017-08-17 16:25
viprfs://hdp01.ns01.federation1/user
```

DistCp in Kerberized and non-Kerberized clusters

Perform DistCp testing in Kerberized and non-Kerberized clusters as follows:

1. Run `disctp` to copy sample files from the local DAS HDFS to the remote ECS HDFS in a non-Kerberized cluster:

```
sudo -u hdfs hadoop distcp -skipcrccheck -update
/tmp/redhat-release viprfs://hdp01.ns01.federation1/tmp/
sudo -u hdfs hdfs dfs -ls -R
viprfs://hdp01.ns01.federation1/tmp/redhat-release
-rw-r--r--   3 root hdfs         27 2017-09-07 13:26
viprfs://hdp01.ns01.federation1/tmp/redhat-release
```

2. Run `disctp` to copy sample files from the remote ECS HDFS to the local DAS HDFS in a Kerberized cluster:

```
sudo -u hdfs hadoop distcp -pc
viprfs://hdp01.ns01.federation1/tmp/redhat-release
hdfs://hdp-master04.bigdata.emc.local:8020/tmp/
sudo -u hdfs hdfs dfs -ls -R hdfs://hdp-
master04.bigdata.emc.local:8020/tmp/redhat-release
-rw-r--r--   3 hdfs hdfs         27 2017-09-07 13:26
hdfs://hdp-master04.bigdata.emc.local:8020/tmp/redhat-
release
```

Validation results

Table 13 describes detailed results of the validation procedures as PASS/FAIL.

Table 13. Test cases, expected results, and status

Category	Test case name	Expected test results	Status
Ambari	Validation of install/configuration	Deployment is completed successfully	PASS
	Usability and functionality test of GUI	Service check runs successfully	PASS
MapReduce	Word count	Command runs successfully	PASS
Spark	Line count and word count	Command runs successfully	PASS
Hive/Tez and Hive/MapReduce	DDL operations 1. LOAD data local inpath 2. INSERT into table 3. INSERT Overwrite TABLE	Command runs successfully	PASS
	DML operations 1. Query local database tables 2. Query remote database tables	Command runs successfully	PASS
	JOIN tables in local database	Command runs successfully	PASS
	JOIN tables in remote database	Command runs successfully	PASS
	JOIN tables between local_db and remote_db	Command runs successfully	PASS

Category	Test case name	Expected test results	Status
	Local temporary table from remote_db table	Command runs successfully	PASS
	IMPORT and EXPORT operations	Command runs successfully	PASS
	Table-level and column-level statistics	Command runs successfully	PASS
HIVE TPC-DS	Prepare Hive-testbench	Command runs successfully	PASS
	Database creation on remote hadoop cluster, and data load	Command runs successfully	PASS
	TPC-DS benchmarking	Command runs successfully	PASS
Kerberos environment configuration	Kerberos setup	Command runs successfully	PASS
Kerberos users	Kerberos user setup and testing	Command runs successfully	PASS
	Non-Kerberos user setup and testing	Command runs successfully	PASS
Kerberos security	MapReduce > WordCount	Command runs successfully	PASS
	Spark > LineCount & WordCount	Command runs successfully	PASS
Kerberos security on Hive warehouse	DDL operations 1. LOAD data local inpath 2. INSERT into table 3. INSERT Overwrite TABLE	Command runs successfully	PASS
	DML operations 1. Query local database tables 2. Query remote database tables	Command runs successfully	PASS
	JOIN tables in local database	Command runs successfully	PASS
	JOIN tables in remote database	Command runs successfully	PASS
	JOIN tables between local_db and remote_db	Command runs successfully	PASS
	Local temporary table from remote_db table	Command runs successfully	PASS
	IMPORT and EXPORT operations	Command runs successfully	PASS
	Table-level and column-level statistics	Command runs successfully	PASS
DistCp in Kerberized and non-Kerberized cluster	DisctCp and script	Command runs successfully	PASS

Chapter 6 Sample Use Cases: MapReduce, Spark, and Hive

This chapter presents the following topics:

Isilon use cases

MapReduce

The steps for performing MapReduce testing are as follows.

Note: For a list of specific test steps, see Appendix B on page 87.

1. Run a MapReduce WordCount job on input from the local DAS HDFS, with output to the remote Isilon HDFS2 cluster:

```
sudo -u hdfs yarn jar /usr/hdp/current/hadoop-mapreduce-
client/hadoop-mapreduce-examples.jar wordcount hdfs://hdp-
master03.bigdata.emc.local:8020/tmp/mr/redhat-release
hdfs://isi-cluster-
hdfs2.bigdata.emc.local:8020/tmp/mr/redhat-release-das-hdfs2
sudo -u hdfs hdfs dfs -ls -R hdfs://isi-cluster-
hdfs2.bigdata.emc.local:8020/tmp/mr/ redhat-release-das-
hdfs2
drwxr-xr-x   - root hdfs          0 2017-08-04 01:49
hdfs://isi-cluster-
hdfs2.bigdata.emc.local:8020/tmp/mr/redhat-release-das-hdfs2
-rw-r--r--   3 root hdfs          0 2017-08-04 01:49
hdfs://isi-cluster-
hdfs2.bigdata.emc.local:8020/tmp/mr/redhat-release-das-
hdfs2/_SUCCESS
-rw-r--r--   3 root hdfs         68 2017-08-04 01:49
hdfs://isi-cluster-
hdfs2.bigdata.emc.local:8020/tmp/mr/redhat-release-das-
hdfs2/part-r-00000
```

2. Run a MapReduce WordCount job on input from the remote Isilon HDFS2, with output to the local DAS HDFS:

```
sudo -u hdfs yarn jar /usr/hdp/current/hadoop-mapreduce-
client/hadoop-mapreduce-examples.jar wordcount hdfs://isi-
cluster-hdfs2.bigdata.emc.local:8020/tmp/mr/redhat-release
hdfs://hdp-master03.bigdata.emc.local:8020/tmp/mr/redhat-
release-hdfs2-das

sudo -u hdfs hdfs dfs -ls -R hdfs://hdp-
master03.bigdata.emc.local:8020/tmp/mr/
drwxr-xr-x   - hdfs hdfs          0 2017-08-04 09:50
hdfs://hdp-master03.bigdata.emc.local:8020/tmp/mr/redhat-
release-hdfs2-das
-rw-r--r--   3 hdfs hdfs          0 2017-08-04 09:50
hdfs://hdp-master03.bigdata.emc.local:8020/tmp/mr/redhat-
release-hdfs2-das/_SUCCESS
-rw-r--r--   3 hdfs hdfs         68 2017-08-04 09:50
hdfs://hdp-master03.bigdata.emc.local:8020/tmp/mr/redhat-
release-hdfs2-das/part-r-00000
```

Spark

The steps for performing Spark testing are as follows.

Note: For a list of specific test steps, see Appendix B on page 87.

1. Create word count and line count Scala files for Spark testing:

```
cat >/tmp/spark_line_word_count.scala <<EOF
val args=sc.getConf.get("spark.driver.args").split("\\\\s+")
var input=args(0)
var output1=args(1) + "-wc"
var text_file=sc.textFile(input)
val word_count=text_file.flatMap(line => line.split("
")).map(word => (word, 1)).reduceByKey(_ + _)
word_count.saveAsTextFile(output1)
var output2=args(1) + "-lc"
var line_count=sc.parallelize(Seq(text_file.count()))
line_count.saveAsTextFile(output2)
exit
EOF
```

2. Run the Spark shell to perform a word count and line count on input from the local DAS HDFS, with output to the remote Isilon HDFS cluster:

```
sudo -u hdfs spark-shell -i /tmp/spark_line_word_count.scala
--conf 'spark.driver.args=hdfs://hdp-
master03.bigdata.emc.local:8020/tmp/spark/redhat-release
hdfs://isi-cluster-
hdfs1.bigdata.emc.local:8020/tmp/spark/redhat-release-das-
hdfs1'

sudo -u hdfs spark-shell -i /tmp/spark_line_word_count.scala
--conf 'spark.driver.args=hdfs://isi-cluster-
hdfs1.bigdata.emc.local:8020/tmp/spark/redhat-release
hdfs://hdp-master03.bigdata.emc.local:8020/tmp/spark/redhat-
release-hdfs1-das'
```

Hive on Tez execution engine

The steps for performing Hive testing on the Tez execution engine are as follows.

Note: For a list of specific test steps, see Appendix B on page 87.

1. Create a remote database location on the remote Isilon HDFS cluster and create an internal partitioned table:

```
CREATE database remote_db COMMENT 'Holds all the tables data
in remote location Hadoop cluster' LOCATION 'hdfs://isi-
cluster-hdfs1.bigdata.emc.local:8020/user/hive/remote_db'
OK
Time taken: 0.081 seconds

USE remote_db
OK
```

```
Time taken: 0.03 seconds

CREATE TABLE passwd_int_part (user_name STRING, password
STRING, user_id STRING, user_id_info STRING, home_dir
STRING, shell STRING) PARTITIONED BY (group_id STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ':'
OK
Time taken: 0.074 seconds
```

2. Create a local database location on the local DAS HDFS cluster, and create an internal transactional table:

```
CREATE database local_db COMMENT 'Holds all the tables data
in local Hadoop cluster' LOCATION 'hdfs://hdp-
master03.bigdata.emc.local:8020/user/hive/local_db'
OK
Time taken: 0.066 seconds

USE local_db
OK
Time taken: 0.013 seconds
CREATE TABLE passwd_int_trans (user_name STRING, password
STRING, user_id STRING, group_id STRING, user_id_info
STRING, home_dir STRING, shell STRING) CLUSTERED
by(user_name) into 3 buckets stored as orc tblproperties
("transactional"="true")
OK
Time taken: 0.062 seconds
```

Hive on MapReduce execution engine

The steps for performing Hive testing on the MapReduce execution engine are as follows.

Note: For a list of the specific tests steps, see Appendix B on page 87.

1. Create a local database location on the local DAS HDFS, and create an internal nonpartioned remote table:

```
CREATE database local_db COMMENT 'Holds all the tables data
in local Hadoop cluster' LOCATION 'hdfs://hdp-
master03.bigdata.emc.local:8020/user/hive/local_db'
OK
Time taken: 0.066 seconds

USE local_db
OK
Time taken: 0.013 seconds
CREATE TABLE passwd_int_nonpart_remote (user_name STRING,
password STRING, user_id STRING, group_id STRING,
user_id_info STRING, home_dir STRING, shell STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ':' LOCATION
'hdfs://hdp-
```

```
master03.bigdata.emc.local:8020/user/hive/local_db/passwd_in
t_nonpart_remote'
OK
Time taken: 0.075 seconds
```

2. Create an external nonpartitioned table:

```
CREATE EXTERNAL TABLE passwd_ext_nonpart (user_name STRING,
password STRING, user_id STRING, group_id STRING,
user_id_info STRING, home_dir STRING, shell STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ':' LOCATION
'hdfs://hdp-
master03.bigdata.emc.local:8020/user/hive/local_db/passwd_in
t_nonpart_remote'
OK
Time taken: 0.056 seconds
```

Hive TPC-DS

The steps for performing Hive TPC-DS benchmark testing are as follows.

Note: For a list of specific test steps, see Appendix B on page 87.

1. Prepare Hive-testbench, run `tpcdc-build.sh` to build TPC-DS and the data generator, run `tpcds-setup` to set up the testbench database, and load the data into the created tables:

```
sudo ./tpcds-build.sh
sudo ./tpcds-setup.sh 5
(A map reduce job runs to create the data and load the data
into hive. This will take some time to complete. The last
line in the script is: Data loaded into database
tpcds_bin_partitioned_orc_5.)
```

2. Create a new remote Low Latency Analytical Processing (LLAP) database on the remote Isilon HDFS:

```
DROP database if exists llap CASCADE;
CREATE database if not exists llap LOCATION 'hdfs://isi-
cluster-hdfs1.bigdata.emc.local:8020/user/hive/llap.db';

drop table if exists llap.call_center;
create table llap.call_center
stored as orc
as select * from tpcds_text_5.call_center;
```

3. Create 24 tables and load data from the tables that you previously created.
4. Run the benchmark queries on the tables that you created on the remote LLAP database:

```
hive> use llap;
hive> source query52.sql;
hive> source query55.sql;
hive> source query91.sql;
```

```
hive> source query42.sql;
hive> source query12.sql;
hive> source query73.sql;
hive> source query20.sql;
hive> source query3.sql;
hive> source query89.sql;
hive> source query48.sql;
```

ECS use cases

MapReduce

The steps for performing MapReduce testing are as follows.

Note: For a list of specific test steps, see Appendix C on page 98.

1. Run a MapReduce WordCount job on input from the local DAS HDFS, with output to the remote ECS HDFS2 cluster:

```
sudo -u hdfs yarn jar /usr/hdp/current/hadoop-mapreduce-
client/hadoop-mapreduce-examples.jar wordcount hdfs://hdp-
master04.bigdata.emc.local:8020/tmp/mr/redhat-release
viprfs://hdp01.ns01.federation1/tmp/mr/redhat-release-das-
hdfs2

sudo -u hdfs hdfs dfs -ls -R
viprfs://hdp01.ns01.federation1/tmp/mr/redhat-release-das-
hdfs2

drwxr-xr-x   - root hdfs          0 2017-08-04 01:49
viprfs://hdp01.ns01.federation1/tmp/mr/redhat-release-das-
hdfs2
-rw-r--r--   3 root hdfs          0 2017-08-04 01:49
viprfs://hdp01.ns01.federation1/tmp/mr/redhat-release-das-
hdfs2/_SUCCESS
-rw-r--r--   3 root hdfs         68 2017-08-04 01:49
viprfs://hdp01.ns01.federation1/tmp/mr/redhat-release-das-
hdfs2/part-r-00000
```

2. Run a MapReduce WordCount job on input from the remote ECS HDFS2, with output to the local DAS HDFS:

```
sudo -u hdfs yarn jar /usr/hdp/current/hadoop-mapreduce-
client/hadoop-mapreduce-examples.jar wordcount
viprfs://hdp01.ns01.federation1/tmp/mr/redhat-release
hdfs://hdp-master04.bigdata.emc.local:8020/tmp/mr/redhat-
release-hdfs1-das

sudo -u hdfs hdfs dfs -ls -R hdfs://hdp-
master04.bigdata.emc.local:8020/tmp/mr/
```

```
drwxr-xr-x   - hdfs hdfs          0 2017-08-04 09:50
hdfs://hdp-master04.bigdata.emc.local:8020/tmp/mr/redhat-
release-hdfs2-das
-rw-r--r--   3 hdfs hdfs          0 2017-08-04 09:50
hdfs://hdp-master04.bigdata.emc.local:8020/tmp/mr/redhat-
release-hdfs2-das/_SUCCESS
-rw-r--r--   3 hdfs hdfs         68 2017-08-04 09:50
hdfs://hdp-master04.bigdata.emc.local:8020/tmp/mr/redhat-
release-hdfs2-das/part-r-00000
```

Spark

The steps for performing Spark testing are as follows.

Note: For a list of specific test steps, see Appendix C on page 98.

1. Create word count and line count Scala files for Spark testing:

```
cat >/tmp/spark_line_word_count.scala <<EOF
val args=sc.getConf.get("spark.driver.args").split("\\\\s+")
var input=args(0)
var output1=args(1) + "-wc"
var text_file=sc.textFile(input)
val word_count=text_file.flatMap(line => line.split("
")).map(word => (word, 1)).reduceByKey(_ + _)
word_count.saveAsTextFile(output1)
var output2=args(1) + "-lc"
var line_count=sc.parallelize(Seq(text_file.count()))
line_count.saveAsTextFile(output2)
exit
EOF
```

2. Run the Spark shell to perform a word count and line count on input from the local DAS HDFS, with output to the remote ECS HDFS cluster.

```
sudo -u hdfs spark-shell -i /tmp/spark_line_word_count.scala
--conf 'spark.driver.args=hdfs://hdp-
master04.bigdata.emc.local:8020/tmp/spark/redhat-release
viprfs://hdp01.ns01.federation1/tmp/spark/redhat-release-
das-hdfs1'

sudo -u hdfs spark-shell -i /tmp/spark_line_word_count.scala
--conf 'spark.driver.args=
viprfs://hdp01.ns01.federation1/tmp/spark/redhat-release
hdfs://hdp-master04.bigdata.emc.local:8020/tmp/spark/redhat-
release-hdfs1-das'
```

Hive on Tez execution engine

The steps for performing Hive testing on the Tez execution engine are as follows.

Note: For a list of the specific tests steps, see Appendix C on page 98.

1. Create a remote database location on the remote isilon HDFS cluster and create an internal partitioned table:

```
CREATE database remote_db COMMENT 'Holds all the tables data
in remote location Hadoop cluster' LOCATION
'viprfs://hdp01.ns01.federation1/user/hive/remote_db'
OK
Time taken: 0.277 seconds

USE remote_db
OK
Time taken: 0.062 seconds

CREATE TABLE passwd_int_part (user_name STRING, password
STRING, user_id STRING, user_id_info STRING, home_dir
STRING, shell STRING) PARTITIONED BY (group_id STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ':'
OK
Time taken: 0.223 seconds
```

2. Create a local database location on the local DAS HDFS cluster and create an internal transactional table:

```
CREATE database local_db COMMENT 'Holds all the tables data
in local Hadoop cluster' LOCATION 'hdfs://hdp-
master04.bigdata.emc.local:8020/user/hive/local_db'
OK
Time taken: 0.066 seconds

USE local_db
OK
Time taken: 0.013 seconds
CREATE TABLE passwd_int_trans (user_name STRING, password
STRING, user_id STRING, group_id STRING, user_id_info
STRING, home_dir STRING, shell STRING) CLUSTERED
by(user_name) into 3 buckets stored as orc tblproperties
("transactional"="true")
OK
Time taken: 0.062 seconds
```

Hive on MapReduce execution engine

The steps for performing Hive testing on the MapReduce execution engine are as follows.

Note: For a list of specific test steps, see Appendix C on page 98.

1. Create a local database location on the local DAS HDFS, and create an internal nonpartitioned remote table:

```
CREATE database local_db COMMENT 'Holds all the tables data
in local Hadoop cluster' LOCATION 'hdfs://hdp-
master04.bigdata.emc.local:8020/user/hive/local_db'
OK
Time taken: 0.066 seconds

USE local_db
OK
Time taken: 0.013 seconds
CREATE TABLE passwd_int_nonpart_remote (user_name STRING,
password STRING, user_id STRING, group_id STRING,
user_id_info STRING, home_dir STRING, shell STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ':' LOCATION
'hdfs://hdp-
master03.bigdata.emc.local:8020/user/hive/local_db/passwd_in
t_nonpart_remote'
OK
Time taken: 0.075 seconds
```

2. Create an external nonpartitioned table:

```
CREATE EXTERNAL TABLE passwd_ext_nonpart (user_name STRING,
password STRING, user_id STRING, group_id STRING,
user_id_info STRING, home_dir STRING, shell STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ':' LOCATION
'hdfs://hdp-
master04.bigdata.emc.local:8020/user/hive/local_db/passwd_in
t_nonpart_remote'
OK
Time taken: 0.056 seconds
```

Hive TPC-DS

For details about Hive TPC-DS benchmark testing, see Hive TPC-DS on page 72.

Chapter 7 Conclusion

This chapter presents the following topics:

Summary

Businesses of all sizes must be able to increase their analytics capability to lower operational expenses and improve the customer experience. Most enterprises cannot afford to risk success by implementing homegrown solutions. Hortonworks, in partnership with Dell EMC, offers a documented set of proven configurations with functional validations that operate and scale to all customer needs, with an integrated set of technologies and detailed deployment and implementation guidance. Our approach provides a low-risk option with fast time to value.

This solution provides Hortonworks validated system configurations for DAS storage for Hadoop with an Isilon or ECS infrastructure cluster to support big data analytics. With this solution, you can accommodate your current needs with an approved configuration that you can easily scale to meet future requirements. These configurations are widely applicable, cost-effective, and easy to implement and support.

Chapter 8 References

This chapter presents the following topics:

Dell EMC documentation

The following documentation on DellEMC.com or Dell EMC Online Support provides additional and relevant information. Access to these documents depends on your login credentials. If you do not have access to a document, contact your Dell EMC representative.

- *Elastic Cloud Storage (ECS) Data Access Guide*
- *EMC Isilon OneFS with Hadoop and Hortonworks for Kerberos Installation Guide*

Hortonworks documentation

The following documentation on the Hortonworks website provides additional and relevant information:

- *Apache Ambari Installation*
- *Apache Ambari Security*
- *Hortonworks Data Platform—Security*

VMware documentation

The following documentation on the VMware website provides additional and relevant information:

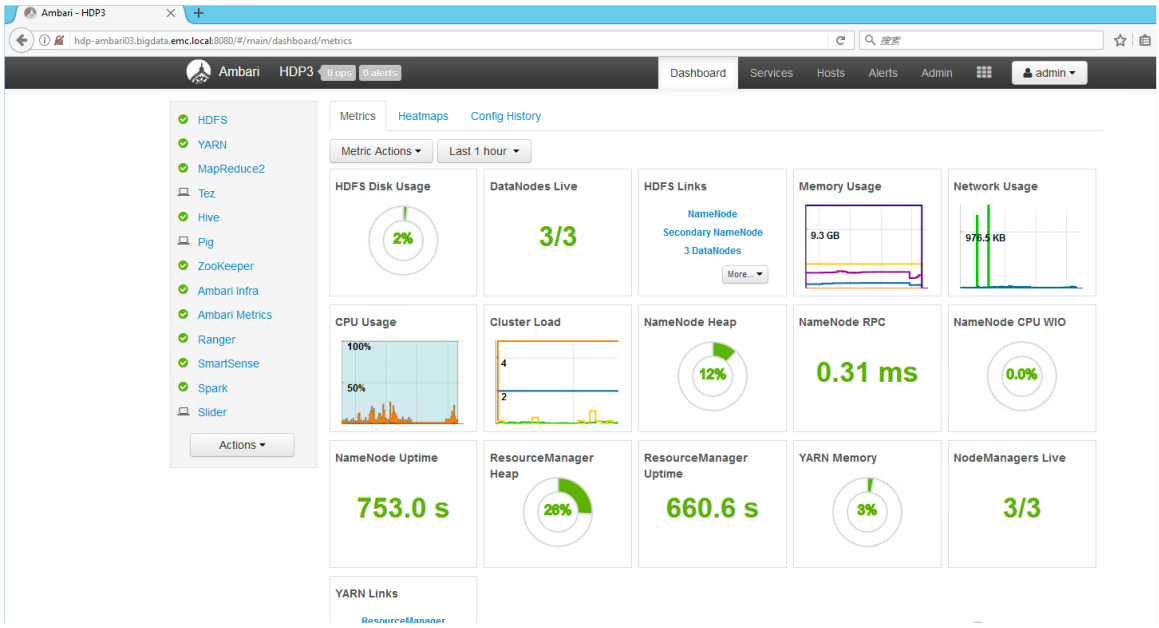
- *VMware Virtual SAN 6.0 Performance—Scalability and Best Practices Technical White Paper*
- *Performance Best Practices for VMware vSphere 6.0*

Appendix A Ambari Smoke Test Screenshots

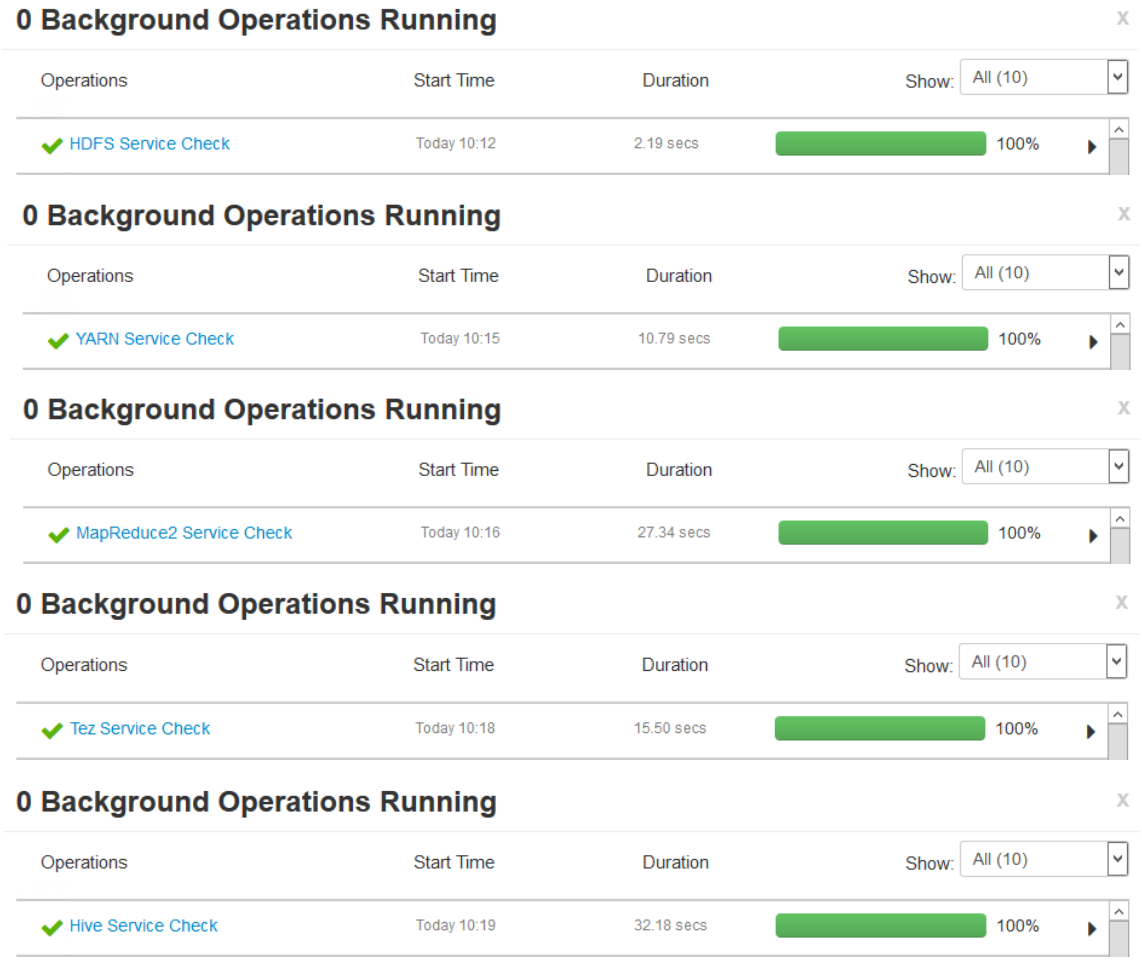
This appendix presents the following topics:

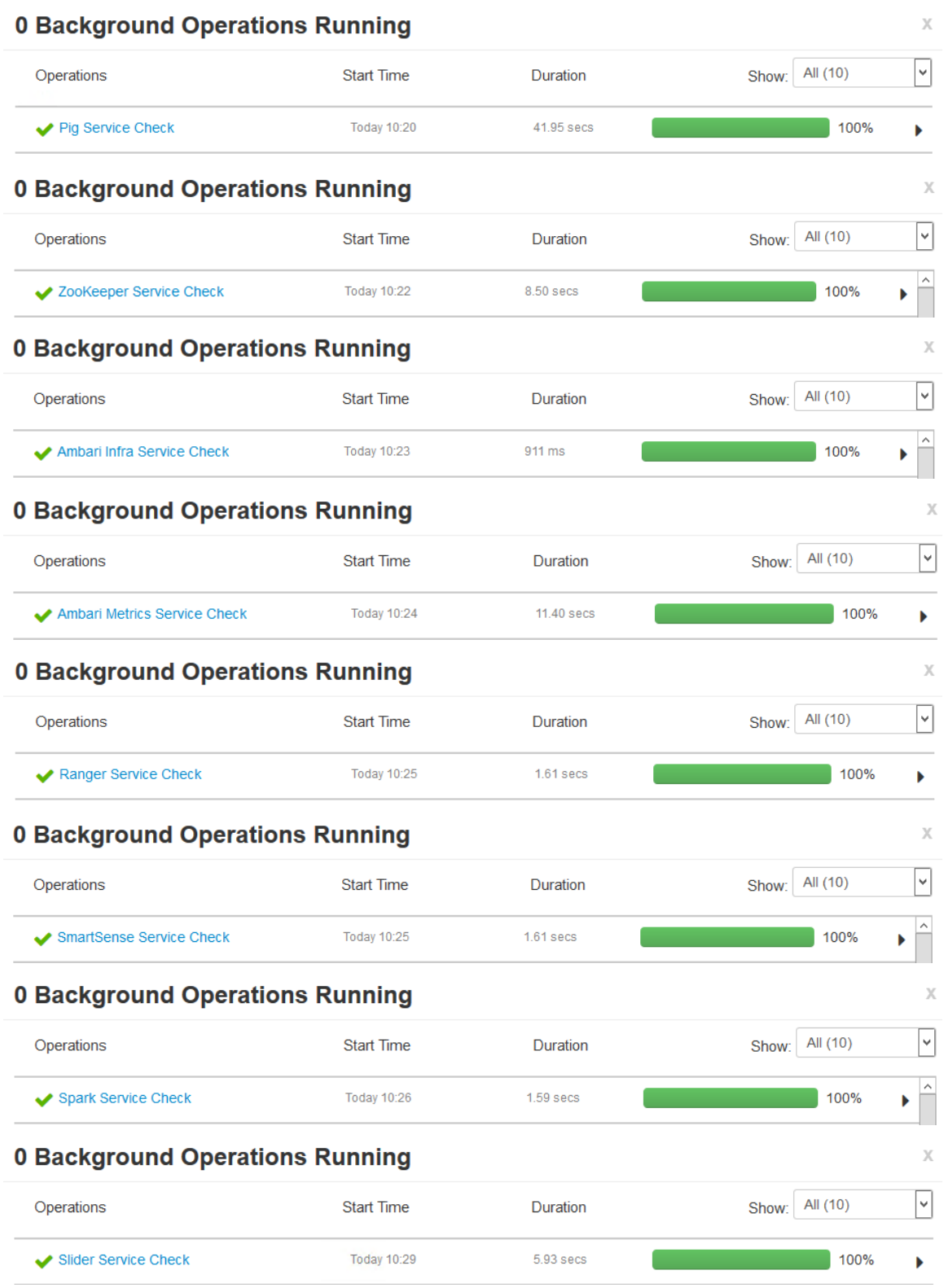
Ambari Server screenshots: Hadoop/Isilon

Home page

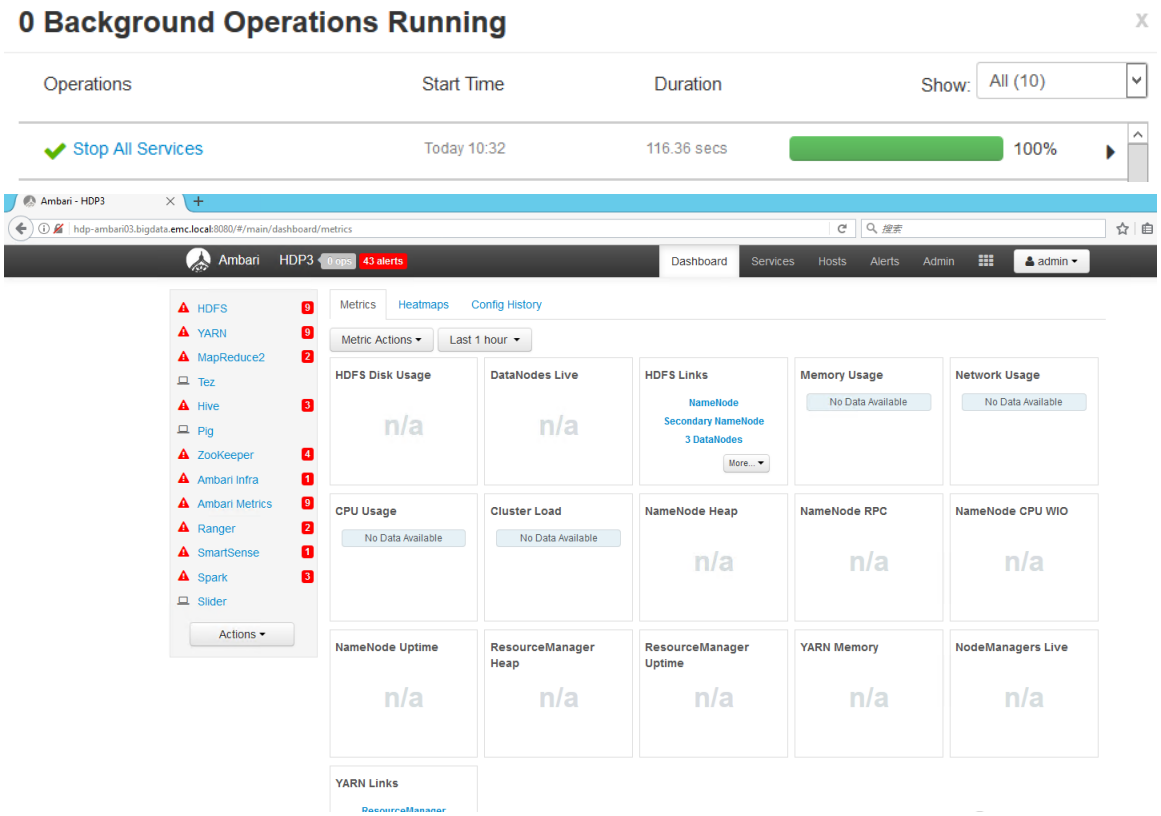


Run Service Check

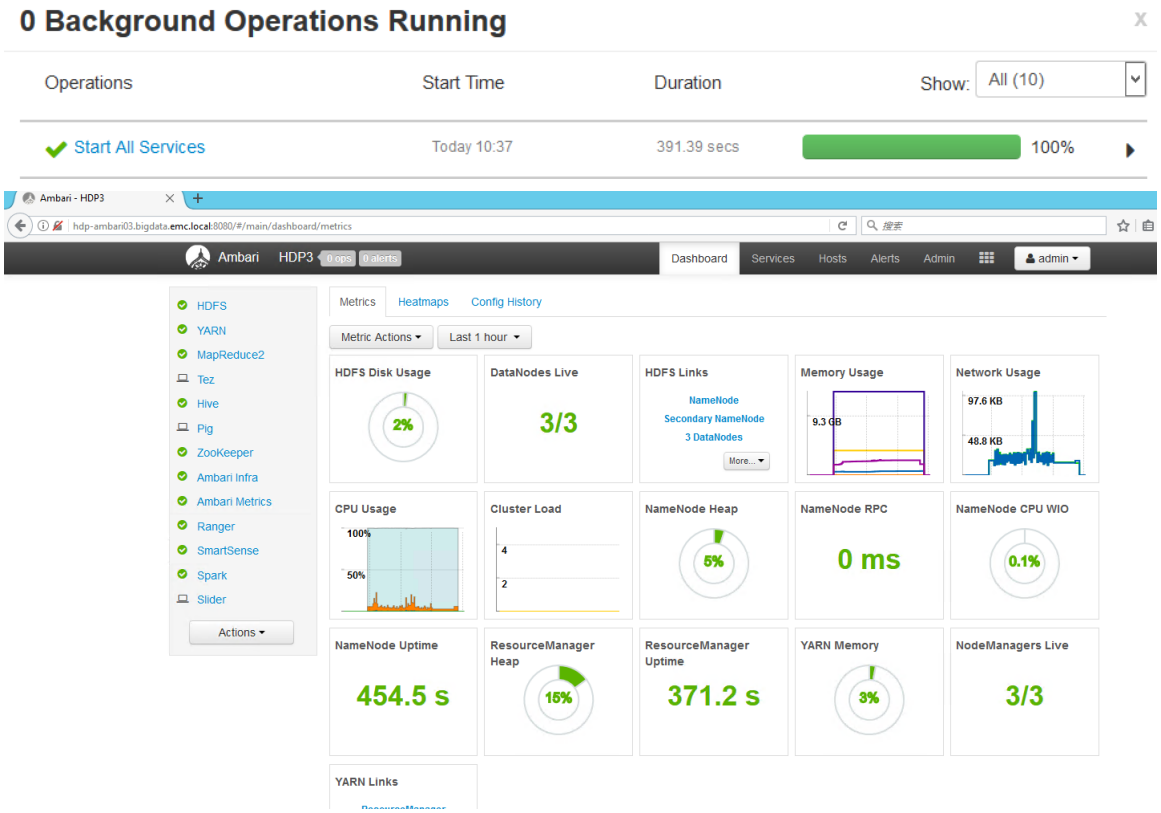



0 Background Operations Running
Operations
Start Time
Duration
Show: All (10)
Pig Service Check
Today 10:20
41.95 secs
100%
0 Background Operations Running
Operations
Start Time
Duration
Show: All (10)
ZooKeeper Service Check
Today 10:22
8.50 secs
100%
0 Background Operations Running
Operations
Start Time
Duration
Show: All (10)
Ambari Infra Service Check
Today 10:23
911 ms
100%
0 Background Operations Running
Operations
Start Time
Duration
Show: All (10)
Ambari Metrics Service Check
Today 10:24
11.40 secs
100%
0 Background Operations Running
Operations
Start Time
Duration
Show: All (10)
Ranger Service Check
Today 10:25
1.61 secs
100%
0 Background Operations Running
Operations
Start Time
Duration
Show: All (10)
SmartSense Service Check
Today 10:25
1.61 secs
100%
0 Background Operations Running
Operations
Start Time
Duration
Show: All (10)
Spark Service Check
Today 10:26
1.59 secs
100%
0 Background Operations Running
Operations
Start Time
Duration
Show: All (10)
Slider Service Check
Today 10:29
5.93 secs
100%

Stop All Services

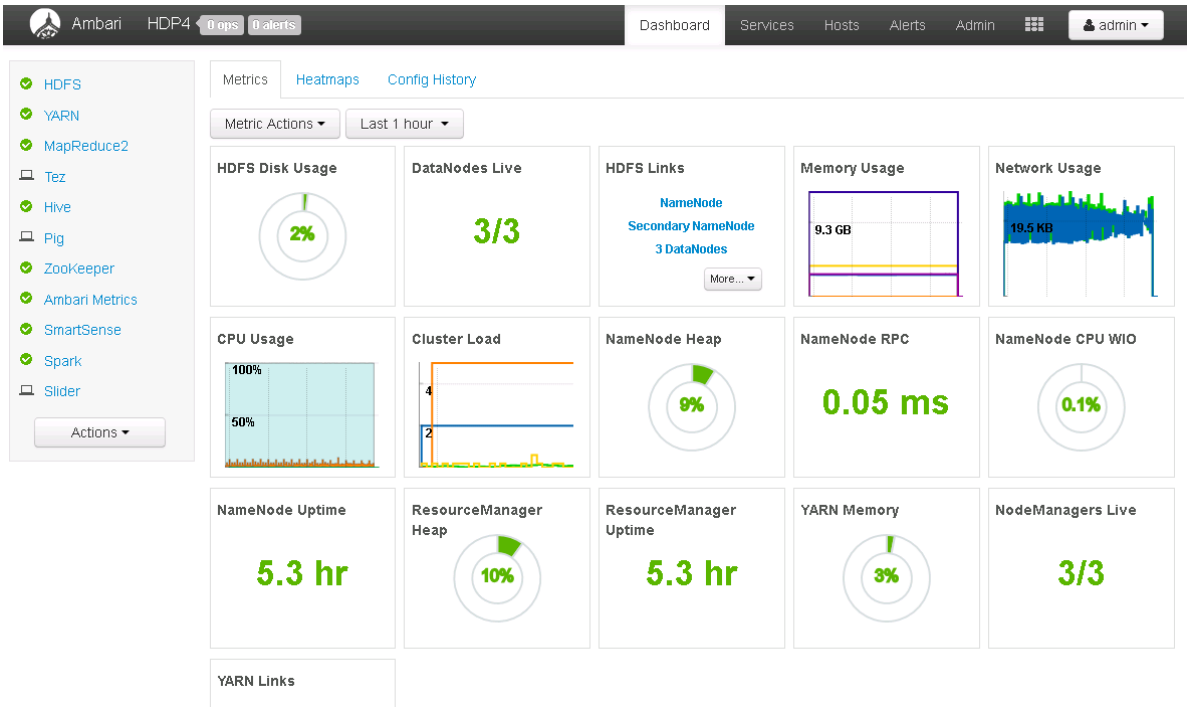


Start All Services

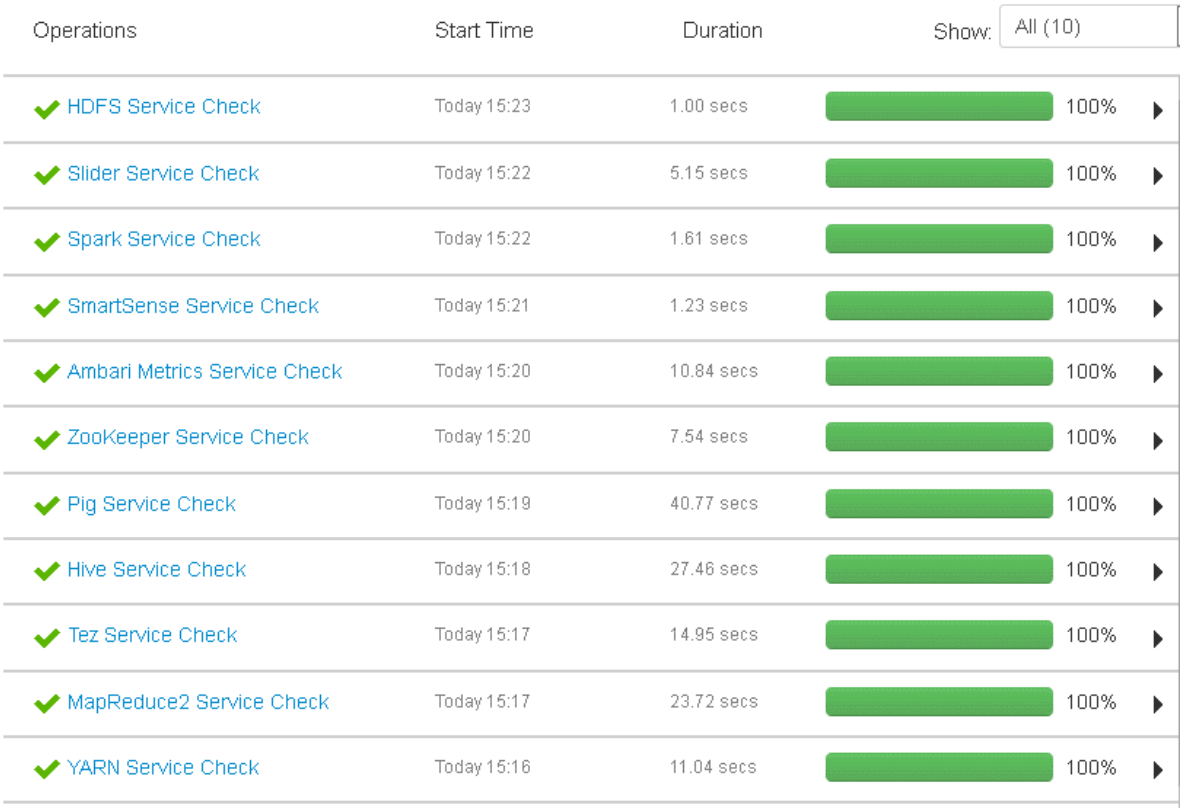


Ambari Server screenshots: Hadoop/ECS

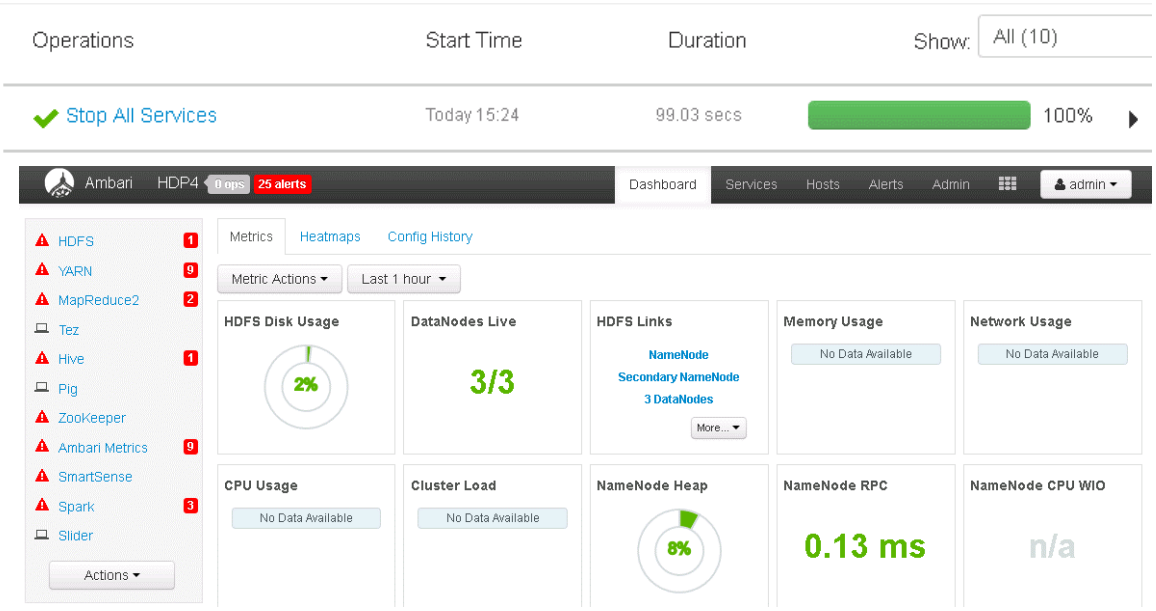
Home page



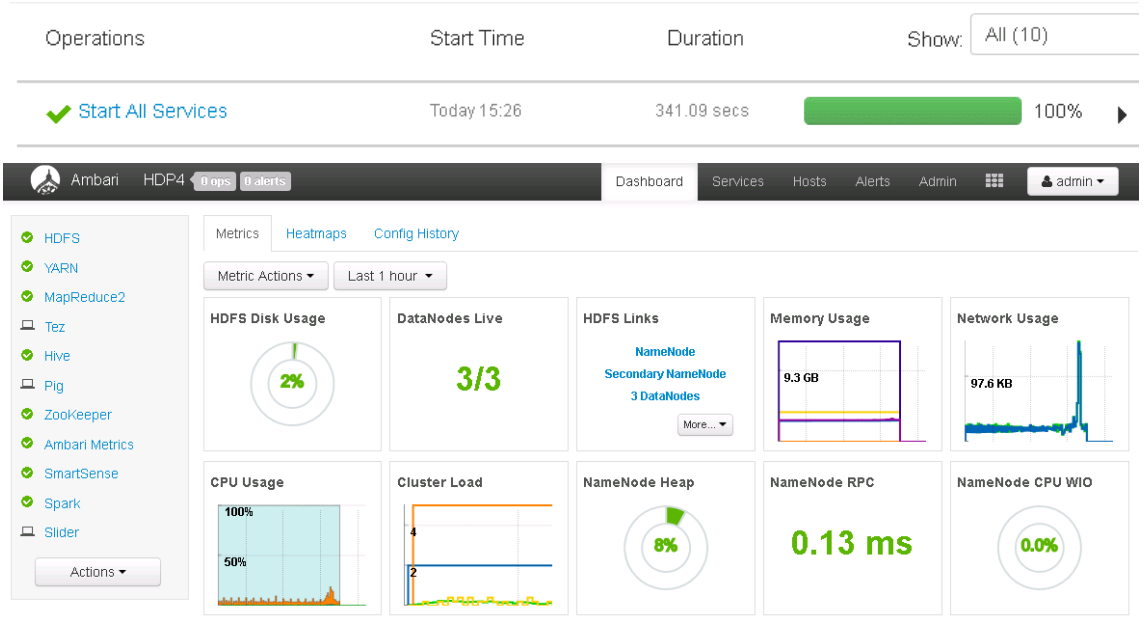
Run Service Check



Stop All Services



Start All Services



Appendix B Hadoop/Isilon Tests

This appendix presents the following topics:

Ambari GUI smoke testing

Test case name	Step	Description
Validation of install/ configuration	1	Set up 5-node HDP cluster as primary Hadoop cluster.
	2	Set up 3-node Isilon cluster as secondary Hadoop cluster.
	3	Create two access zones: HDFS1, HDFS2.
Usability and functionality test of GUI	1	Log in to Ambari Server web UI with admin privileges.
	2	Click **Run Service Check** from **HDFS** > **Service Action**.
	3	Click **Run Service Check** from **YARN** > **Service Action**.
	4	Click **Run Service Check** from **MapReduce2** > **Service Action**.
	5	Click **Run Service Check** from **Tez** > **Service Action**.
	6	Click **Run Service Check** from **Hive** > **Service Action**.
	7	Click **Run Service Check** from **Pig** > **Service Action**.
	8	Click **Run Service Check** from **Zookeeper** > **Service Action**.
	9	Click **Run Service Check** from **Ambari Infra** > **Service Action**.
	10	Click **Run Service Check** from **Ambari Metrics** > **Service Action**.
	11	Click **Run Service Check** from **Ranger** > **Service Action**.
	12	Click **Run Service Check** from **SmartSense** > **Service Action**.
	13	Click **Run Service Check** from **Spark** > **Service Action**.
	14	Click **Run Service Check** from **Slider** > **Service Action**.
	15	Click **Stop All** from home page > **Actions**.
	16	Change some configurations and restart related services.
	17	Add Ambari server node into the cluster.

MapReduce testing without Kerberos

Test case name	Step	Description
Word count	1	Put local file `/etc/redhat-release` on primary HDFS.
	2	Put local file `/etc/redhat-release` on secondary HDFS on Isilon access zone HDFS1.
	3	Put local file `/etc/redhat-release` on secondary HDFS on Isilon access zone HDFS2.
	4	Run MapReduce WordCount job on input from primary HDFS with output to Isilon HDFS1.
	5	Run MapReduce WordCount job on input from primary HDFS with output to Isilon HDFS2.

Test case name	Step	Description
	6	Run MapReduce WordCount job on input from Isilon HDFS1 with output to primary HDFS.
	7	Run MapReduce WordCount job on input from Isilon HDFS2 with output to primary HDFS.
	8	Run MapReduce WordCount job on input from Isilon HDFS1 with output to Isilon HDFS2.
	9	Run MapReduce WordCount job on input from Isilon HDFS2 with output to Isilon HDFS1.

Spark testing without Kerberos

Test case name	Step	Description
Line count and word count	1	Put local file `/etc/passwd` on primary HDFS.
	3	Put local file `/etc/passwd` on secondary HDFS on Isilon access zone HDFS1.
	5	Put local file `/etc/passwd` on secondary HDFS on Isilon access zone HDFS2.
	6	Run Spark LineCount/WordCount job on input from primary HDFS with output to Isilon HDFS1.
	7	Run Spark LineCount/WordCount job on input from primary HDFS with output to Isilon HDFS2.
	8	Run Spark LineCount/WordCount job on input from Isilon HDFS1 with output to primary HDFS.
	9	Run Spark LineCount/WordCount job on input from Isilon HDFS2 with output to primary HDFS.
	10	Run Spark LineCount/WordCount job on input from Isilon HDFS1 with output to Isilon HDFS2.
	11	Run Spark LineCount/WordCount job on input from Isilon HDFS2 with output to Isilon HDFS1.

Hive-MapReduce/Tez testing without Kerberos

Test case name	Step	Description
DDL operations 1. LOAD data local inpath 2. INSERT into table	1	Drop remote database if `EXISTS` cascade.
	2	Create remote_db with Hive warehouse on remote Isilon tier HDFS access zone.
	3	Create internal nonpartitioned table on remote_db.
	4	`LOAD data local inpath` into table created in preceding step.

Test case name	Step	Description
3. INSERT Overwrite TABLE	5	Create internal nonpartitioned table on remote Isilon HDFS access zone.
	6	`LOAD data local inpath` into table created in preceding step.
	7	Create internal transactional table on remote_db.
	8	`INSERT` into table from internal nonpartitioned table.
	9	Create internal partitioned table on remote_db.
	10	`INSERT OVERWRITE TABLE` from internal nonpartitioned table.
	11	Create external nonpartitioned table on remote_db.
	12	Drop local database if `EXISTS` cascade.
	13	Create local_db with, Hive warehouse on local DAS Hadoop cluster.
	14	Create internal nonpartitioned table on local_db.
	15	`LOAD data local inpath` into table created in preceding step.
	16	Create internal nonpartitioned table on local DAS Hadoop cluster.
	17	`LOAD data local inpath` into table created in preceding step.
	18	Create internal transactional table on local_db.
	19	`INSERT` into table from internal nonpartitioned table.
	20	Create internal partitioned table on local_db.
	21	`INSERT OVERWRITE TABLE` from internal nonpartitioned table.
	22	Create external nonpartitioned table on local_db.
DML operations 1. Query local database tables 2. Query remote database tables	1	Query data from local external nonpartitioned table.
	2	Query data from local internal nonpartitioned table.
	3	Query data from local nonpartitioned remote data table.
	4	Query data from local internal partitioned table.
	5	Query data from local internal transactional table.
	6	Query data from remote external nonpartitioned table.
	7	Query data from remote internal nonpartitioned table.
	8	Query data from remote nonpartitioned remote data table.
	9	Query data from remote internal partitioned table.
	10	Query data from remote internal transactional table.
JOIN tables in local database	1	`JOIN` external nonpartitioned table with internal nonpartitioned table.
	2	`JOIN` internal nonpartitioned table with internal nonpartitioned remote table.

Test case name	Step	Description
	3	JOIN internal nonpartitioned remote table with internal partitioned table.
	4	JOIN internal partitioned table with internal transactional table.
	5	JOIN internal transactional table with external nonpartitioned table.
JOIN tables in remote database	1	JOIN external nonpartitioned table with internal nonpartitioned table.
	2	JOIN internal nonpartitioned table with internal nonpartitioned remote table.
	3	JOIN internal nonpartitioned remote table with internal partitioned table.
	4	JOIN internal partitioned table with internal transactional table.
	5	JOIN internal transactional table with external nonpartitioned table.
JOIN tables between local_db and remote_db	1	JOIN local_db external nonpartitioned table with remote_db internal nonpartitioned table.
	2	JOIN local_db internal nonpartitioned table with remote_db internal nonpartitioned remote table.
	3	JOIN local_db internal nonpartitioned remote table with remote_db internal partitioned table.
	4	JOIN local_db internal partitioned table with remote_db internal transactional table.
	5	JOIN local_db internal transactional table with remote_db external nonpartitioned table.
Local temporary table from remote_db table	1	Create temporary table on local_db AS select query from remote_db table.
	2	Query data from temporary table.
IMPORT and EXPORT operations	1	EXPORT local_db internal partitioned table to remote Isilon tier HDFS access zone location.
	2	List the directory/file created on remote Hadoop cluster by EXPORT operation.
	3	IMPORT table to create table in local_db from the EXPORT data from the preceding step on remote Isilon tier HDFS access zone cluster.
	4	List the directory/file created on local Hadoop cluster by IMPORT operation.
	5	Query data from local_db table created by IMPORT operation.
	6	EXPORT remote_db external table to the local DAS Hadoop cluster location.
	7	List the directory/file created on local Hadoop cluster by EXPORT operation.

Test case name	Step	Description
	8	`IMPORT` table to create table on remote_db from the `EXPORT` data from the preceding step on local DAS Hadoop cluster.
	9	List directory/file created on remote Isilon tier HDFS access zone by preceding `IMPORT` operation.
	10	Query data from remote_db table created by preceding `IMPORT` operation.
Table-level and column-level statistics	1	Run table-level statistics command on external nonpartitioned table.
	2	Run `DESCRIBE EXTENDED` to check the statistics of the nonpartitioned table.
	3	Run column-level statistics command on internal partitioned table.
	4	Run `DESCRIBE EXTENDED` command to check the statics of the partitioned table.

TPC-DS testing

Test case name	Step	Description
Prepare Hive-testbench	1	Download latest Hive-testbench from Hortonworks github repository.
	2	Run `tpcds-build.sh` to build TPC-DS data generator.
	3	Run `tpcds-setup` to set up testbench database and load the data into created tables.
Database on remote Hadoop cluster and load data	1	Create LLAP database on remote Isilon tier HDFS location.
	2	Create 24 tables in LLAP database required to run the Hive test benchmark queries.
	3	Check the Hadoop file system location for the 24 table directories created on the remote location.
TPC-DS benchmarking	1	Switch from default database to LLAP database.
	2	Run `query52.sql` script.
	3	Run `query55.sql` script.
	4	Run `query91.sql` script.
	5	Run `query42.sql` script.
	6	Run `query12.sql` script.
	7	Run `query73.sql` script.
	8	Run `query20.sql` script.
	9	Run `query3.sql` script.
	10	Run `query89.sql` script.
	11	Run `query48.sql` script.

Kerberos security testing

Test case name	Step	Description
Kerberos user setup and testing	1	Create user hdp-user1 on all the nodes of HDP (local DAS HDFS) cluster.
	2	Add hdp-user1 principal in the Kerberos KDC server and assign password.
	3	Create home directory and assign permission for hdp-user1 in local DAS HDFS and remote Isilon cluster.
	4	Switch to hdp-user1 in Hadoop client node and query data from local and remote Isilon HDFS cluster.
	5	Put local file `/etc/redhat-release` on HDP (local DAS HDFS) file system.
	6	Put local file `/etc/redhat-release` on Isilon HDFS.
	7	Run MapReduce WordCount job on input from HDP (local DAS HDFS), with output to Isilon HDFS.
	8	Run MapReduce WordCount job on input from Isilon HDFS1, with output to HDP (local DAS HDFS).
Non-Kerberos user setup and testing	1	Create user hdp-user2 in Hadoop client node.
	2	Switch to hdp-user2 in Hadoop client node and query data from local DAS HDFS and remote Isilon HDFS cluster.
	3	Create home directory and assign permission for hdp-user2 in local DAS and remote Isilon HDFS.
	4	Put local file `/etc/redhat-release` on HDP (local DAS HDFS) file system.
	5	Put local file `/etc/redhat-release` on Isilon HDFS.
	6	Run MapReduce WordCount job on input from HDP (local DAS HDFS), with output to Isilon HDFS.
	7	Run MapReduce WordCount job on input from Isilon HDFS, with output to HDP (local DAS HDFS).

Ranger policy testing

Test case name	Step	Description
Access and restriction policy	1	Create directory `GRANT_ACCESS` on remote HDFS (Isilon cluster).
	2	Create directory `RESTRICT_ACCESS` on remote HDFS (Isilon cluster).
	3	Create hdp-user1 on all the nodes of both the Hadoop cluster (HDP3 local HDFS) and Isilon cluster.
	4	Assign RWX access for the hdp-user1 on `GRANT_ACCESS` from Ranger UI under hdp3_hadoop Service Manager.
	5	Put local file `/etc/redhat-release` into `GRANT_ACCESS` folder.

Test case name	Step	Description
	6	Put local file `/etc/redhat-release` into `RESTRICT_ACCESS` folder.
	7	Assign only read/write access for hdp-user1 on `RESTRICT_ACCESS` folder from Ranger UI.
	8	Copy file from `GRANT_ACCESS` to `RESTRICT_ACCESS` folder.
	9	Assign only read access for hdp-user1 on `RESTRICT_ACCESS` folder from Ranger UI.
	10	Delete `GRANT_ACCESS` and `RESTRICT_ACCESS` folders.

Ranger policy with Kerberos security testing

Test case name	Step	Description
MapReduce (word count)	1	Create `hdp-user1` home directory on HDP3 (local HDFS) and Isilon cluster.
	2	Assign RWX on `/user/hdp-user1` directory for hdp-user1 on HDP3 (local HDFS) and Isilon cluster using Ranger UI.
	3	Put local file `/etc/redhat-release` on HDP3 (local HDFS) file system.
	4	Put local file `/etc/redhat-release` on Isilon HDFS.
	5	Run MapReduce WordCount job on input from HDP3 (local HDFS), with output to Isilon HDFS.
	6	Run MapReduce WordCount job on input from Isilon HDFS1, with output to HDP3 (local HDFS).
Spark (line count and word count)	1	Put local file `/etc/passwd` on HDP3 (local HDFS) file system.
	2	Put local file `/etc/passwd` on Isilon HDFS.
	3	Run Spark LineCount/WordCount job on input from primary HDP3 (local HDFS), with output to secondary Isilon HDFS.
	4	Run Spark LineCount/WordCount job on input from secondary Isilon HDFS, with output to primary HDP3 (local HDFS) HDFS.

Ranger policy with Kerberos security testing on Hive warehouse

Test case name	Step	Description
Hive data warehouse Ranger policy setup	1	Assign RWX on `/user/hive` directory for hdp-user1 on HDP (local DAS HDFS) and Isilon cluster using Ranger UI.
DDL operations 1. LOAD data local inpath 2. INSERT into table 3. INSERT Overwrite TABLE	1	Drop remote database if `EXISTS` cascade.
	2	Create remote_db with hive warehouse on remote Isilon HDFS.
	3	Create internal nonpartitioned table on remote_db.
	4	`LOAD data local inpath` into table created in preceding step.
	5	Create internal nonpartitioned table on remote Isilon HDFS.
	6	`LOAD data local inpath` into table created in preceding step.
	7	Create internal transactional table on remote_db.
	8	`INSERT` into table from internal nonpartitioned table.
	9	Create internal partitioned table on remote_db.
	10	`INSERT OVERWRITE TABLE` from internal nonpartitioned table.
	11	Create external nonpartitioned table on remote_db.
	12	Drop local database if `EXISTS` cascade.
	13	Create local_db with hive warehouse on local DAS Hadoop cluster.
	14	Create internal nonpartitioned table on local_db.
	15	`LOAD data local inpath` into table created in preceding step.
	16	Create internal nonpartitioned table on local DAS Hadoop cluster.
	17	`LOAD data local inpath` into table created in preceding step.
	18	Create internal transactional table on local_db.
	19	`INSERT` into table from internal nonpartitioned table.
	20	Create internal partitioned table on local_db.
	21	`INSERT OVERWRITE TABLE` from internal nonpartitioned table.
	22	Create external nonpartitioned table on local_db.
DML operations 1. Query local database tables 2. Query remote database tables	1	Query data from local external nonpartitioned table.
	2	Query data from local internal nonpartitioned table.
	3	Query data from local nonpartitioned remote data table.
	4	Query data from local internal partitioned table.
	5	Query data from local internal transactional table.
	6	Query data from remote external nonpartitioned table.
	7	Query data from remote internal nonpartitioned table.

Test case name	Step	Description
	8	Query data from remote nonpartitioned remote data table.
	9	Query data from remote internal partitioned table.
	10	Query data from remote internal transactional table.
JOIN tables in local database	1	`JOIN` external nonpartitioned table with internal nonpartitioned table.
	2	`JOIN` internal nonpartitioned table with internal nonpartitioned remote table.
	3	`JOIN` internal nonpartitioned remote table with internal partitioned table.
	4	`JOIN` internal partitioned table with internal transactional table.
	5	`JOIN` internal transactional table with external nonpartitioned table.
JOIN tables in remote database	1	`JOIN` external nonpartitioned table with internal nonpartitioned table.
	2	`JOIN` internal nonpartitioned table with internal nonpartitioned remote table.
	3	`JOIN` internal nonpartitioned remote table with internal partitioned table.
	4	`JOIN` internal partitioned table with internal transactional table.
	5	`JOIN` internal transactional table with external nonpartitioned table.
JOIN tables between local_db and remote_db	1	`JOIN` local_db external nonpartitioned table with remote_db internal nonpartitioned table.
	2	`JOIN` local_db internal nonpartitioned table with remote_db internal nonpartitioned remote table.
	3	`JOIN` local_db internal nonpartitioned remote table with remote_db internal partitioned table.
	4	`JOIN` local_db internal partitioned table with remote_db internal transactional table.
	5	`JOIN` local_db internal transactional table with remote_db external nonpartitioned table.
Local temporary table from remote_db table	1	Create temporary table on local_db AS select query from remote_db table.
	2	Query data from temporary table.
IMPORT and EXPORT operations	1	`EXPORT` local_db internal partitioned table to remote Isilon Hadoop cluster location.
	2	List the directory/file created on remote Hadoop cluster by `EXPORT` operation.
	3	`IMPORT` table to create table in local_db from the `EXPORT` data on remote Hadoop cluster.
	4	List the directory/file created on local Hadoop cluster by `IMPORT` operation.
	5	Query data from local_db table created by `IMPORT` operation.

Test case name	Step	Description
	6	`EXPORT` remote_db external table to local HDP (local DAS HDFS) Hadoop cluster location.
	7	List the directory/file created on local Hadoop cluster by `EXPORT` operation.
	8	`IMPORT` table to create table on remote_db from the `EXPORT` data on local Hadoop cluster.
	9	List directory/file created on remote Isilon Hadoop cluster by `IMPORT` operation.
	10	Query data from remote_db table created by `IMPORT` operation.
Table-level and column-level statistics	1	Run table-level statistics command on external nonpartitioned table.
	2	Run `DESCRIBE EXTENDED` to check the statics of the nonpartitioned table.
	3	Run column-level statistics command on internal partitioned table.
	4	Run `DESCRIBE EXTENDED` to check the statics of the partitioned table.

DistCp in Kerberized and non-Kerberized cluster

Test case name	Step	Description
DistCp and script	1	Use DistCp to copy sample file from local DAS HDFS to remote Isilon HDFS cluster.
	2	Use DistCp to copy sample file from remote Isilon HDFS to local DAS HDFS cluster.

Appendix C Hadoop/ECS Tests

This appendix presents the following topics:

Ambari GUI smoke testing

Test case name	Step	Description
Validation of install/ configuration	1	Set up 5-node HDP cluster as primary Hadoop cluster.
	2	Set up 4-node ECS cluster as secondary Hadoop cluster.
	3	Create two buckets: HDFS1 and HDFS2.
Usability and functionality test of GUI	1	Log in to Ambari Server web UI with admin credentials.
	2	Click **Run Service Check** from **HDFS** > **Service Action**.
	3	Click **Run Service Check** from **YARN** > **Service Action**.
	4	Click **Run Service Check** from **MapReduce2** > **Service Action**.
	5	Click **Run Service Check** from **Tez** > **Service Action**.
	6	Click **Run Service Check** from **Hive** > **Service Action**.
	7	Click **Run Service Check** from **Pig** > **Service Action**.
	8	Click **Run Service Check** from **Zookeeper** > **Service Action**.
	9	Click **Run Service Check** from **Ambari Infra** > **Service Action**.
	10	Click **Run Service Check** from **Ambari Metrics** > **Service Action**.
	11	Click **Run Service Check** from **SmartSense** > **Service Action**.
	12	Click **Run Service Check** from **Spark** > **Service Action**.
	13	Click **Run Service Check** from **Slider** > **Service Action**.
	14	Click **Stop All** from home page > **Actions**.
	15	Change some configurations and restart related services.
	16	Add Ambari server node into the cluster.

MapReduce testing without Kerberos

Test case name	Step	Description
Word count	1	Put local file `/etc/redhat-release` on primary HDFS.
	2	Put local file `/etc/redhat-release` on secondary HDFS on ECS (bucket HDFS1).
	3	Put local file `/etc/redhat-release` on secondary HDFS on ECS (bucket HDFS2).
	4	Run MapReduce WordCount job on input from primary HDFS, with output to ECS HDFS1.
	5	Run MapReduce WordCount job on input from primary HDFS, with output to ECS HDFS2.
	6	Run MapReduce WordCount job on input from ECS HDFS1, with output to primary HDFS.

Test case name	Step	Description
	7	Run MapReduce WordCount job on input from ECS HDFS2, with output to primary HDFS.
	8	Run MapReduce WordCount job on input from ECS HDFS1, with output to ECS HDFS2.
	9	Run MapReduce WordCount job on input from ECS HDFS2, with output to ECS HDFS1.

Spark testing without Kerberos

Test case name	Step	Description
Line count and word count	1	Put local file `/etc/passwd` on primary HDFS.
	3	Put local file `/etc/passwd` on secondary HDFS on ECS (bucket HDFS1).
	5	Put local file `/etc/passwd` on secondary HDFS on ECS (bucket HDFS2).
	6	Run Spark LineCount/WordCount job on input from primary HDFS, with output to ECS HDFS1.
	7	Run Spark LineCount/WordCount job on input from primary HDFS, with output to ECS HDFS2.
	8	Run Spark LineCount/WordCount job on input from ECS HDFS1, with output to primary HDFS.
	9	Run Spark LineCount/WordCount job on input from ECS HDFS2, with output to primary HDFS.
	10	Run Spark LineCount/WordCount job on input from ECS HDFS1, with output to ECS HDFS2.
	11	Run Spark LineCount/WordCount job on input from ECS HDFS2, with output to ECS HDFS1.

Hive-MapReduce/Tez testing without Kerberos

Test case name	Step	Description
DDL operations 1. LOAD data local inpath 2. INSERT into table 3. INSERT Overwrite TABLE	1	Drop remote database if `EXISTS` cascade.
	2	Create remote_db with hive warehouse on remote ECS tier HDFS bucket.
	3	Create internal nonpartitioned table on remote_db.
	4	`LOAD data local inpath` into table created in preceding step.
	5	Create internal nonpartitioned table on remote ECS tier HDFS bucket.
	6	`LOAD data local inpath` into table created in preceding step.

Test case name	Step	Description
	7	Create internal transactional table on remote_db.
	8	`INSERT` into table from internal nonpartitioned table.
	9	Create internal partitioned table on remote_db.
	10	`INSERT OVERWRITE TABLE` from internal nonpartitioned table.
	11	Create external nonpartitioned table on remote_db.
	12	Drop local database if `EXISTS` cascade.
	13	Create local_db, hive warehouse resides on local DAS Hadoop cluster.
	14	Create internal nonpartitioned table on local_db.
	15	`LOAD data local inpath` into table created in preceding step.
	16	Create internal nonpartitioned table on local DAS Hadoop cluster.
	17	`LOAD data local inpath` into table created in preceding step.
	18	Create internal transactional table on local_db.
	19	`INSERT` into table from internal nonpartitioned table.
	20	Create internal partitioned table on local_db.
	21	`INSERT OVERWRITE TABLE` from internal nonpartitioned table.
	22	Create external nonpartitioned table on local_db.
DML operations 1. Query local database tables 2. Query remote database tables	1	Query data from local external nonpartitioned table.
	2	Query data from local internal nonpartitioned table.
	3	Query data from local nonpartitioned remote data table.
	4	Query data from local internal partitioned table.
	5	Query data from local internal transactional table.
	6	Query data from remote external nonpartitioned table.
	7	Query data from remote internal nonpartitioned table.
	8	Query data from remote nonpartitioned remote data table.
	9	Query data from remote internal partitioned table.
	10	Query data from remote internal transactional table.
JOIN tables in local database	1	`JOIN` external nonpartitioned table with internal nonpartitioned table.
	2	`JOIN` internal nonpartitioned table with internal nonpartitioned remote table.
	3	`JOIN` internal nonpartitioned remote table with internal partitioned table.
	4	`JOIN` internal partitioned table with internal transactional table.
	5	`JOIN` internal transactional table with external nonpartitioned table.

Test case name	Step	Description
JOIN tables in remote database	1	`JOIN` external nonpartitioned table with internal nonpartitioned table.
	2	`JOIN` internal nonpartitioned table with internal nonpartitioned remote table.
	3	`JOIN` internal nonpartitioned remote table with internal partitioned table.
	4	`JOIN` internal partitioned table with internal transactional table.
	5	`JOIN` internal transactional table with external nonpartitioned table.
JOIN tables between local_db and remote_db	1	`JOIN` local_db external nonpartitioned table with remote_db internal no partitioned table.
	2	`JOIN` local_db internal nonpartitioned table with remote_db internal nonpartitioned remote table.
	3	`JOIN` local_db internal nonpartitioned remote table with remote_db internal partitioned table.
	4	`JOIN` local_db internal partitioned table with remote_db internal transactional table.
	5	`JOIN` local_db internal transactional table with remote_db external nonpartitioned table.
Local temporary table from remote_db table	1	Create temporary table on local_db AS select query from remote_db table.
	2	Query data from temporary table.
IMPORT and EXPORT operations	1	`EXPORT` local_db internal partitioned table to remote Isilon tier HDFS access zone location.
	2	List the directory/file created on remote Hadoop cluster by `EXPORT` operation.
	3	`IMPORT` table to create table in local_db from the `EXPORT` data on remote Isilon tier HDFS access zone cluster.
	4	List the directory/file created on local Hadoop cluster by `IMPORT` operation.
	5	Query data from local_db table created by `IMPORT` operation.
	6	`EXPORT` remote_db external table to the local DAS Hadoop cluster location.
	7	List the directory/file created on local Hadoop cluster by `EXPORT` operation.
	8	`IMPORT` table to create table on remote_db from the `EXPORT` data on local DAS Hadoop cluster.
	9	List directory/file created on remote Isilon tier HDFS access zone by preceding `IMPORT` operation.
	10	Query data from remote_db table created by `IMPORT` operation.

Test case name	Step	Description
Table-level and column-level statistics	1	Run table-level statistics command on external nonpartitioned table.
	2	Run `DESCRIBE EXTENDED` to check the statics of the nonpartitioned table.
	3	Run column-level statics command on internal partitioned table.
	4	Run `DESCRIBE EXTENDED` to check the statics of the partitioned table.

TPC-DS testing

Test case name	Step	Description
Prepare Hive-testbench	1	Download latest Hive-testbench from Hortonworks github repository.
	2	Run `tpcds-build.sh` to build TPC-DS data generator.
	3	Run `tpcds-setup` to set up testbench database and load the data into created tables.
Database on remote Hadoop cluster and load data	1	Create LLAP database on remote ECS tier HDFS location.
	2	Create 24 tables in LLAP database required to run Hive-testbench queries.
	3	Check the Hadoop file system location for the 24 table directories created on the remote location.
TPC-DS benchmarking	1	Switch from working database to LLAP database.
	2	Run `query52.sql` script.
	3	Run `query55.sql` script.
	4	Run `query91.sql` script.
	5	Run `query42.sql` script.
	6	Run `query12.sql` script.
	7	Run `query73.sql` script.
	8	Run `query20.sql` script.
	9	Run `query3.sql` script.
	10	Run `query89.sql` script.
	11	Run `query48.sql` script.

Kerberos security testing

Test case name	Step	Description
Kerberos user setup and testing	1	Create user hdp-user1 on all the nodes of HDP (local DAS HDFS) cluster.
	2	Add hdp-user1 principal in the Kerberos KDC server and assign password.
	3	Create home directory and assign permission for hdp-user1 in local DAS HDFS and remote ECS cluster.
	4	Switch to hdp-user1 in Hadoop client node and query data from local and remote ECS HDFS cluster.
	5	Put local file `/etc/redhat-release` on HDP (local DAS HDFS) file system.
	6	Put local file `/etc/redhat-release` on ECS HDFS.
	7	Run MapReduce WordCount job on input from HDP (local DAS HDFS), with output to Isilon HDFS.
	8	Run MapReduce WordCount job on input from ECS HDFS1, with output to HDP (local DAS HDFS).
Non-Kerberos user setup and testing	1	Create user hdp-user2 in Hadoop client node.
	2	Switch to hdp-user2 in Hadoop client node and query data from local DAS HDFS and remote ECS HDFS cluster.
	3	Create home directory and assign permission for hdp-user2 in local DAS and remote ECS HDFS.
	4	Put local file `/etc/redhat-release` on HDP (local DAS HDFS) file system.
	5	Put local file `/etc/redhat-release` on ECS HDFS.
	6	Run MapReduce WordCount job on input from HDP (local DAS HDFS), with output to ECS HDFS.
	7	Run MapReduce WordCount job on input from ECS HDFS, with output to HDP (local DAS HDFS).

MapReduce word count and Spark word count, line count on Kerberized cluster

Test case name	Step	Description
MapReduce (word count)	1	Create hdp-user1 home directory on HDP3 (local HDFS) and ECS.
	3	Put local file `/etc/redhat-release` on HDP3 (local HDFS) file system.
	4	Put local file `/etc/redhat-release` on ECS HDFS.
	5	Run MapReduce WordCount job on input from HDP3 (local HDFS), with output to ECS HDFS.
	6	Run MapReduce WordCount job on input from ECS HDFS1, with output to HDP3 (local HDFS).
Spark (line count and word count)	1	Put local file `/etc/passwd` on HDP3 (local HDFS) file system.
	2	Put local file `/etc/passwd` on ECS HDFS.
	3	Run Spark LineCount/WordCount job on input from primary HDP3 (local HDFS), with output to secondary ECS HDFS.
	4	Run Spark LineCount/WordCount job on input from secondary ECS HDFS, with output to primary HDP3 (local HDFS).

Kerberos security testing on Hive warehouse

Test case name	Step	Description
DDL operations 1. LOAD data local inpath 2. INSERT into table 3. INSERT Overwrite TABLE	1	Drop remote database if `EXISTS` cascade.
	2	Create remote_db, hive warehouse resides on remote ECS HDFS.
	3	Create internal nonpartitioned table on remote_db.
	4	`LOAD data local inpath` into table created in preceding step.
	5	Create internal nonpartitioned table on remote ECS HDFS.
	6	`LOAD data local inpath` into table created in preceding step.
	7	Create internal transactional table on remote_db.
	8	`INSERT` into table from internal nonpartitioned table.
	9	Create internal partitioned table on remote_db.
	10	`INSERT OVERWRITE TABLE` from internal nonpartitioned table.
	11	Create external nonpartitioned table on remote_db.
	12	Drop local database if `EXISTS` cascade.
	13	Create local_db, hive warehouse resides on local DAS Hadoop cluster.
	14	Create internal nonpartitioned table on local_db.

Test case name	Step	Description
	15	`LOAD data local inpath` into table created in preceding step.
	16	Create internal nonpartitioned table on local DAS Hadoop cluster.
	17	`LOAD data local inpath` into table created in preceding step.
	18	Create internal transactional table on local_db.
	19	`INSERT` into table from internal nonpartitioned table.
	20	Create internal partitioned table on local_db.
	21	`INSERT OVERWRITE TABLE` from internal nonpartitioned table.
	22	Create external nonpartitioned table on local_db.
DML operations 1. Query local database tables 2. Query remote database tables	1	Query data from local external nonpartitioned table.
	2	Query data from local internal nonpartitioned table.
	3	Query data from local nonpartitioned remote data table.
	4	Query data from local internal partitioned table.
	5	Query data from local internal transactional table.
	6	Query data from remote external nonpartitioned table.
	7	Query data from remote internal nonpartitioned table.
	8	Query data from remote nonpartitioned remote data table.
	9	Query data from remote internal partitioned table.
	10	Query data from remote internal transactional table.
JOIN tables in local database	1	`JOIN` external nonpartitioned table with internal nonpartitioned table.
	2	`JOIN` internal nonpartitioned table with internal nonpartitioned remote table
	3	`JOIN` internal nonpartitioned remote table with internal partitioned table.
	4	`JOIN` internal partitioned table with internal transactional table.
	5	`JOIN` internal transactional table with external nonpartitioned table.
JOIN tables in remote database	1	`JOIN` external nonpartitioned table with internal nonpartitioned table.
	2	`JOIN` internal nonpartitioned table with internal nonpartitioned remote table.
	3	`JOIN` internal nonpartitioned remote table with internal partitioned table.
	4	`JOIN` internal partitioned table with internal transactional table.
	5	`JOIN` internal transactional table with external nonpartitioned table.

Test case name	Step	Description
JOIN tables between local_db and remote_db	1	`JOIN` local_db external nonpartitioned table with remote_db internal nonpartitioned table.
	2	`JOIN` local_db internal nonpartitioned table with remote_db internal nonpartitioned remote table.
	3	`JOIN` local_db internal nonpartitioned remote table with remote_db internal partitioned table.
	4	`JOIN` local_db internal partitioned table with remote_db internal transactional table.
	5	`JOIN` local_db internal transactional table with remote_db external nonpartitioned table.
Local temporary table from remote_db table	1	Create temporary table on local_db AS select query from remote_db table.
	2	Query data from temporary table.
IMPORT and EXPORT operations	1	`EXPORT` local_db internal partitioned table to remote Isilon Hadoop cluster location.
	2	List the directory/file created on remote Hadoop cluster by `EXPORT` operation.
	3	`IMPORT` table to create table in local_db from the `EXPORT` data on remote Hadoop cluster.
	4	List the directory/file created on local Hadoop cluster by `IMPORT` operation.
	5	Query data from local_db table created by `IMPORT` operation.
	6	`EXPORT` remote_db external table to the local HDP (local DAS HDFS) Hadoop cluster location.
	7	List the directory/file created on local Hadoop cluster by `EXPORT` operation.
	8	`IMPORT` table to create table on remote_db from the `EXPORT` data on local Hadoop cluster.
	9	List directory/file created on remote ECS Hadoop cluster by preceding `IMPORT` operation.
	10	Query data from remote_db table created by `IMPORT` operation.
Table-level and column-level statistics	1	Run table-level statistics command on external nonpartitioned table.
	2	Run DESCRIBE EXTENDED to check the statics of the nonpartitioned table.
	3	Run column-level statistics command on internal partitioned table.
	4	Run DESCRIBE EXTENDED to check the statics of the partitioned table.

DistCp in Kerberized and non-Kerberized cluster

Test case name	Step	Description
DistCp and script	1	Use DistCp to copy sample file from local DAS HDFS to remote ECS HDFS cluster.
	2	Use DistCp to copy sample file from remote ECS HDFS to local DAS HDFS cluster.