Implementation Guide

Hadoop Tiered Storage with Dell EMC PowerScale and Cloudera CDP Private Cloud Base

Hadoop Tiered Storage implementation and validation guide

Abstract

This solution guide describes how to easily expand storage of existing traditional Hadoop® clusters with Dell EMC™ PowerScale™ to supply immediate capacity, better storage efficiency, and reduced total cost of ownership.

May 2021

H18805

Revisions

Date	Description
May 2021	Initial release

Acknowledgments

Author: Kirankumar Bhusanurmath, Analytics Solutions Architect, Dell EMC

Table of contents

Executive summary

Enterprises implementing digital transformation initiatives and data-driven decision-making must deal with exponential data growth that is not supplied for in their IT budgets. For most enterprises, most of this data growth is “cold data,” which is historical in nature and does not need frequent or low-latency access. The rest of the data growth is in “hot data,” which is recently generated data that needs 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 and low-latency cluster. With low cost per MB/s and cold data in a capacity-dense cluster with low cost per TB.

Disclaimer

1 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™ PowerScale™ 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 PowerScale 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 PowerScale 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 a PowerScale 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 PowerScale clusters.

1.1 Key results

Dell EMC and Cloudera have validated multiple configurations for Hadoop tiered storage with a logical Hadoop cluster (DAS storage) and an infrastructure cluster (PowerScale system) that meet or exceed the functional objectives of this solution. You can match most needs with an approved configuration. By combining the Cloudera Data Platform Private Cloud Base (CDP Private Cloud Base) cluster (logical Hadoop cluster) with the flexibility of a PowerScale infrastructure cluster, you can scale the solution to handle future requirements without extensive upgrades or expensive re-platforming.

2 Technical overview

2.1 Reference Architecture

Figure 1 shows the reference architecture of Hadoop tiered storage with a PowerScale. This reference architecture supports 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.

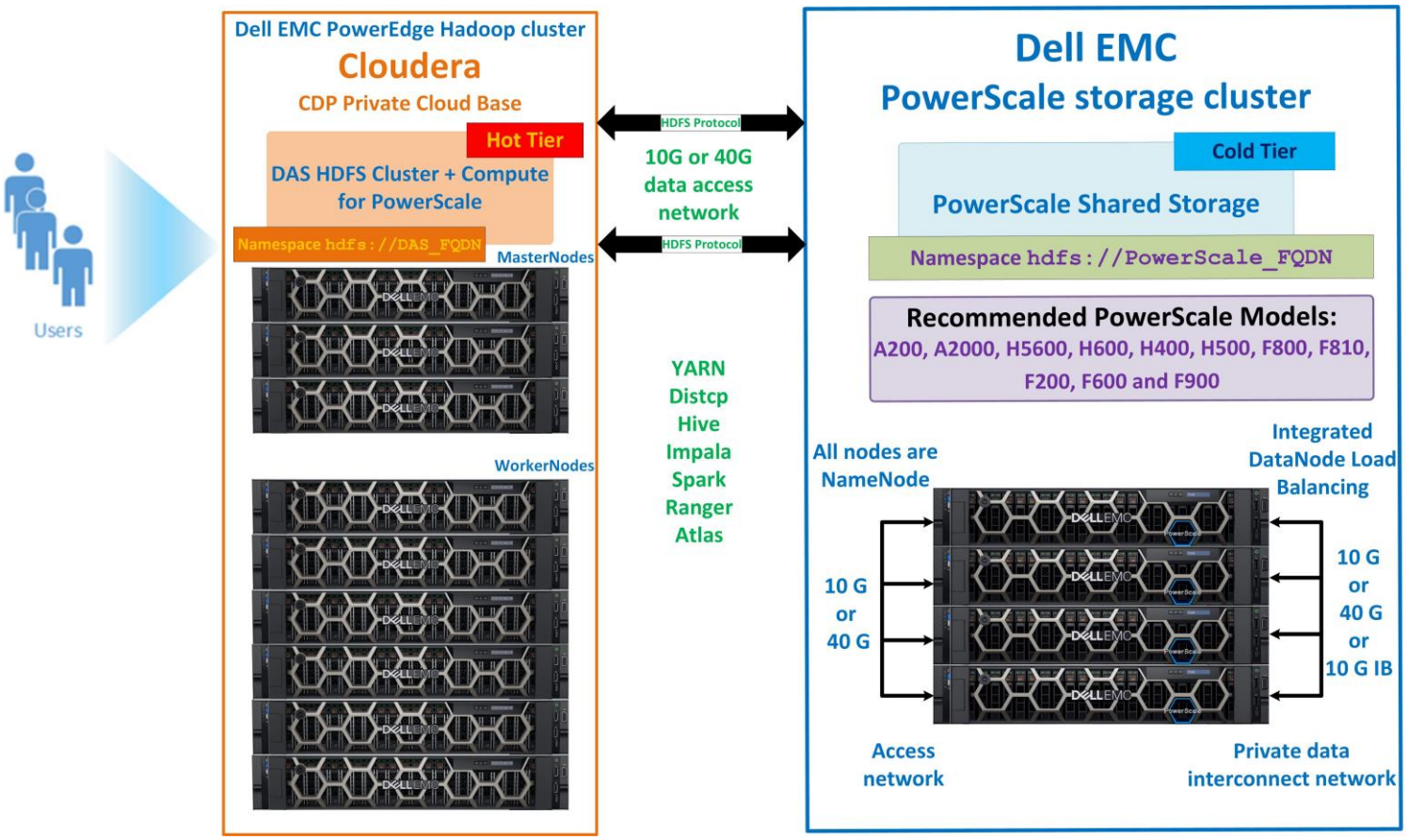


Figure 1 Reference architecture of Hadoop tiered storage with a PowerScale cluster

2.2 Key components

2.2.1 Dell EMC PowerScale

The Dell EMC PowerScale 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 PowerScale OneFS™ operating system, a PowerScale 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 a PowerScale 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 PowerScale 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.

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

2.2.2 Cloudera Data Platform (CDP)

Cloudera Data Platform (CDP) Private Cloud Base is the on-premises version of the Cloudera Data Platform. This new product combines the best of Cloudera Enterprise Data Hub and Hortonworks Data Platform Enterprise along with new features and enhancements across the stack. This unified distribution is a scalable and customizable platform where you can securely run many types of workloads.

CDP Private Cloud Base supports various hybrid solutions where compute tasks are separated from data storage and where data can be accessed from remote clusters. This hybrid approach provides a foundation for containerized applications by managing storage, table schema, authentication, authorization, and governance.

CDP Private Cloud Base consists of various components such as Apache HDFS, Apache Hive 3, Apache HBase, and Apache Impala, along with many other components for specialized workloads. You can select any combination of these services to create clusters that address your business requirements and workloads. Several preconfigured packages of services are also available for common workloads. These include:

- Data Engineering
  Ingest, transform, and analyze data.
  Services: HDFS, YARN, YARN Queue Manager, Ranger, Atlas, Hive metastore, Hive on Tez, Spark, Oozie, Hue, and Data Analytics Studio
- Data Mart
  Browse, query, and explore your data in an interactive way.
  Services: HDFS, YARN, YARN Queue Manager, Ranger, Atlas, Hive metastore, Impala, and Hue
- Operational Database
  Low latency writes, reads, and persistent access to data for Online Transactional Processing (OLTP) use cases.
  Services: HDFS, Ranger, Atlas, and HBase

When installing a CDP Data Center cluster, you install a single parcel called Cloudera Runtime that. contains all of the components. For a complete list of the included components, see Cloudera Runtime Component Versions.

In addition to the Cloudera Runtime components, the CDP Data Center includes powerful tools to manage, govern, and secure your cluster.

CDP - Data Center Component Documentation links to documentation for Cloudera Runtime components.

2.2.2.1 Cloudera Manager

Cloudera Manager is an end-to-end application for managing CDP clusters. Cloudera Manager provides granular visibility into and control over every part of the CDP cluster empowering operators to improve performance, enhance quality of service, increase compliance, and reduce administrative costs.

2.2.2.2 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 CDP. Kerberos provides secure and reliable authentication to multiple applications. PowerScale systems support the Kerberos authentication feature using Kerberos Key Distribution Center (KDC) services.

2.3 Software resources

Table 1 lists the solution software resources.

Table 1 Software resources

Software	Version
CentOS Linux 64-bit	CentOS Linux release 7.7.1908 (Core)
Cloudera Data Platform Private Cloud Base	Cloudera Manager 7.3.1: See download information here
	Cloudera Runtime 7.1.6: See download information here
MIT Kerberos	Kerberos 5 version 1.15.1
Dell EMC OneFS	Isilon OneFS 9.2.0.0 (Release, Build B_9_2_0_002(RELEASE), 2021-04-02 11:58:34, 0x902005000000002)

3 Solution design

3.1 Deployment best practices

3.1.1 Spreading data to a PowerScale cluster

Spread data to a PowerScale 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 a PowerScale cluster equals 60 percent of DAS.
- The rack space of a Hadoop cluster with a PowerScale cluster equals 40 percent of DAS.

For more than 1 PB of usable data:

- The acquisition cost of a Hadoop cluster with a PowerScale cluster could be more than 60 percent of DAS.

3.1.2 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.

3.1.3 ORCFile format for Hive tables

Hive supports ORCFile, a new table storage format that provides 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.

3.1.4 Directory structure considerations

You can use directory structures to organize data by department, business unit, life-cycle stage (new compare with old, hot compare with cold, raw compare with 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 file system; directories can represent your metadata.

Tools like Hive understand partition pruning during query execution. Each partition is 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 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 datasets 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, datasets 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 life cycle 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 datasets. Even partitions can be separately controlled in terms of access (on user type or line of business for datasets, for example).

Directory structure design is a complex topic. Dell EMC offers professional services to help with directory structure design and other Hadoop-related services.

3.2 Hadoop tiered storage with PowerScale cluster

3.2.1 Overview

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

Below figure shows the Hadoop with PowerScale solution architecture.

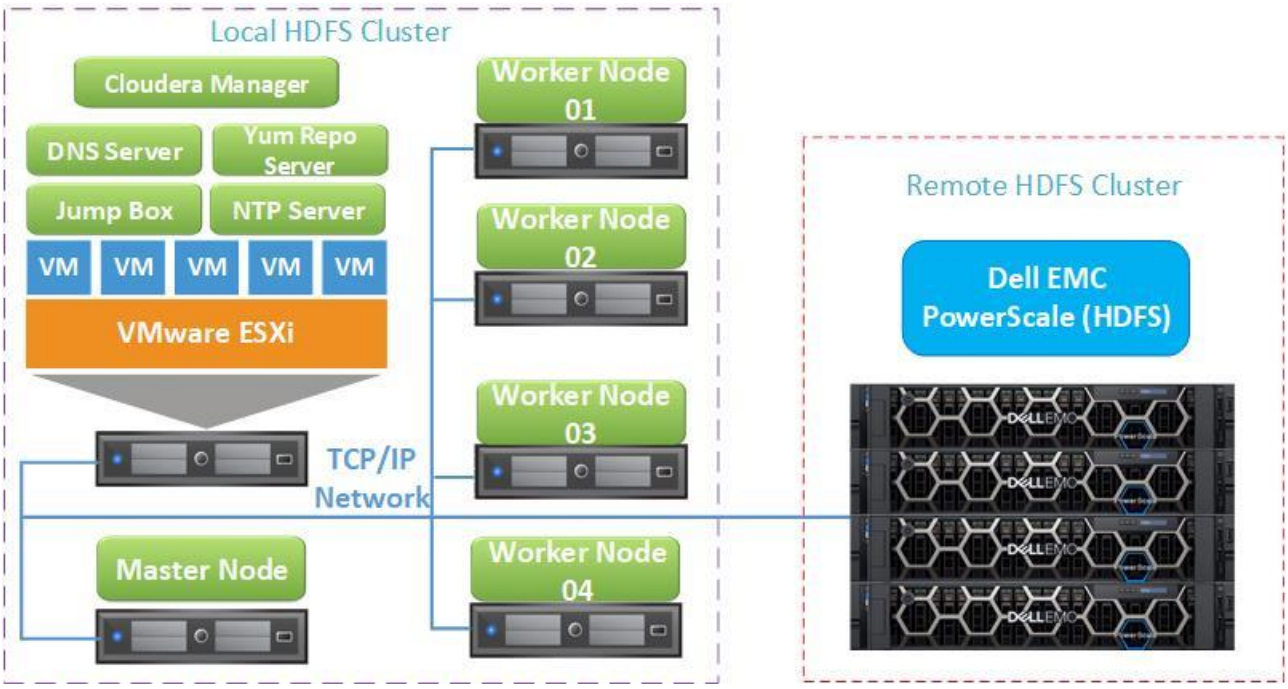


Figure 2 Hadoop tiered storage with PowerScale solution architecture

4 Hadoop cluster deployment and integration with PowerScale cluster

4.1 Overview

Below table lists the process flow for the Hadoop cluster deployment with a PowerScale cluster.

Step	Action
1	Set up the CDP Private Cloud Base cluster
2	Set up the PowerScale cluster
3	Create a PowerScale access zone
4	Enable Kerberos on the CDP Private Cloud Base cluster
5	Enable Kerberos on the PowerScale cluster
6	Validate CDP Private Cloud Base deployment and PowerScale integration

4.2 Setting up the CDP cluster

4.2.1 Installing Cloudera Manager

Official Cloudera manager install guide is referred to deploy CM version 7.3.1, please see here for detailed information.

4.2.2 Installing CDP Private Cloud Base

Official Cloudera CDP install guide is referred to deploy CDP Private Cloud Base 7.1.6 as an HDFS cluster, please see here for detailed information.

4.3 Setting up the PowerScale cluster

To set up the PowerScale cluster infrastructure, after setting up the CDP Private Cloud Base cluster, contact your Dell EMC or partner representative.

4.4 Creating PowerScale access zones

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

1. On a node in the PowerScale 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
```

4.4.1 Creating HDFS users and groups

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

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

Go to Tools for Using Hadoop with OneFS to set up the users and directories on the cluster. After the users are created on the PowerScale hdfs access zone, run the hdfs client user creation bash script created by the Isilon_create_users.sh on all the CDP hosts. Or manually adjust the UID/GID of the users and groups if CDP cluster is already deployed.

NOTE: It is mandatory to maintain UID/GID parity between PowerScale OneFS and CDP compute cluster hosts. Otherwise data access workload fails with the authentication error.

4.4.2 Creating and configuring the PowerScale HDFS root

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. Create a role for Hadoop access zone

```
isi auth roles create --name=HdfsAccess --description="Bypass FS permissions"
--zone=hdfs1
isi auth roles create --name=HdfsAccess --description="Bypass FS permissions"
--zone=hdfs2
```

4. Add restore privileges to the new “HdfsAccess” role

```
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
```

5. Add backup privileges to the new “HdfsAccess” role

```
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
```

6. Add user hdfs to the new “HdfsAccess” role.

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

4.5 Enabling Kerberos on the CDP cluster

Official Cloudera CDP Kerberos setup guide is referred to enable Kerberos on CDP 7.1.6, for more details please see here. Below section describes some extra configuration added in HDFS cluster to make PowerScale work from this CDP cluster.

4.5.1 Configurations required with Kerberos and PowerScale

Specify the following configurations in Cloudera Manager on the **Clusters** > ***HDFS Service*** > **Configuration** tab:

1. In **HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** `hdfs-site.xml` and the **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml** properties for the PowerScale service, set the value of the `dfs.client.file-block-storage-locations.timeout.millis` property to `10000`.
2. In the HDFS **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml** property for the PowerScale service, set the value of the `hadoop.security.token.service.use_ip` property to `FALSE`.
3. If you see errors that reference the `.Trash` directory, ensure that **Use Trash** property is selected.

Note: See Cloudera documentation here for more information about setting up these configurations for PowerScale on HDFS compute cluster.

4.6 Enabling Kerberos on the PowerScale cluster

Kerberize the PowerScale cluster and synchronize it to the CDH 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 a PowerScale OneFS cluster and specify MIT KDC as a PowerScale 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 PowerScale OneFS web administration interface, under **Data Protection** > **Authentication**, enable Kerberos and provide the required information, as shown in Figure 3 and Figure 4.

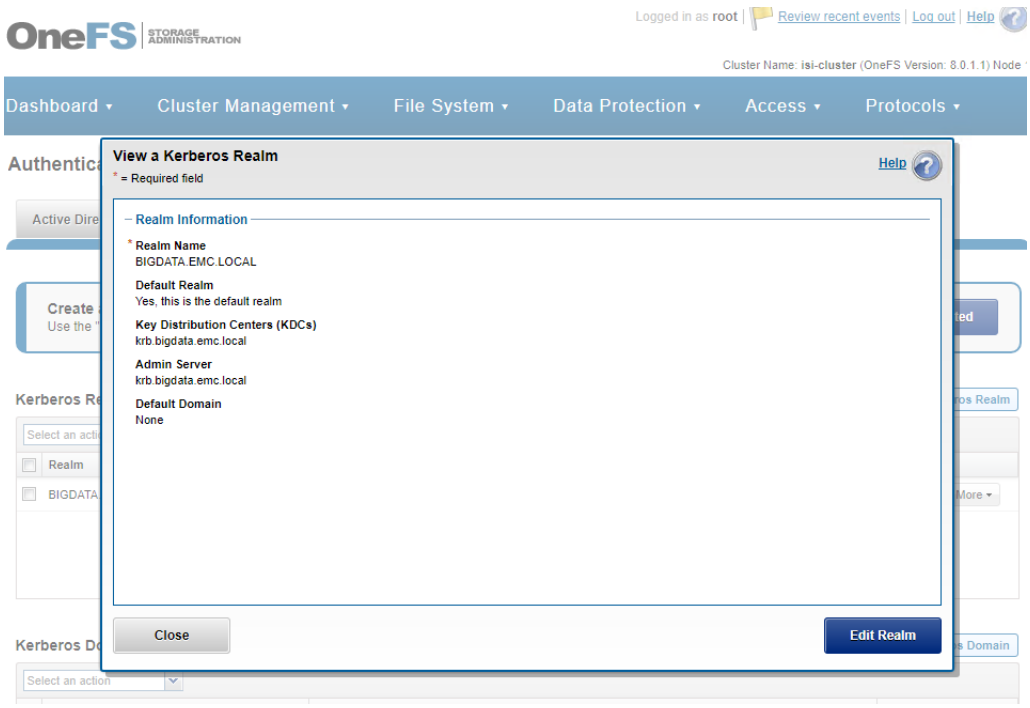


Figure 3 Enabling Kerberos authentication in the OneFS web administration interface

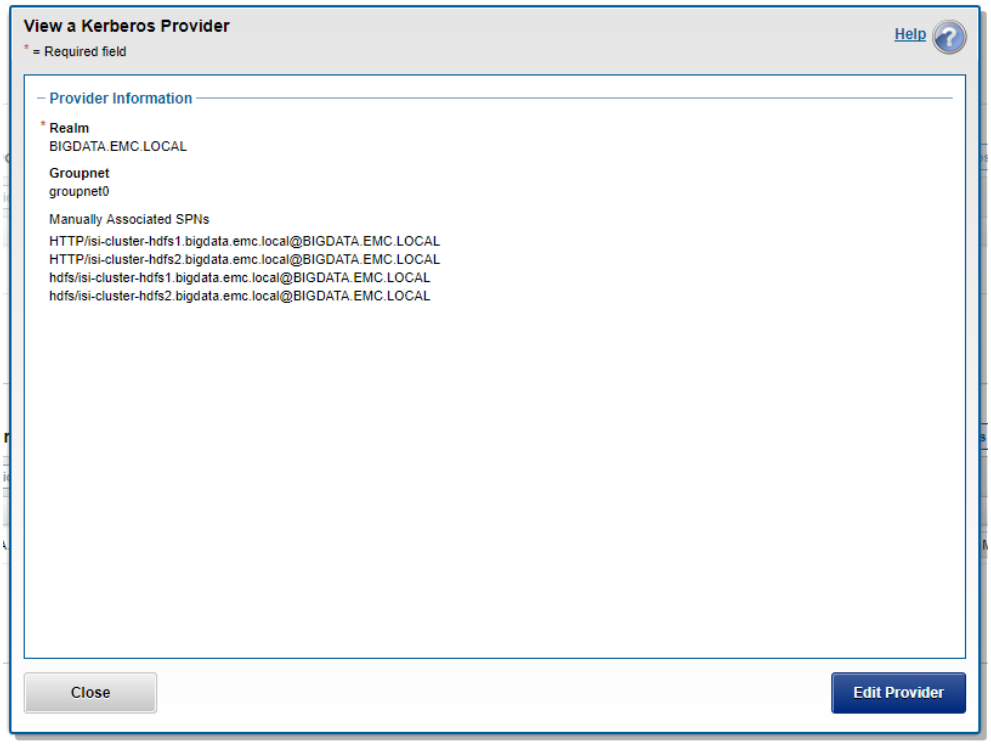


Figure 4 List of Kerberos providers in the OneFS web administration interface

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=$PowerScale_zone --authentication-
mode=kerberos_only
```

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

4.7 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.

1. Add Ranger service on HDP cluster and enable necessary plugins.
2. To enable Ranger on Isilon login into Isilon OneFS web UI
3. Under **Protocols ➔ Hadoop (HDFS) ➔ Ranger Plugin settings**

    a. Select the Current Access Zone
    b. Select Enable Ranger Plugin.
    c. In the **Policy manager URL** text box, type the URL for the policy manager.
    In the **Repository name** text box, type the repository name.

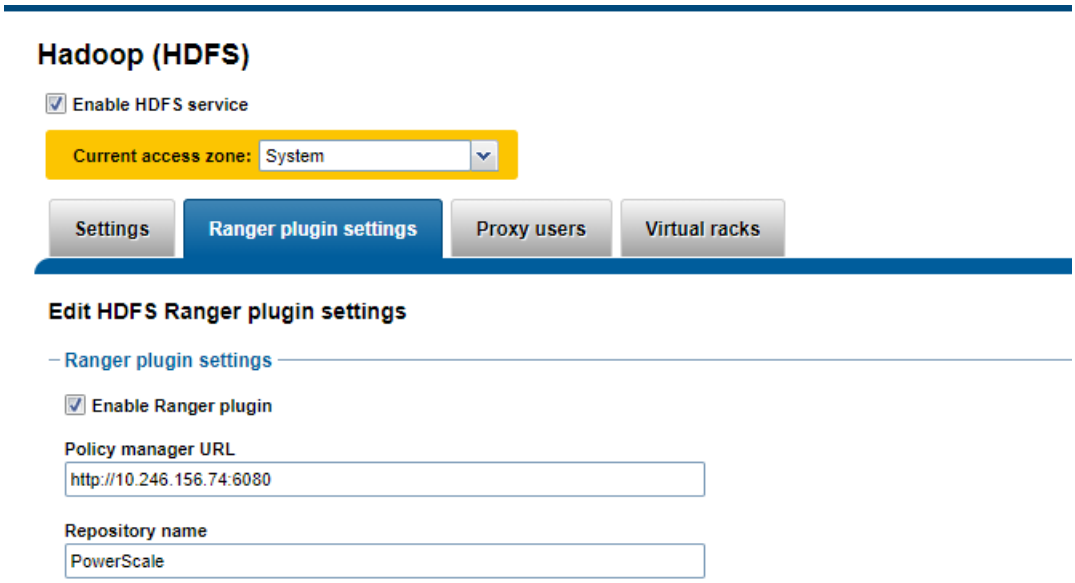


Figure 5 Ranger Plugin Settings tab

4. Enabling HDFS, YARN, HIVE, and so on, Ranger plug-in in the HDFS cluster automatically adds the HDFS, YARN, HIVE, and so on, service manager plugins in Ranger Admin UI.
5. Now PowerScale HDFS Ranger plug-in needs to be added manually in the Ranger Admin UI under HDFS service manager.
6. Create HDFS service for PowerScale HDFS as shown in the below figures.

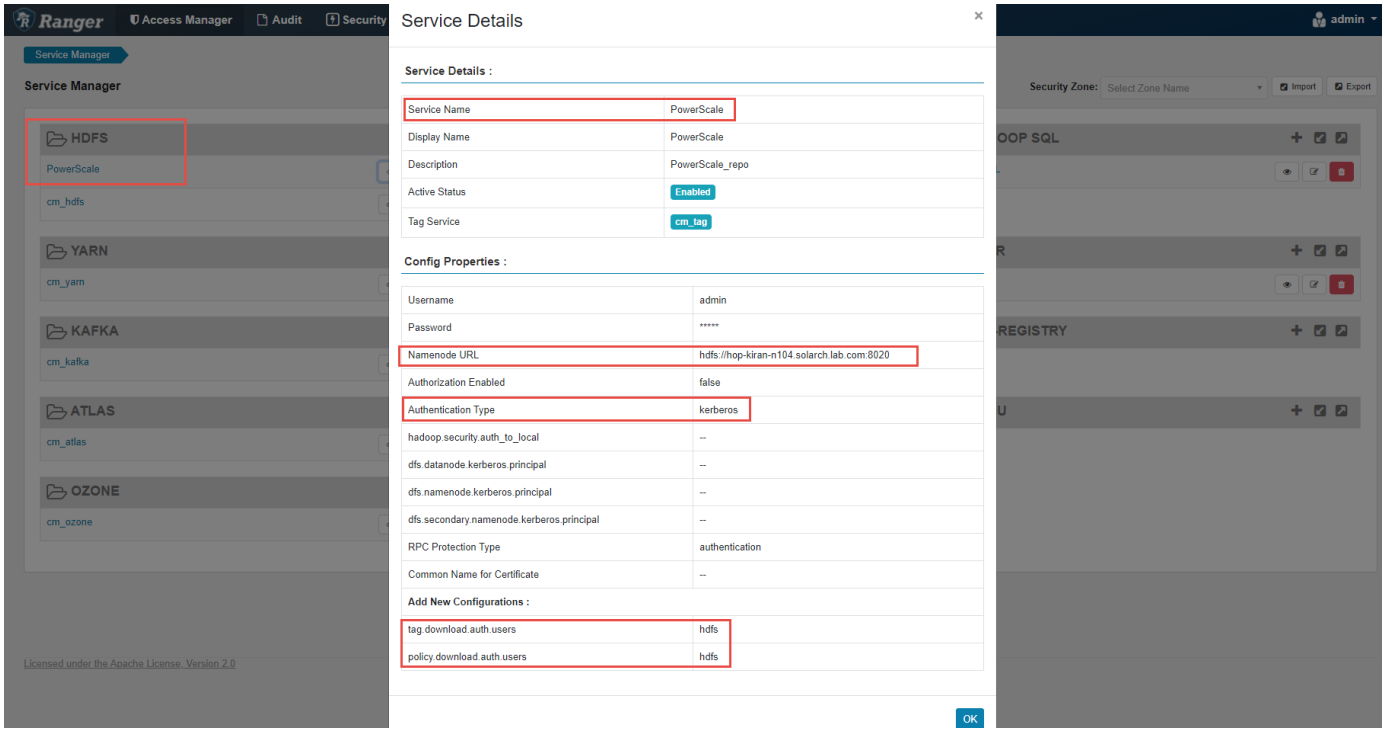


Figure 6 HDFS create service for Isilon HDFS repository.

4.8 Validating CDP deployment and PowerScale integration

4.8.1 Overview

Table 2 lists the validation procedures for the deployment of Hadoop tiered storage and integration with a PowerScale cluster.

Table 2 Validation procedure

Step	Category	Validation Procedures
1	Cloudera Manager and CDP Private Cloud Base	Set up the CDP Private Cloud Base cluster
2	Kerberos setup and configuration	Kerberos functionality testing
3	Kerberos security	Kerberos user and non-kerberos user testing on kerberized Hadoop and PowerScale clusters
4	RANGER	Cross namespace Ranger setup and functionality testing
5	HDFS	Hadoop commands for all permutations of default and nondefault file systems as inputs and output directories.
6	Yarn/MapReduce	MapReduce – WordCount job run for all permutations of default and nondefault file systems as input directories.

7	Spark	Spark – WordCount and LineCount job run for all permutations of default and nondefault file systems as input and output directories.
	Spark SQL Hive ACID tables	Hive ACID table operation from Spark.
8	DistCP	DistCp operation testing in and between local DAS HDFS and remote PowerScale HDFS.
9	Atlas	Functional validation.
10	Hive / Hive on TEZ	DDL and DML operations, including ACID tables (ORC) operations.
11	Impala	DDL and DML operations.
12	Zeppelin	Validate Livy and Spark interpreters.
13	HUE	Validate Hive and Impala table access from PowerScale
14	DAS	Create table and insert data into database on PowerScale

Note: In the secured environment, the spark shell explicitly needs remote HDFS information. Pass the information as a parameter while submitting the job.
Example: `spark.yarn.access.hadoopFileSystems=hdfs://PowerScale.dellcloudera.com:8020`

Note: set `hive.auto.convert.join=false` if you encounter help space errors on join queries between the tables on local and remote HDFS.

4.8.2 Validation procedures

4.8.2.1 Cloudera Manager and CDP Private Cloud Base

The steps for performing a Cloudera Manager and CDP Private Cloud Base smoke test are as follows.

Set up a five-node CDP cluster for local DAS HDFS and a four-node PowerScale cluster as the remote HDFS cluster.

In the Cloudera Manager web UI, run all the service checks, and stop and restart all the services.

For more information about the detailed test cases and their results see validation results section

4.8.2.2 Kerberos and Ranger environment configuration

Table 3 outlines the validation procedures for the Kerberos and Ranger environments.

Table 3 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	Kerberize the CDP (local DAS HDFS) and PowerScale cluster

	4	Verify that keytabs were generated for all the CDP 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 HDFS cluster
	3	Add the Ranger plug-in for HDFS, YARN, and Hive

For more information about the detailed test cases and their results see validation results section

4.8.2.3 Kerberos security

We validated Kerberos security as follows.

1. Kerberize the local DAS HDFS and remote PowerScale 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 2020-03-24 13:13 hdfs://hdp-
master03.bigdata.emc.local:8020/user/hdp-user1/mr/redhat-release
```

3. Access the remote PowerScale 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 2020-03-24 13:13 hdfs://isi-
cluster-hdfs1.bigdata.emc.local:8020/user/hdp-user1/mr/redhat-release
```

For more information about the detailed test cases and their results see validation results section

4.8.2.4 Ranger policies

The steps for validating Ranger policies are as follows.

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 2020-03-24 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.authorization.hado
op.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"
```

Note: CDP Ranger had an issue described at RANGER-2626. A OneFS custom patch was developed to fix this Ranger issue for the QATS testing, same fix is incorporated in the ***"[PSP-1091]"*** patch will be officially made available as a Roll Up Patch (RUP) in May 2021.

4.8.2.5 HDFS, YARN, MapReduce, and Spark services

Table 4 outlines the procedures for validating HDFS, YARN, MapReduce, and Spark services with Ranger policies and Kerberos security enabled.

Note: For a list of specific test steps and results, see appendix HDFS, YARN/MapReduce, Spark, Spark SQL.

Table 4 Validation of HDFS, YARN, MapReduce, and Spark.

Test case name	Step	Description

MapReduce /YARN (word count)	1	Create a hdp-user1 home directory on the CDH (local HDFS) and PowerScale clusters
	2	In the Ranger UI, assign RWX on the `/user/hdp-user1` directory for hdp-user1 on the CDH (local HDFS) and PowerScale cluster
	3	Put local file `/etc/redhat-release` on the CDH (local HDFS) file system
	4	Put local file `/etc/redhat-release` on the PowerScale HDFS
	5	Run MapReduce WordCount job on input from CDH (local HDFS) with output to the PowerScale HDFS
	6	Run MapReduce WordCount job on input from PowerScale HDFS1, with output to CDH (local HDFS)
Spark (line count and word count)	1	Put local file `/etc/passwd` on the CDH (local HDFS) file system
	2	Put local file `/etc/passwd` on the PowerScale HDFS
	3	Run a Spark LineCount/WordCount job on input from the primary CDH (local HDFS), with output to the secondary PowerScale HDFS
	4	Run a Spark LineCount/WordCount job on input from the secondary PowerScale HDFS, with output to the primary CDH (local HDFS)
Spark SQL Hive ACID tables	1	Log in into the Spark Shell and pass hive-warehouse-connector-assembly jar hive-warehouse-connector-assembly-1.0.0.7.1.6.0-297.jar and following parameters. `--conf "spark.sql.extensions=com.qubole.spark.hiveacid.HiveAcidAutoConvertExtension" --conf "spark.datasource.hive.warehouse.read.via.llap=false" \ --conf "spark.sql.hive.hwc.execution.mode=spark" \ --conf "spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKyroRegistrator" \ --conf "spark.yarn.access.hadoopFileSystems=$POWERSCALE_HDFS"`
	2	Access Hive Data warehouse and run select query on hive transaction tables.

For the detailed test cases and their results see validation results section

4.8.2.6 Hive and Hive on Tez

The steps for validating Hive service with Ranger and Kerberos security enabled.

Note: For a list of specific test steps and output, see appendix Hive.

1. Ensure that the local DAS HDFS and remote PowerScale HDFS clusters are Kerberized and the necessary permission for user hdp-user1 RWX access are provided.
2. Switch to hdp-user1 and run the beeline CLI to create a remote database location on the remote PowerScale 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
```

3. Create an internal nonpartitioned table and load data using local in path:

```
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
```

For the detailed test cases and their results see validation results section

4.8.2.7 Impala

The steps for validating Impala service with Kerberos security enabled.

Note: For a list of specific test steps and output, see appendix Impala.

1. Ensure that the local DAS HDFS and remote PowerScale HDFS clusters are Kerberized and the necessary permission for user hdp-user1 RWX access are provided.
2. Switch to hdp-user1 and run the impala-shell to create a remote database location on the remote PowerScale 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
```

3. Create an internal nonpartitioned table and load data using local in path:

```
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
```

For the detailed test cases and their results see validation results section

4.8.2.8 Distcp between HDFS and PowerScale cluster.

The steps for validating Distcp with Kerberos security enabled between and within HDFS and PowerScale clusters.

Note: For a list of specific test steps and output, see appendix Distcp.

1. Run DistCp to copy a sample file from the remote PowerScale HDFS to the local DAS HDFS.

```
sudo -u hdfs hadoop distcp -pc
hdfs://PowerScale.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 2020-03-07 13:26 hdfs://hdp-
master.bigdata.emc.local:8020/tmp/redhat-release
```

For the detailed test cases and their results see validation results section

4.8.2.9 ATLAS service

The steps for validating Atlas with Ranger policies and Kerberos security enabled. Between and within HDFS and PowerScale clusters.

Note: For a list of specific test steps and output, see appendix ATLAS.

Table 5 Validation step for Atlas service

Test case name	Step	Description
Atlas	1	Validate service health check is green
	2	• Log in into the webui • Check the HDFS path for the data located on PowerScale • Check databases and table lineage for data stored on PowerScale

For the detailed test cases and their results see validation results section

4.8.2.10 ZEPPELIN service

The steps for validating Zeppelin with Ranger policies and Kerberos security enabled. Between and within HDFS and PowerScale clusters.

Note: For a list of specific test steps and output, see appendix Zeppelin.

Table 6 Validation step for PIG service

Test case name	Step	Description
Zeppelin	1	Validate service health check is green
	2	• Log in into webui • Check Livy, and Spark interpreter are connecting • Run sample Spark wordcount and linecount job for the data stored on PowerScale. • Store the wordcount and line count job output back to PowerScale and check the results from the terminal.

For the detailed test cases and their results see validation results section

4.8.2.11 DAS service

The steps for validating DAS with Ranger policies and Kerberos security enabled. Between and within HDFS and PowerScale clusters.

Note: For a list of specific test steps and output, see appendix DAS.

Table 7 Validation step for DAS service

Test case name	Step	Description
DAS	1	Validate service health check is green
	2	• Log in into webui • Check the databases and tables created on PowerScale. • Compose a new create external table on PowerScale and load data. • Import table data form local into the database located on PowerScale.

For the detailed test cases and their results see validation results section

Note: If you see 500 error for the DAS webui then change “data_analytics_studio_user_authentication” from “Default” to “None”.

4.8.2.12 HUE service

The steps for validating HUE with Ranger policies and Kerberos security enabled. Between and within HDFS and PowerScale clusters.

Note: For a list of specific test steps and output, see appendix HUE

Table 8 Validation step for PIG service

Test case name	Step	Description

HUE	1	Validate service health check is green
	2	• Log in into webui • Browse metastore schema for the database and tables created on the PowerScale. • Run sample Hive and Impala queries on the tables located on PowerScale.

For the detailed test cases and their results see validation results section

4.8.2.13 Validation results

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

Category	Test case name	Expected test results	Status
Cloudera Manager	Validation of install and configuration	Deployment is completed successfully	PASS
	Usability and functionality test of UI	Service check runs successfully	PASS
Kerberos and Ranger environment configuration	Kerberos setup	Command runs successfully	PASS
	Ranger setup	Command runs successfully	PASS
MapReduce	Word count	Command runs successfully	PASS
Spark	Line count and word count	Command runs successfully	PASS
Spark SQL	Connect Hive Metastore and operations on transaction (ACID) tables	Command runs successfully	PASS
Hive and Hive on Tez	DDL operations • LOAD data local in path • INSERT into table • INSERT Overwrite TABLE	Command runs successfully	PASS
	DML operations • Query local database tables • 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
Impala	DDL operations • LOAD data local in path • INSERT into table • INSERT Overwrite TABLE	Command runs successfully	PASS
	DML operations • Query local database tables • 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
	Table-level and column-level statistics	Command runs successfully	PASS
DistCp	DisctCp and backup script	Command runs successfully	PASS
ATLAS	ATLAS operations	Command runs successfully	PASS
ZEPPELIN	Zeppelin operations	Command runs successfully	PASS
DAS	DAS operations	Command runs successfully	PASS
Hue	Hue operations	Command runs successfully	PASS

5 Conclusion

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. Cloudera, 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 Cloudera validated system configurations for DAS storage for Hadoop with a PowerScale 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.

A Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

A.1 Related resources

1. Cloudera Manager Installation
2. Installing CDH and other services
3. Enabling Kerberos authentication for CDH
4. Using CDH with PowerScale storage
5. OneFS 8.2.0 HDFS Reference Guide
6. OneFS 8.2.0 Web Admin Guide
7. VMware Virtual SAN 6.0 Performance—Scalability and Best Practices Technical White Paper
8. Performance Best Practices for VMware vSphere 6.0

B Test results for reference

B.1 Kerberos

```
[root@hop-kiran-cdp ~]# kinit -kt /etc/security/keytabs/hive.service.keytab hive/hop-kiran-cdp.solarch.lab.emc.com
[root@hop-kiran-cdp ~]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hive/hop-kiran-cdp.solarch.lab.emc.com@EXAMPLE2.COM

Valid starting       Expires              Service principal
04/06/2020 10:08:52  04/07/2020 10:08:52  krbtgt/EXAMPLE2.COM@EXAMPLE2.COM
[root@hop-kiran-cdp ~]# hadoop fs -ls hdfs://kb-hdp-z3.hop-isi-dd.solarch.lab.emc.com:8020/RESTRICTED_ACCESS
```

Figure 7 Sample Kerberos kinit

B.2 HDFS

Sample HDFS commands on HDFS and PowerScale cluster

```
sudo -u hdfs kinit -kt /etc/security/keytabs/hdfs.headless.keytab hdfs
++ sudo -u hdfs hdfs dfs -rm -skipTrash -r hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/user/hdfs/mr
Deleted hdfs://hop-kiran-cdp.solarch.lab.emc.com:8020/user/hdfs/mr
++ sudo -u hdfs hdfs dfs -rm -skipTrash -r hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020/user/hdfs/mr
Deleted hdfs://kb-hdp-z3.hop-isi-dd.solarch.lab.emc.com:8020/user/hdfs/mr
++ sudo -u hdfs hdfs dfs -mkdir -p hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/user/hdfs/mr
++ sudo -u hdfs hdfs dfs -mkdir -p hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020/user/hdfs/mr
++ sudo -u hdfs hdfs dfs -put /etc/redhat-release hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/user/hdfs/mr/
++ sudo -u hdfs hdfs dfs -put /etc/redhat-release hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020/user/hdfs/mr/
++ sudo -u hdfs hdfs dfs -ls hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/user/hdfs/mr/
Found 1 items
-rw-r--r--   3 hdfs hdfs         37 2020-04-02 05:25 hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/user/hdfs/mr/redhat-release
++ sudo -u hdfs hdfs dfs -ls hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020/user/hdfs/mr/
Found 1 items
-rw-r--r--   3 hdfs hdfs         37 2020-04-02 05:25 hdfs://kb-hdp-z3.hop-
isi-dd.solarch.lab.emc.com:8020/user/hdfs/mr/redhat-release
```

B.3 YARN/MapReduce

Mapreduce word count job sample results for source and destination on HDFS and PowerScale and conversely.

```
++ sudo -u hdfs yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-
mapreduce-examples.jar wordcount hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/user/hdfs/mr/redhat-release hdfs://kb-hdp-
z3.hop-isi-dd.solarch.lab.emc.com:8020/user/hdfs/mr/output
20/04/02 05:25:21 INFO client.RMProxy: Connecting to ResourceManager at hop-
kiran-cdp.solarch.lab.emc.com/10.246.156.100:8050
20/04/02 05:25:22 INFO client.AHSProxy: Connecting to Application History
server at hop-kiran-cdp.solarch.lab.emc.com/10.246.156.100:10200
20/04/02 05:25:22 INFO hdfs.DFSClient: Created token for hdfs:
HDFS_DELEGATION_TOKEN owner=hdfs@EXAMPLE2.COM, renewer=yarn/hop-kiran-
cdp.solarch.lab.emc.com@EXAMPLE2.COM, realUser=, issueDate=1585819526100,
maxDate=1586424326100, sequenceNumber=0, masterKeyId=0 on kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020
"
"
"
20/04/02 05:25:24 INFO mapreduce.Job: Running job: job_1585709743543_0003
20/04/02 05:25:34 INFO mapreduce.Job: Job job_1585709743543_0003 running in
uber mode : false
20/04/02 05:25:34 INFO mapreduce.Job:  map 0% reduce 0%
20/04/02 05:25:39 INFO mapreduce.Job:  map 100% reduce 0%
20/04/02 05:25:44 INFO mapreduce.Job:  map 100% reduce 100%
20/04/02 05:25:45 INFO mapreduce.Job: Job job_1585709743543_0003 completed
successfully
"
"
"
      Shuffle Errors
            BAD_ID=0
            CONNECTION=0
            IO_ERROR=0
            WRONG_LENGTH=0
            WRONG_MAP=0
            WRONG_REDUCE=0
      File Input Format Counters
            Bytes Read=37
      File Output Format Counters
            Bytes Written=47
```

B.4 Spark

Spark word count and line count job sample results for source and destination on HDFS and PowerScale and conversely.

```
++ sudo -u hdfs spark-shell -i /tmp/spark_line_word_count.scala --conf
spark.yarn.access.hadoopFileSystems=hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020 --conf 'spark.driver.args=hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/user/hdfs/spark/redhat-release hdfs://kb-hdp-
z3.hop-isi-dd.solarch.lab.emc.com:8020/user/hdfs/spark/output'
Setting default log level to "WARN".
```

```
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
Spark context Web UI available at http://hop-kiran-
cdp.solarch.lab.emc.com:4040
Spark context available as 'sc' (master = yarn, app id =
application_1586182567199_0016).
Spark session available as 'spark'.

[Stage 0:>                                                          (0 + 2) /
2]

++ sudo -u hdfs spark-shell -i /tmp/spark_line_word_count.scala --conf
spark.yarn.access.hadoopFileSystems=hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020 --conf 'spark.driver.args=hdfs://kb-hdp-z3.hop-
isi-dd.solarch.lab.emc.com:8020/user/hdfs/spark/redhat-release hdfs://hop-
kiran-cdp.solarch.lab.emc.com:8020/user/hdfs/spark/output'
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
Spark context Web UI available at http://hop-kiran-
cdp.solarch.lab.emc.com:4040
Spark context available as 'sc' (master = yarn, app id =
application_1586182567199_0017).
Spark session available as 'spark'.

[Stage 0:>                                                          (0 + 2) /
2]
[Stage 0:=============================>                           (1 + 1) /
2]
[Stage 1:=============================>                           (1 + 1) /
2]
```

B.5 Spark SQL

Spark SQL access Hive Metastore and run queries on Hive ACID tables.

```
sudo -u hdfs spark-shell -i /tmp/spark_hive_trans_tables.scala --jars
/opt/cloudera/parcels/CDH/lib/hive_warehouse_connector/hive-warehouse-
connector-assembly-1.0.0.7.1.6.0-297.jar --conf
spark.sql.extensions=com.qubole.spark.hiveacid.HiveAcidAutoConvertExtension -
-conf spark.datasource.hive.warehouse.read.via.llap=false --conf
spark.sql.hive.hwc.execution.mode=spark --conf
spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKyroRegistrator
--conf spark.yarn.access.hadoopFileSystems=hdfs://hop-kiran-
n104.solarch.lab.com:8020
Spark context Web UI available at http://hop-kiran-n71.solarch.lab.com:4042
Spark context available as 'sc' (master = yarn, app id =
application_1621156118210_0024).
Spark session available as 'spark'.
```

```
21/05/17 02:47:07 WARN conf.HiveConf: HiveConf of name hive.masking.algo does
not exist
Hive Session ID = 6bb9c232-08da-457f-bea5-d7f93b268fef
== Physical Plan ==
Execute ShowDatabasesCommand
   +- ShowDatabasesCommand


+------------------+
|      databaseName|
+------------------+
|           default|
|   impala_local_db|
|  impala_remote_db|
|information_schema|
|          local_db|
|         remote_db|
|               sys|
+------------------+

21/05/17 02:47:10 WARN conf.HiveConf: HiveConf of name hive.masking.algo does
not exist
21/05/17 02:47:11 WARN conf.HiveConf: HiveConf of name hive.masking.algo does
not exist
21/05/17 02:47:11 WARN conf.HiveConf: HiveConf of name hive.masking.algo does
not exist
== Physical Plan ==
*(1) Scan
HiveAcidRelation(org.apache.spark.sql.SparkSession@2d30d676,remote_db.passwd_
int_trans,Map(transactional -> true, numFilesErasureCoded -> 0,
bucketing_version -> 2, transient_lastDdlTime -> 1620219012,
transactional_properties -> default, table -> remote_db.passwd_int_trans))
[user_name#38,password#39,user_id#40,group_id#41,user_id_info#42,home_dir#43,
shell#44] PushedFilters: [], ReadSchema:
struct<user_name:string,password:string,user_id:string,group_id:string,user_i
d_info:string,home_d...


21/05/17 02:47:12 WARN conf.HiveConf: HiveConf of name hive.masking.algo does
not exist
 [Stage 0:>                                                          (0 + 0)
/ 1] [Stage 0:>                                                          (0 +
1) / 1]
+------------+--------+-------+--------+------------+------------------+-----
--------+
|   user_name|password|user_id|group_id|user_id_info|          home_dir|
shell|
+------------+--------+-------+--------+------------+------------------+-----
--------+
```

```
|        HTTP|       x|   1012|    1014|            |        /home/HTTP|
/bin/bash|
|         adm|       x|      3|       4|         adm|
/var/adm|/sbin/nologin|
|       atlas|       x|   1003|    1005|            |       /home/atlas|
/bin/bash|
|       flume|       x|   1008|    1010|            |       /home/flume|
/bin/bash|
|         ftp|       x|     14|      50|    FTP User|
/var/ftp|/sbin/nologin|
|        halt|       x|      7|       0|        halt|             /sbin|
/sbin/halt|
|         hue|       x|   1014|    1016|            |         /home/hue|
/bin/bash|
|      impala|       x|   1015|    1017|            |      /home/impala|
/bin/bash|
|       kafka|       x|   1016|    1018|            |       /home/kafka|
/bin/bash|
|        knox|       x|   1019|    1021|            |        /home/knox|
/bin/bash|
|        kudu|       x|   1021|    1023|            |        /home/kudu|
/bin/bash|
|        livy|       x|   1023|    1025|            |        /home/livy|
/bin/bash|
|       llama|       x|   1022|    1024|            |       /home/llama|
/bin/bash|
|          lp|       x|      4|       7|          lp|
/var/spool/lpd|/sbin/nologin|
|      mapred|       x|   1024|    1026|            |      /home/mapred|
/bin/bash|
|    operator|       x|     11|       0|    operator|
/root|/sbin/nologin|
|     phoenix|       x|   1026|    1028|            |     /home/phoenix|
/bin/bash|
|     postfix|       x|     89|      89|
|/var/spool/postfix|/sbin/nologin|
|rangerlookup|       x|   1029|    1031|            |/home/rangerlookup|
/bin/bash|
|        root|       x|      0|       0|        root|             /root|
/bin/bash|
+------------+--------+-------+--------+------------+------------------+-----
---------+
only showing top 20 rows

21/05/17 02:47:19 WARN conf.HiveConf: HiveConf of name hive.masking.algo does
not exist
21/05/17 02:47:19 WARN conf.HiveConf: HiveConf of name hive.masking.algo does
not exist
== Physical Plan ==
*(2) BroadcastHashJoin [user_id#140], [user_id#133], Inner, BuildRight
```

```
:- *(2) Filter ((isnotnull(group_id#141) && (group_id#141 = 1002)) &&
isnotnull(user_id#140))
:  +- *(2) Scan
HiveAcidRelation(org.apache.spark.sql.SparkSession@2d30d676,local_db.passwd_i
nt_trans_insert_only,Map(transactional -> true, numFilesErasureCoded -> 0,
bucketing_version -> 2, transient_lastDdlTime -> 1620224164,
serialization.format -> 1, transactional_properties -> insert_only, table ->
local_db.passwd_int_trans_insert_only))
[user_name#138,password#139,user_id#140,group_id#141,user_id_info#142,home_di
r#143,shell#144] PushedFilters: [IsNotNull(group_id), EqualTo(group_id,1002),
IsNotNull(user_id)], ReadSchema:
struct<user_name:string,password:string,user_id:string,group_id:string,user_i
d_info:string,home_d...
+- BroadcastExchange HashedRelationBroadcastMode(List(input[2, string,
false])), [id=#38]
   +- *(1) Filter isnotnull(user_id#133)
      +- Scan hive remote_db.passwd_ext_nonpart [user_name#131, password#132,
user_id#133, group_id#134, user_id_info#135, home_dir#136, shell#137],
HiveTableRelation `remote_db`.`passwd_ext_nonpart`,
org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, [user_name#131,
password#132, user_id#133, group_id#134, user_id_info#135, home_dir#136,
shell#137]


21/05/17 02:47:20 WARN conf.HiveConf: HiveConf of name hive.masking.algo does
not exist
 [Stage 3:==============================>                          (1 + 0)
/ 2]
+---------+--------+-------+--------+------------+------------+---------+----
-----+--------+-------+--------+------------+------------+---------+
|user_name|password|user_id|group_id|user_id_info|    home_dir|
shell|user_name|password|user_id|group_id|user_id_info|    home_dir|
shell|
+---------+--------+-------+--------+------------+------------+---------+----
-----+--------+-------+--------+------------+------------+---------+
|   apache|       x|   1002|    1002|            |/home/apache|/bin/bash|
apache|       x|   1002|    1002|            |/home/apache|/bin/bash|
+---------+--------+-------+--------+------------+------------+---------+----
-----+--------+-------+--------+------------+------------+---------+
```

B.6 Hive

Hive DML and DDL queries run on TEZ engine, here only the subset of DML and DDL is shown for reference.

```
++ sudo -u hdfs kinit -kt /etc/security/keytabs/hive.service.keytab hive/hop-
kiran-cdp.solarch.lab.emc.com
++ cat
```

```
++ sudo -u hdfs beeline -u 'jdbc:hive2://hop-kiran-
cdp.solarch.lab.emc.com:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespa
ce=hiveserver2' -f /tmp/hive_ddl.sql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/3.1.5.0-152/hive/lib/log4j-slf4j-
impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/3.1.5.0-152/hadoop/lib/slf4j-
log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://hop-kiran-
cdp.solarch.lab.emc.com:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespa
ce=hiveserver2
20/04/03 06:00:01 [main]: INFO jdbc.HiveConnection: Connected to hop-kiran-
cdp.solarch.lab.emc.com:10000
Connected to: Apache Hive (version 3.1.0.3.1.5.0-152)
Driver: Hive JDBC (version 3.1.0.3.1.5.0-152)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://hop-kiran-cdp.solarch.lab.emc> -- Create remote DB DDL
0: jdbc:hive2://hop-kiran-cdp.solarch.lab.emc> DROP DATABASE IF EXISTS
remote_DB CASCADE;
INFO  : Compiling command(queryId=hive_20200403060002_740fe86a-cace-437a-
8ab0-4336692ff588): DROP DATABASE IF EXISTS remote_DB CASCADE
INFO  : Semantic Analysis Completed (retrial = false)
“
“
No rows affected (0.666 seconds)

0: jdbc:hive2://hop-kiran-cdp.solarch.lab.emc> CREATE database remote_DB
COMMENT 'Holds all the tables da
ta in remote location Hadoop cluster' LOCATION 'hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020/user
/hive/remote_DB' ;
“
“
INFO  : Compiling command(queryId=hive_20200403060002_ea8a895f-0a92-451f-
a3dc-
No rows affected (0.07 seconds)

0: jdbc:hive2://hop-kiran-cdp.solarch.lab.emc> USE remote_DB;
“
INFO  : OK
No rows affected (0.026 seconds)
0: jdbc:hive2://hop-kiran-cdp.solarch.lab.emc>
0: jdbc:hive2://hop-kiran-cdp.solarch.lab.emc> 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 ':'
. . . . . . . . . . . . . . . . . . . . . . . .> STORED AS TEXTFILE;
INFO  : Compiling command(queryId=hive_20200403060003_a3788aeb-4baa-48e0-
a727-“
“
“
“
No rows affected (0.113 seconds)
0: jdbc:hive2://hop-kiran-cdp.solarch.lab.emc> LOAD data local inpath
'/etc/passwd'  into TABLE passwd_in
t_nonpart;
“
“
“
Time taken: 0.202 seconds
INFO  : OK
No rows affected (0.266 seconds)
```

B.7 Impala

Here are only a subset of results from the Impala service operations for reference.

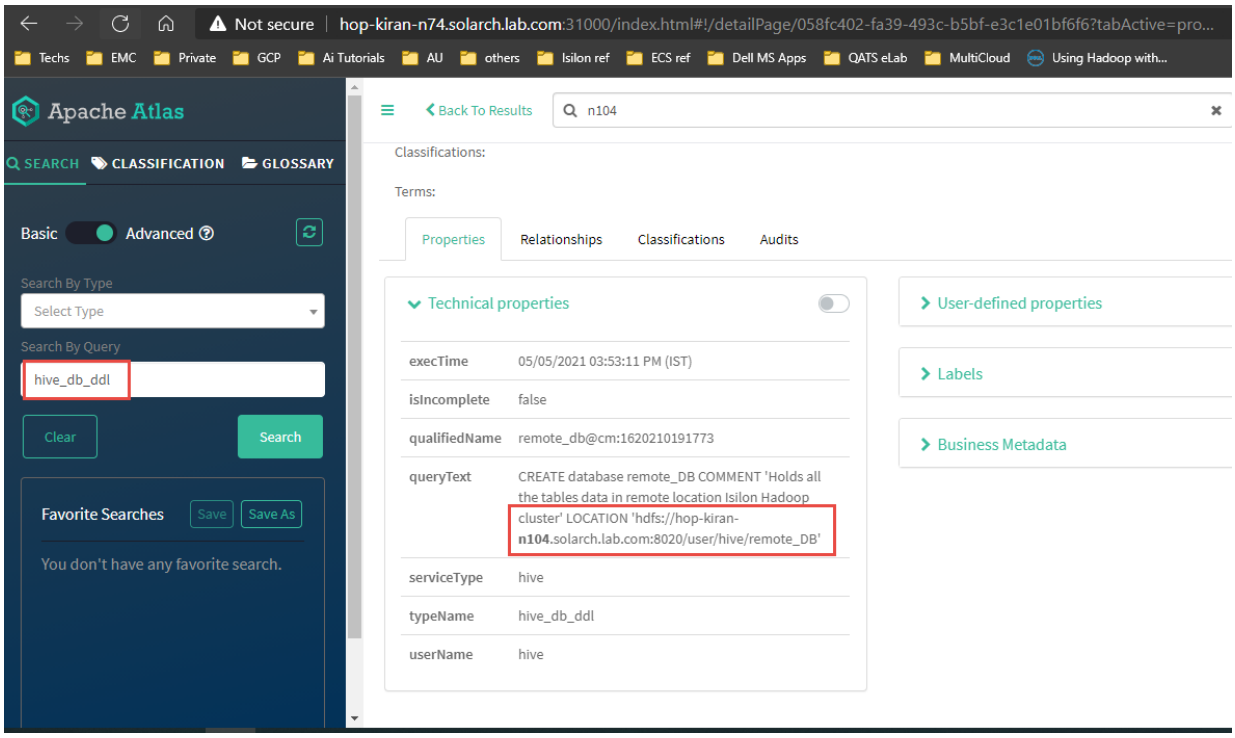
```
++ sudo -u hdfs impala-shell -k -i hop-rd540-08.dellcloudera.com:21000 -u hive -
f /tmp/hive_dml.sql
Starting Impala Shell using Kerberos authentication
Using service name 'impala'
Opened TCP connection to hop-rd540-08.dellcloudera.com:21000
Connected to hop-rd540-08.dellcloudera.com:21000
Server version: impalad version 3.2.0-cdh6.3.2 RELEASE (build
1bb9836227301b839a32c6bc230e35439d5984ac)
Query: INVALIDATE METADATA
Query submitted at: 2020-05-04 20:58:39 (Coordinator: http://hop-rd540-
08.dellcloudera.com:25000)
Query progress can be monitored at: http://hop-rd540-
08.dellcloudera.com:25000/query_plan?query_id=85488bad65661b80:025e2b2b00000000
Fetched 0 row(s) in 4.27s
Query: SELECT * FROM local_db.passwd_ext_nonpart
Query submitted at: 2020-05-04 20:58:43 (Coordinator: http://hop-rd540-
08.dellcloudera.com:25000)
Query progress can be monitored at: http://hop-rd540-
08.dellcloudera.com:25000/query_plan?query_id=32408fe106fe7eee:4110995a00000000
+-----------------------------------------------------------+----------+--------
-+----------+-------------------------------------------------------------------+-
------------------------------+----------------+
| user_name                                                 | password | user_id
| group_id | user_id_info                                                      |
home_dir                      | shell          |
```

```
+-----------------------------------------------------------+----------+--------
-+----------+-------------------------------------------------------------------+-
------------------------------+----------------+
| root                                                      | x        | 0
| 0        | root                                                                 |
/root                         | /bin/bash      |
“
“                                                  | x        | 12      “
“
| systemd-network                                           | x        | 192
| 192      | systemd Network Management                                        | /
| /sbin/nologin  |
| dbus                                                      | x        | 81
| 81       | System message bus                                                | /
| nfsnobody                                                 | x        | 65534
| 65534    | Anonymous NFS User                                                |
/var/lib/nfs                  | /sbin/nologin  |
| tcpdump                                                   | x        | 72
| 72       |                                                                   | /
| /sbin/nologin  |
+-----------------------------------------------------------+----------+--------
-+----------+-------------------------------------------------------------------+-
------------------------------+----------------+
Fetched 100 row(s) in 4.95s
Query: SELECT * FROM remote_db.passwd_int_trans
Query submitted at: 2020-05-04 20:59:23 (Coordinator: http://hop-rd540-
08.dellcloudera.com:25000)
Query progress can be monitored at: http://hop-rd540-
08.dellcloudera.com:25000/query_plan?query_id=e04fc7150df7da9a:c2340aa700000000
Fetched 0 row(s) in 3.89s
Query: -- joins within local_db
SELECT * FROM local_db.passwd_ext_nonpart t1 JOIN local_db.passwd_int_nonpart t2
ON (t1.user_id = t2.user_id AND t1.group_id = '1002')
Query submitted at: 2020-05-04 20:59:27 (Coordinator: http://hop-rd540-
08.dellcloudera.com:25000)
Query progress can be monitored at: http://hop-rd540-
08.dellcloudera.com:25000/query_plan?query_id=fa48d7701ba0c31b:661dbfa900000000
+-----------+----------+---------+----------+--------------------+------------+-
----------+-----------+----------+---------+----------+--------------------+----
--------+-----------+
| user_name | password | user_id | group_id | user_id_info       | home_dir   |
shell     | user_name | password | user_id | group_id | user_id_info       |
home_dir   | shell     |
+-----------+----------+---------+----------+--------------------+------------+-
----------+-----------+----------+---------+----------+--------------------+----
--------+-----------+
| yarn      | x        | 1002    | 1002     | hadoop-svc-account | /home/yarn |
/bin/bash | yarn      | x        | 1002    | 1002     | hadoop-svc-account |
/home/yarn | /bin/bash |
```

```
+----------+---------+--------+---------+------------------+-----------+-
---------+----------+---------+--------+---------+------------------+----
-------+----------+
Fetched 1 row(s) in 0.58s
```

B.8 ATLAS

Here is only a subset of results from the ATLAS service operations for reference.



B.9 Distcp

Here we show the results from distcp operation between HDFS and PowerScale cluster.

```
++ sudo -u hdfs hadoop distcp -pc hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020/tmp/redhat-release hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/tmp/
ERROR: Tools helper /usr/hdp/3.1.5.0-152/hadoop/libexec/tools/hadoop-
distcp.sh was not found.
"
"
20/04/02 05:44:27 INFO tools.DistCp: Input Options:
20/04/02 05:44:30 INFO tools.DistCp: DistCp job-id: job_1585709743543_0005
20/04/02 05:44:30 INFO mapreduce.Job: Running job: job_1585709743543_0005
20/04/02 05:44:39 INFO mapreduce.Job: Job job_1585709743543_0005 running in
uber mode : false
20/04/02 05:44:39 INFO mapreduce.Job:  map 0% reduce 0%
20/04/02 05:44:44 INFO mapreduce.Job:  map 100% reduce 0%
```

```
20/04/02 05:44:44 INFO mapreduce.Job: Job job_1585709743543_0005 completed
successfully
20/04/02 05:44:44 INFO mapreduce.Job: Counters: 36
      File System Counters
            FILE: Number of bytes read=0
            FILE: Number of bytes written=242419
            FILE: Number of read operations=0
            FILE: Number of large read operations=0
            FILE: Number of write operations=0
            "
"
"
      DistCp Counters
            Bandwidth in Btyes=37
            Bytes Copied=37
            Bytes Expected=37
            Files Copied=1

++ sudo -u hdfs hdfs dfs -ls -R hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/tmp/redhat-release
-rw-r--r--   3 hdfs hdfs         37 2020-04-02 05:44 hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/tmp/redhat-release
++ sudo -u hdfs hdfs dfs -rm -skipTrash -r hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/tmp/redhat-release
Deleted hdfs://hop-kiran-cdp.solarch.lab.emc.com:8020/tmp/redhat-release
++ sudo -u hdfs hdfs dfs -rm -skipTrash -r hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020/tmp/redhat-release
Deleted hdfs://kb-hdp-z3.hop-isi-dd.solarch.lab.emc.com:8020/tmp/redhat-
release
++ sudo -u hdfs hdfs dfs -put /etc/redhat-release hdfs://hop-kiran-
cdp.solarch.lab.emc.com:8020/tmp/

++ sudo -u hdfs hadoop distcp -skipcrccheck -update /tmp/redhat-release
hdfs://kb-hdp-z3.hop-isi-dd.solarch.lab.emc.com/tmp/
"
"
"
      DistCp Counters
            Bandwidth in Btyes=37
            Bytes Copied=37
            Bytes Expected=37
            Files Copied=1
++ sudo -u hdfs hdfs dfs -ls -R hdfs://kb-hdp-z3.hop-isi-
dd.solarch.lab.emc.com:8020/tmp/redhat-release
-rw-r--r--   3 hdfs hdfs         37 2020-04-02 05:45 hdfs://kb-hdp-z3.hop-
isi-dd.solarch.lab.emc.com:8020/tmp/redhat-release
++ true
```

B.10 Zeppelin

Here is only a subset of results from the Zeppelin service operations for reference.

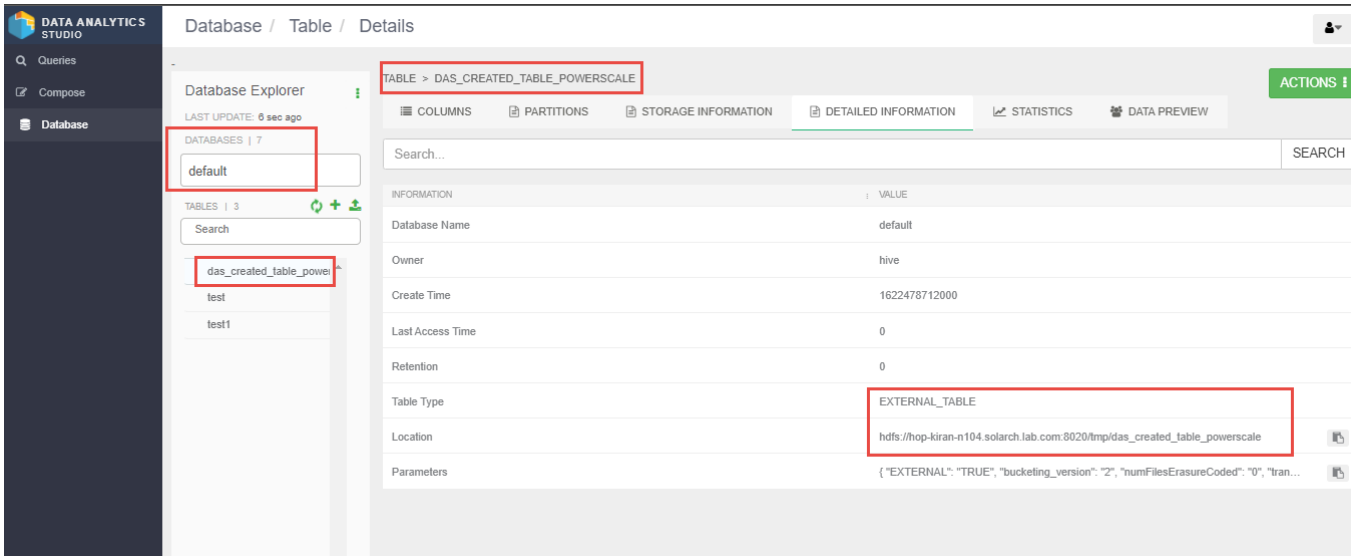
```
%livy
spark.version
var input="hdfs://hop-kiran-n104.solarch.lab.com:8020/tmp/redhat-release"
var output1="hdfs://hop-kiran-n104.solarch.lab.com:8020/tmp/op" + "-wc"
var text_file=sc.textFile(input)
val word_count=text_file.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
word_count.saveAsTextFile(output1)
word_count.collect()
var output2="hdfs://hop-kiran-n104.solarch.lab.com:8020/tmp/op" + "-lc"
var line_count=sc.parallelize(Seq(text_file.count()))
line_count.saveAsTextFile(output2)
line_count.collect()
```

```
res8: String = 2.4.0.7.1.6.0-297
input: String = hdfs://hop-kiran-n104.solarch.lab.com:8020/tmp/redhat-release
output1: String = hdfs://hop-kiran-n104.solarch.lab.com:8020/tmp/op-wc
text_file: org.apache.spark.rdd.RDD[String] = hdfs://hop-kiran-n104.solarch.lab.com:8020/tmp/redhat-release MapPartitionsRDD[15] at textFile at <console>:27
word_count: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[18] at reduceByKey at <console>:25
res10: Array[(String, Int)] = Array(((Core),1), (CentOS,1), (Linux,1), (release,1), (7.6.1810,1))
output2: String = hdfs://hop-kiran-n104.solarch.lab.com:8020/tmp/op-lc
line_count: org.apache.spark.rdd.RDD[Long] = ParallelCollectionRDD[20] at parallelize at <console>:27
res12: Array[Long] = Array(1)
```

Took 6 sec. Last updated by zadmin at May 30 2021, 7:01:11 PM. (outdated)

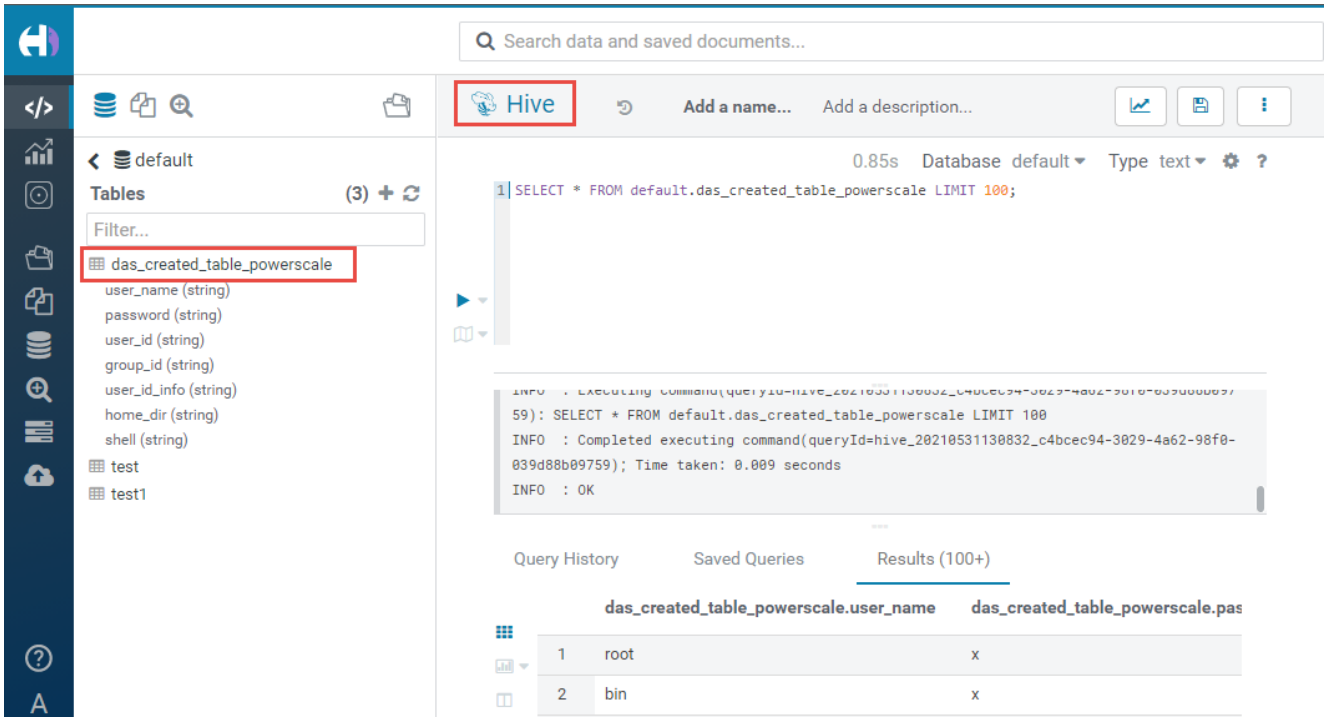
B.11 DAS

Here is only a subset of results from the DAS service operations for reference.



B.12 HUE

Here is only a subset of results from the HUE service operations for reference.


Search data and saved documents...
Hive
Add a name...
Add a description...
default
Tables (3)
Filter...
das_created_table_powerscale
user_name (string)
password (string)
user_id (string)
group_id (string)
user_id_info (string)
home_dir (string)
shell (string)
test
test1
0.85s
Database default
Type text
1 SELECT * FROM default.das_created_table_powerscale LIMIT 100;
59): SELECT * FROM default.das_created_table_powerscale LIMIT 100
INFO : Completed executing command(queryId=hive_20210531130832_c4bcec94-3029-4a62-98f0-039d88b09759); Time taken: 0.009 seconds
INFO : OK
Query History
Saved Queries
Results (100+)
das_created_table_powerscale.user_name
das_created_table_powerscale.pas
1 root x
2 bin x