CLOUDERA CDH WITH ECS CONFIGURATION AND BEST PRACTICES GUIDE

ABSTRACT

This white paper describes the configuration and best practices for using Cloudera CDH with EMC ECS.

March, 2016

REDEFINE

EMC WHITE

EMC²

To learn more about how EMC products, services, and solutions can help solve your business and IT challenges, contact your local representative or authorized reseller, visit www.emc.com, or explore and compare products in the EMC Store

Part Number HXXXXX <required, see Part numbers below for more info>

TABLE OF CONTENTS

PRODUCT VERSIONS

This document applies to following product versions.

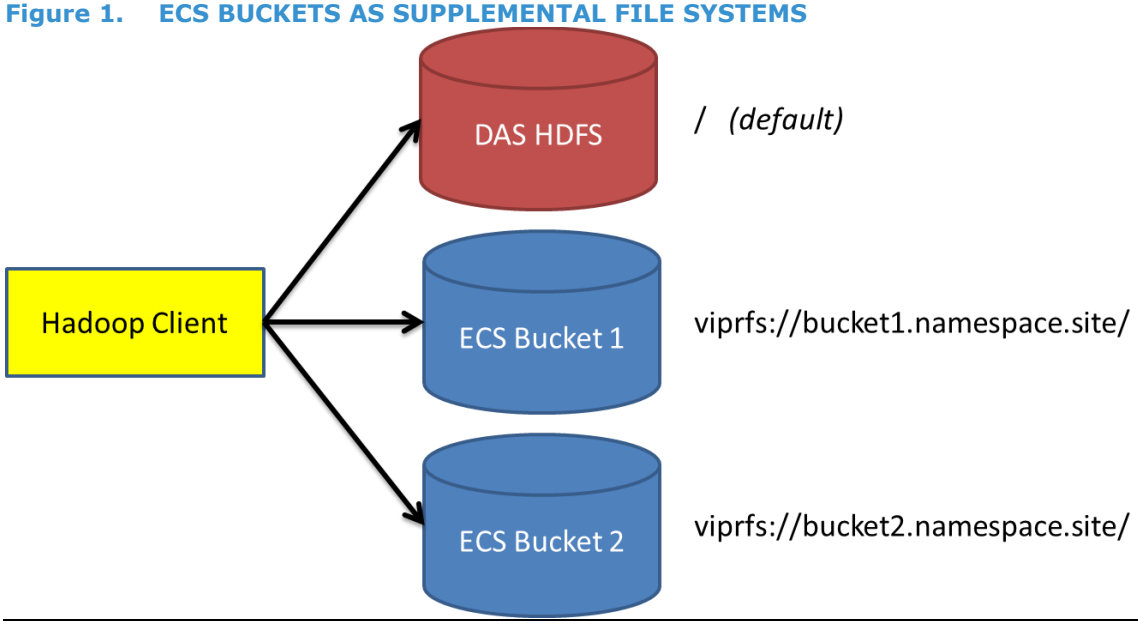
Table 1 PRODUCT VERSIONS

PRODUCT	VERSION
Cloudera CDH	5.4.8
Hadoop	2.6.0-cdh5.4.8
Cloudera Manager	5.5.1
CentOS	6.7
ECS	2.2.0.1.74973.118fe47
ViPRFS Client	2.2.0.0.74973.118fe47

DEPLOYMENT OPTIONS

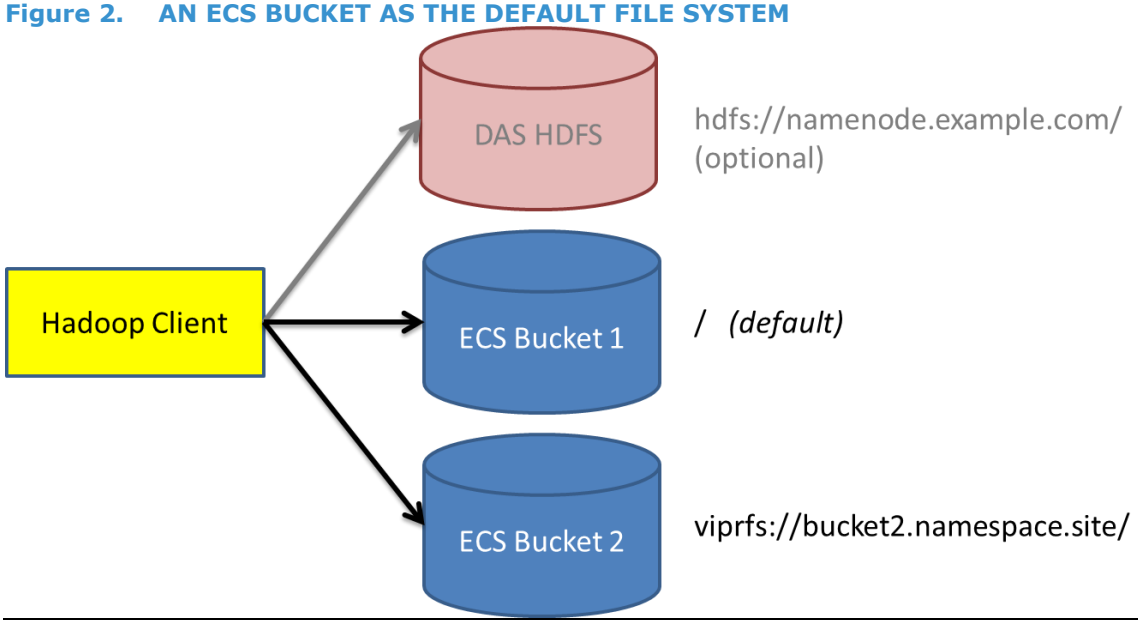
ECS buckets can be used as supplemental file systems. In this scenario, fs.defaultFS in core-site.xml remains as the default and references the HDFS NameNode. Applications can access data in any ECS bucket by using fully-qualified paths in the format "viprfs://bucket.namespace.site/". Jobs and applications may need to be reconfigured or updated to use a fully-qualified path. This is the recommended configuration and is what this document describes.

Figure 1. ECS BUCKETS AS SUPPLEMENTAL FILE SYSTEMS



An ECS bucket can also be used as the default file system in Hadoop, in which all Hadoop paths without an explicit URI prefix such as "hdfs://" refer to objects in a single ECS bucket. In this scenario, fs.defaultFS in core-site.xml is set to "viprfs://bucket.namespace.site/". Data on the traditional HDFS NameNode and DataNodes will be accessible by using the fully-qualified URI such as "hdfs://namenode.example.com/". Some applications that require the Apache HDFS implementation will not work in this Hadoop cluster.

Figure 2. AN ECS BUCKET AS THE DEFAULT FILE SYSTEM



The default authentication in Hadoop is called standard authentication. Standard authentication provides minimal security and is intended to run in an environment where everyone with network access to the Hadoop nodes can be trusted. Environments with untrusted users should use Kerberos to protect access to sensitive data.

COMPATIBLE HADOOP COMPONENTS

The following matrix lists the compatible Hadoop components for the product versions that this document applies to.

Table 2 COMPATIBLE HADOOP COMPONENTS

	NON-KERBEROS	KERBEROS
ECS AS DEFAULT FILE SYSTEM	MapReduce, Pig, Hive	MapReduce
ECS AS SUPPLEMENTAL FILE SYSTEM	MapReduce, Pig, Hive	MapReduce

INSTALLATION OVERVIEW (NON-KERBEROS)

The installation of Cloudera CDH in an ECS environment with standard Hadoop authentication (non-Kerberos) is composed of the following general steps.

1. Deploy a standard CDH cluster. This will use local disks (DAS) for HDFS, not ECS storage.
2. Configure users and buckets on ECS.
3. Configure the ECS client on the CDH nodes.
4. Optionally relocate selected Hadoop services from DAS HDFS to ECS storage.

INSTALLATION PROCEDURE (NON-KERBEROS)

DEPLOY CLOUDERA MANAGER AND CDH

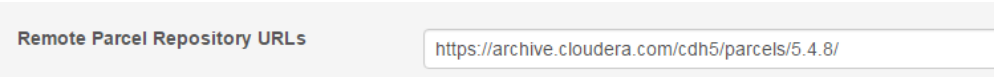
Complete details for this procedure can be found at http://www.cloudera.com/documentation/enterprise/5-4-x/topics/cm_ig_install_path_a.html?scroll=cmig_topic_6_5_unique_2. A concise list of steps is shown below.

```
[root@mycluster1-master-0 ~]#
```

```
wget http://archive.cloudera.com/cm5/installer/5.4.8/cloudera-manager-installer.bin
chmod a+x cloudera-manager-installer.bin
./cloudera-manager-installer.bin
```

Follow the steps as prompted by Cloudera Manager to build a traditional Hadoop cluster.

Note that to install the specific version of CDH that has been tested in this document, you will need to change the Remote Parcel Repository URL for CDH5 to https://archive.cloudera.com/cdh5/parcels/5.4.8/ as shown below.

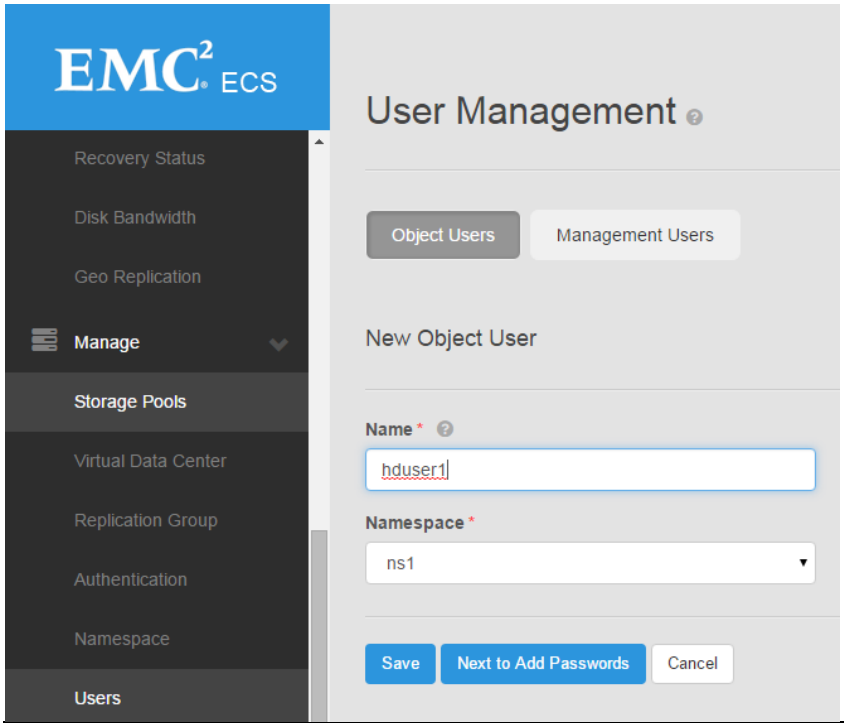


CONFIGURE ECS (NON-KERBEROS)

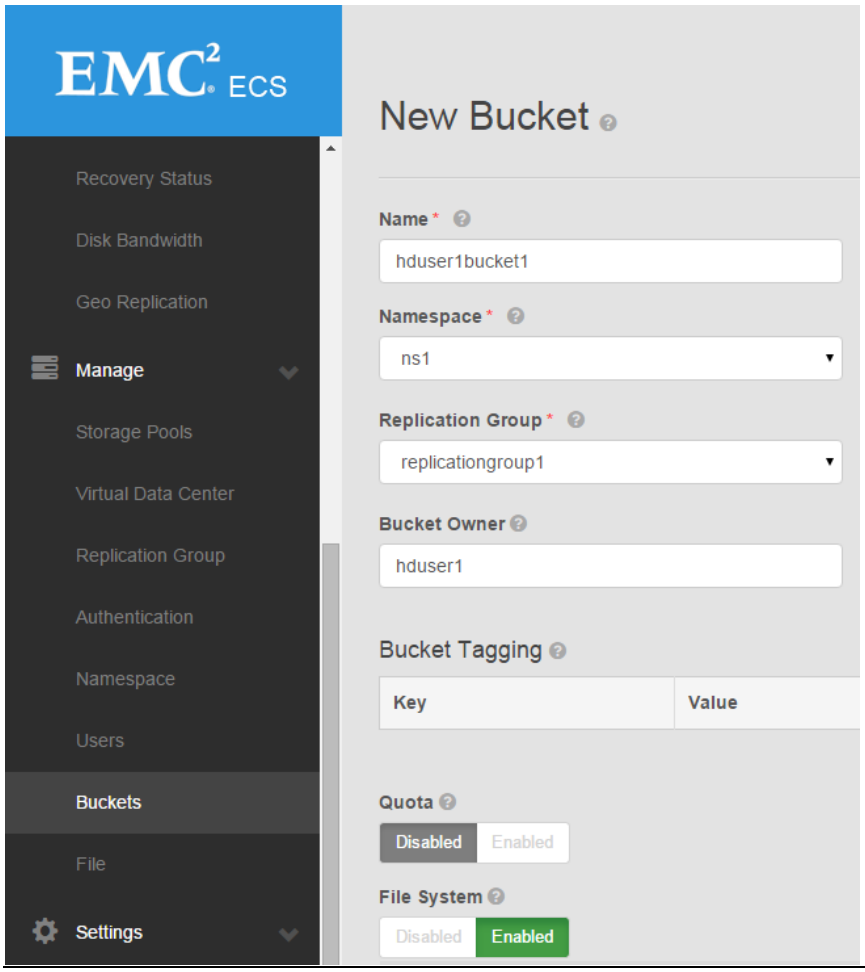
This document assumes that an ECS cluster has been installed and configured, and is ready for users and buckets to be created. For complete details, refer to the ECS 2.2 Product Documentation.

Complete documentation for configuring ECS can be found in the ECS 2.2 Data Access Guide. For convenience, below is a concise list of steps.

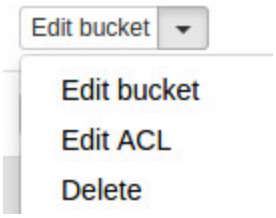
First, create an object user that will be the owner of the ECS bucket that we will use for our Hadoop data. When creating a bucket that will be used as Hadoop's default file system, it is recommended that the owner be called *hdfs* to maintain consistency with Apache Hadoop HDFS.

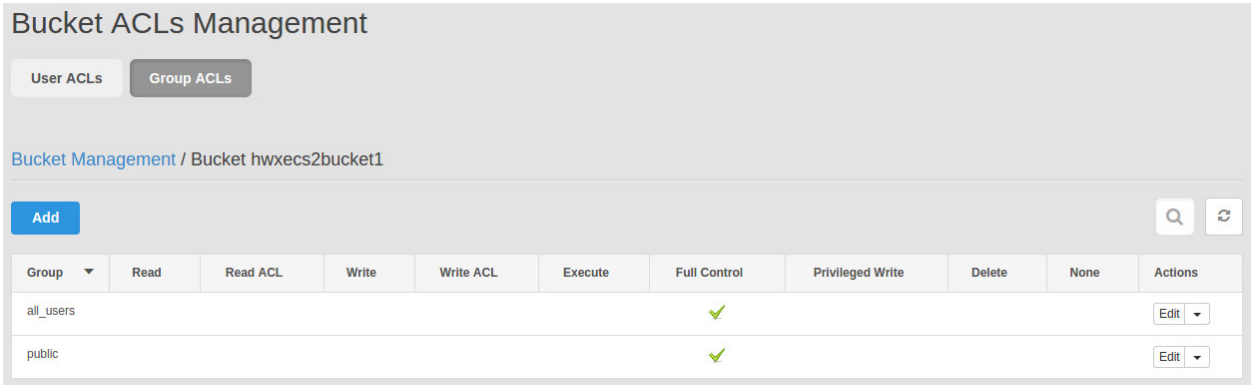


Next, create a bucket owned by the new object user. To allow HDFS access, you must set *File System Enabled* to On.



Next, edit the bucket ACL to allow the all_users and public groups Full Control. This is required for non-Kerberos access.





DEPLOY ECS HDFS CLIENT ON CDH NODES

The ECS HDFS client is a JAR library that allows Hadoop applications access the bucket on an ECS cluster. It is implemented as a Hadoop Compatible File System and uses the prefix "viprfs://". The installation of the ECS HDFS client can be automated by using Cloudera Manager to deploy the VIPRFS parcel.

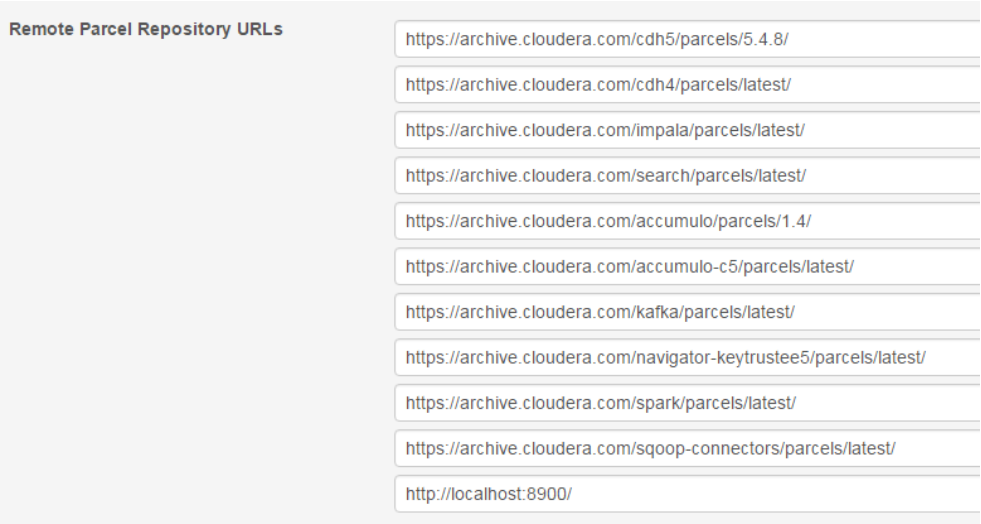
First, download the latest ECS HDFS Client by browsing to https://support.emc.com and searching for "ecs hdfsclient".

Extract the ECS HDFS Client to the same node running Cloudera Manager. Then run a small HTTP server to make the files available to Cloudera Manager.

```
[root@mycluster1-manager-0 ~]#
unzip hdfsclient*.zip
cd viprfs-client*/parcels/cdh-5.4.x
python -m SimpleHTTPServer 8900
```

Next, add this local repository to Cloudera Manager. Concise steps are below and additional details are available at http://www.cloudera.com/documentation/enterprise/5-4-x/topics/cm_ig_create_local_parcel_repo.html#cmig_topic_21_5.

In Cloudera Manager, click Hosts -> Parcels -> Edit Settings. Add the URL http://localhost:8900/ to the list of Remote Parcel Repository URLs.



Save the settings. You can then click the buttons to Download, Distribute, and Activate the VIPRFS parcel on all hosts.

CONFIGURE ECS HDFS CLIENT ON CDH NODES

The ECS HDFS client requires properties to be defined in the Hadoop core-site.xml file. These properties can be set using Cloudera Manager.

In Cloudera Manager, navigate to HDFS -> Configuration.

Set *Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml* to, replacing *hop-u300-02* with a name identifying this ECS installation and replacing *hop-u300-02-pub-01.solarch.lab.emc.com* with the FQDN or public IP address of all of the ECS nodes:

```
<property>
  <name>fs.viprfs.impl</name>
  <value>com.emc.hadoop.fs.vipr.ViPRFileSystem</value>
</property>
```

```
<property>
  <name>fs.AbstractFileSystem.viprfs.impl</name>
  <value>com.emc.hadoop.fs.vipr.ViPRAbstractFileSystem</value>
</property>
<property>
  <name>fs.vipr.installations</name>
  <value>hop-u300-02</value>
</property>
<property>
  <name>fs.vipr.installation.hop-u300-02.hosts</name>
  <value>hop-u300-02-pub-01.solarch.lab.emc.com,hop-u300-02-pub-02.solarch.lab.emc.com,hop-u300-02-
pub-03.solarch.lab.emc.com,hop-u300-02-pub-04.solarch.lab.emc.com</value>
</property>
<property>
  <name>fs.vipr.installation.hop-u300-02.hosts.resolution</name>
  <value>dynamic</value>
</property>
<property>
  <name>fs.vipr.installation.hop-u300-02.resolution.dynamic.time_to_live_ms</name>
  <value>900000</value>
</property>
<property>
  <name>fs.viprfs.auth.identity_translation</name>
  <value>NONE</value>
</property>
<property>
  <name>fs.viprfs.auth.anonymous_translation</name>
  <value>LOCAL_USER</value>
</property>
```

Additional details are available in the ECS 2.2 Data Access Guide.

Finally, use Cloudera Manager to restart all services.

At this point, all Hadoop applications will have the ability to access ECS using the viprfs prefix, although no applications or services will attempt to yet. So far, all data remains on the DAS HDFS that was created during the CDH installation.

CHECK HDFS ACCESS TO ECS

Once all Hadoop services have started (actually just the client configurations need to be refreshed), ensure that the ECS bucket can be accessed using the Hadoop CLI. The URI will be of the form viprfs://*bucket*.*namespace*.*site*/. In the example commands below, first, a directory listing is attempted. A new bucket will be empty and nothing will be returned. Then an empty file is created, followed by another directory listing.

```
[root@mycluster1-master-0 ~]# hdfs dfs -ls viprfs://mycluster1bucket1.ns1.ecs1/
[root@mycluster1-master-0 ~]# hdfs dfs -touchz
```

```
viprfs://mycluster1bucket1.ns1.ecs1/THIS_IS_mycluster1bucket1
[root@mycluster1-master-0 ~]# hdfs dfs -ls viprfs://mycluster1bucket1.ns1.ecs1/
Found 1 items
-rw-------   1 root          0 2015-08-26 19:05
viprfs://mycluster1bucket1.ns1.ecs1/THIS_IS_mycluster1bucket1
```

CONFIGURE SPARK FOR ECS

In Cloudera Manager, open the Spark configuration, search for "spark-env.sh" and set the three safety valves for spark-env.sh to the following value:

```
export SPARK_DIST_CLASSPATH="${SPARK_DIST_CLASSPATH}:/opt/cloudera/parcels/VIPRFS-2.2.0.0-CDH5.4.x/lib/*"
```

RELOCATE HADOOP SERVICES TO A SUPPLEMENTAL ECS BUCKET

In this configuration, the default file system (fs.defaultFS in core-site.xml) remains as the DAS HDFS NameNode. To access data on ECS, the fully-qualified path with the viprfs prefix must be specified in the application. This section provides details on specific applications.

RELOCATE HIVE TO ECS

To have existing Hive tables on the default file system and other tables on ECS, the only requirement is to use an external table with a fully-qualified path as the table's location. However, to have the default Hive warehouse located on ECS, the configuration steps below can be used. This will direct all new non-external Hive tables to be created on ECS.

In Cloudera Manager, click Hive -> Configuration. In the search box, type "hive-site.xml". You will see six properties labeled *Advanced Configuration Snippet (Safety Valve) for hive-site.xml*. For each of these properties, enter the following value with the appropriate viprfs URL.

```
<property>
  <name>hive.metastore.warehouse.dir</name>
  <value>viprfs://hduser1bucket1.ns1.hop-u300-02/user/hive/warehouse</value>
</property>
<property>
  <name>hive.exec.scratchdir</name>
  <value>viprfs://hduser1bucket1.ns1.hop-u300-02/tmp/hive</value>
</property>
```

For the change to take effect, restart all Hive services.

RELOCATE HBASE TO ECS

Unlike MapReduce and Hive, the HBase service stores its data in a single Hadoop directory tree. The steps below will configure HBase to use ECS for its data.

WARNING!!! DO NOT USE THIS PROCEDURE IF YOU ALREADY HAVE DATA IN HBASE. ANY EXISTING DATA WILL BECOME INACCESSIBLE AND POSSIBLY LOST FOREVER USING THIS PROCEDURE.

First, stop all HBase services.

In Cloudera Manager, click HBase -> Configuration. In the search box, type "hbase-site.xml". You will see six properties labeled *Advanced Configuration Snippet (Safety Valve) for hbase-site.xml*. For each of these properties, enter the following value with the appropriate viprfs URL.

```
<property>
```

```
  <name>hbase.rootdir</name>
  <value>viprfs://hduser1bucket1.ns1.hop-u300-02/hbase</value>
</property>
```

Next, delete the HBase metadata that is stored in Zookeeper.

```
[root@mycluster1-master-0~]# hbase zkcli
[zk] rmr /hbase
[zk] quit
```

Finally, start all HBase services.

CONFIGURE ECS AS THE DEFAULT HADOOP FILE SYSTEM

In this configuration, the default file system (fs.defaultFS in core-site.xml) will be changed to an ECS bucket.

First, stop all Hadoop services except HDFS, YARN, and ZooKeeper.

Next, copy all existing files on the DAS HDFS file system to the ECS bucket. Even for a new installation of Hadoop, there are critical directories that must exist in the default Hadoop file system. Use DistCp to perform the file copy.

```
[hdfs@mycluster1-master-0~]$ hadoop distcp -skipcrccheck -update -pugp -i /
viprfs://cdh6defaultfs.ns1.hop-u300-02/
```

Next, use Cloudera Manager and add the following property to the core-site.xml Safety Valve in the HDFS service.

```
<property>
    <name>fs.defaultFS</name>
    <value>viprfs://cdh6defaultfs.ns1.hop-u300-02/</value>
</property>
```

Finally, use Cloudera Manager to restart all Hadoop services and deploy the client config.

INSTALLATION OVERVIEW (KERBEROS)

This section is intended for the following deployment scenario:

- An MIT KDC server will be used to authenticate all Hadoop services and built-in accounts such as hdfs.
- A Windows 2008 R2 Active Directory will be used to authenticate all other user accounts.
- The MIT KDC server will delegate user authentication requests to Active Directory.

The installation of Cloudera CDH in an ECS environment with Kerberos authentication is composed of the following general steps.

1. Deploy a non-Kerberos Hadoop cluster with ECS
2. Deploy an MIT KDC server.
3. Configure Active Directory.
4. Enable Kerberos using Cloudera Manager.
5. Configure ECS for Kerberos and Active Directory.

6. Create users and buckets on ECS.

7. Reconfigure the ECS client on the CDH nodes for Kerberos.

In the subsequent sections, the environment-specific settings listed below should be changed to match the desired target environment.

Table 3 Kerberos-related Example Environment

DESCRIPTION	EXAMPLE VALUE
AD (Active Directory) Domain	solarch.local
AD domain controller host name	dc-01.solarch.local
AD Kerberos Realm	SOLARCH.LOCAL
MIT KDC Kerberos Realm	KR.SOLARCH.LOCAL
MIT KDC host name	kdc-1.solarch.local
Kerberos Encryption Types	aes256-cts-hmac-sha1-96

INSTALLATION PROCEDURE (KERBEROS)

DEPLOY A NON-KERBEROS CLUSTER WITH ECS

Due to the significant complexity of a Kerberos environment, it is highly recommended to get a complete environment running without Kerberos prior to enabling it.

Perform all steps in the following sub-sections of **Installation Procedure (Non-Kerberos)**:

- Deploy Cloudera CDH
- Configure ECS
- Deploy ECS Client on CDH Nodes
- Configure ECS Client on CDH Nodes
- Check HDFS Access to ECS

DEPLOY AN MIT KDC SERVER

A concise list of steps to deploy an MIT KDC is shown below. For more details, including a script to automate much of this can be found at http://blog.cloudera.com/blog/2015/03/how-to-quickly-configure-kerberos-for-your-apache-hadoop-cluster/.

On the server that will be the MIT KDC server:

```
[root@kdc-1 ~]#
yum install krb5-server krb5-libs krb5-auth-dialog krb5-workstation
```

Create the file /etc/krb5.conf with contents shown:

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log
[libdefaults]
 default_realm = KR.SOLARCH.LOCAL
 dns_lookup_realm = false
 dns_lookup_kdc = false
```

```
 ticket_lifetime = 1000d
 renew_lifetime = 1000d
 forwardable = true
 default_tkt_enctypes = aes256-cts-hmac-sha1-96
 default_tgs_enctypes = aes256-cts-hmac-sha1-96
[realms]
 KR.SOLARCH.LOCAL = {
  kdc = kdc-1.solarch.local
  admin_server = kdc-1.solarch.local
 }
 SOLARCH.LOCAL = {
  kdc = dc-01.solarch.local
  admin_server = dc-01.solarch.local
 }
```

Create the file /var/kerberos/krb5kdc/kdc.conf with contents shown:

```
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88
 max_life = 1d
 max_renewable_life = 7d
[realms]
 KR.SOLARCH.LOCAL = {
  acl_file = /var/kerberos/krb5kdc/kadm5.acl
  dict_file = /usr/share/dict/words
  admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
  supported_enctypes = aes256-cts-hmac-sha1-96:normal
 }
```

Create the file /var/kerberos/krb5kdc/kadm5.acl with contents shown:

```
*/admin@KR.SOLARCH.LOCAL   *
```

Initialize the MIT KDC.

```
[root@kdc-1 ~]#
mv /dev/random /dev/random.orig
ln -s /dev/urandom /dev/random
kdb5_util create -s
KDC database master password: <enter new KDC master password>
service krb5kdc start
```

```
service kadmin start
chkconfig krb5kdc on
chkconfig kadmin on
kadmin.local -q "addprinc admin/admin"
admin/admin password: <enter new admin/admin password>
```

Test admin/admin access to the MIT KDC.

```
[root@kdc-1 ~]#
kadmin -p admin/admin
Authenticating as principal admin/admin with password.
Password for admin/admin@KR.SOLARCH.LOCAL: <enter admin/admin password>
kadmin:  listprincs
K/M@KR.SOLARCH.LOCAL
admin/admin@KR.SOLARCH.LOCAL
kadmin/admin@KR.SOLARCH.LOCAL
kadmin/changepw@KR.SOLARCH.LOCAL
kadmin/kdc-1.solarch.local@KR.SOLARCH.LOCAL
krbtgt/KR.SOLARCH.LOCAL@KR.SOLARCH.LOCAL
kadmin:  exit
```

CONFIGURE ACTIVE DIRECTORY

This assumes that an Active Directory domain has already been created and is functioning normally.

Let Active Directory know about the MIT KDC.

```
C:\Windows\system32>
ksetup /addkdc KR.SOLARCH.LOCAL kdc-1.solarch.local
```

Create an account for the MIT KDC to authenticate users in AD.

```
C:\Windows\system32>
netdom trust KR.SOLARCH.LOCAL /domain:solarch.local /add /realm /passwordt:<enter new trust password>
```

Set the Kerberos encryption type used by this account.

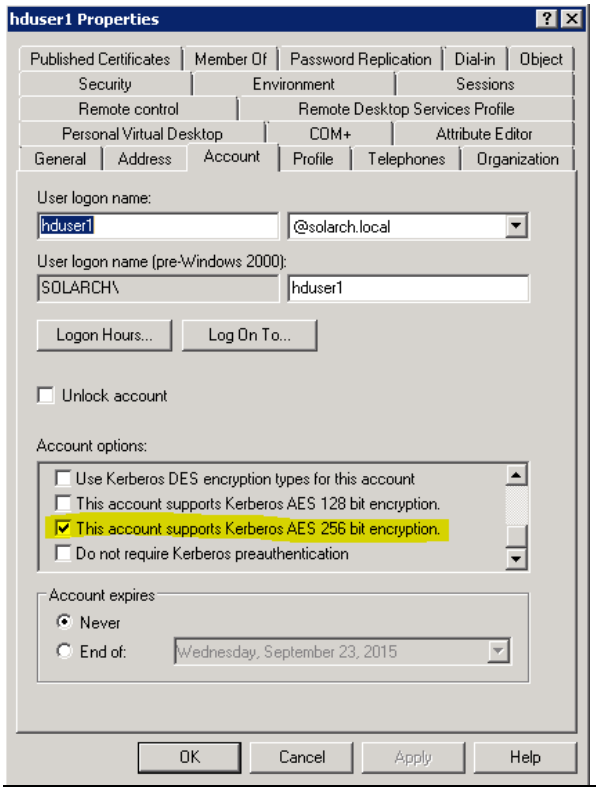
```
C:\Windows\system32>
ksetup /SetEncTypeAttr KR.SOLARCH.LOCAL AES256-CTS-HMAC-SHA1-96
```

Create the principal on MIT KDC to access the AD KDC.

```
[root@kdc-1 ~]# kadmin.local
kadmin.local: addprinc -e "aes256-cts-hmac-sha1-96:normal" krbtgt/KR.SOLARCH.LOCAL@SOLARCH.LOCAL
admin/admin password: <enter trust password>
```

ECS uses an AD account to lookup users and group. Create a normal AD user that will be used by ECS for this purpose. It is assumed that this user will be named "Manager".

Create or Enable User Accounts in Active Directory. Unless enabled by AD policies, each user account needs to be enabled to authenticate with Kerberos. This is done by checking the box "This account supports Kerberos AES 256 bit encryption" in the AD User Account properties.



CONFIRM KERBEROS AUTHENTICATION OF AD ACCOUNTS

Use the steps below to confirm that an AD user can be authenticated using Kerberos. Additionally, ensure that the encryption types match the expected value (aes256-cts-hmac-sha1-96 in this case).

```
[hduser1@kdc-1 ~]# kinit hduser1@SOLARCH.LOCAL
Password for hduser1@SOLARCH.LOCAL: <enter password for hduser1 as defined in AD>
[hduser1@kdc-1 ~]# klist -e
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hduser1@SOLARCH.LOCAL
Valid starting     Expires            Service principal
08/24/15 22:35:26  08/25/15 08:35:18  krbtgt/SOLARCH.LOCAL@SOLARCH.LOCAL
        renew until 08/31/15 22:35:26, Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-
sha1-96
```

ENABLE KERBEROS USING CLOUDERA MANAGER

Follow the procedure at http://www.cloudera.com/documentation/enterprise/5-4-x/topics/cm_sg_intro_kerb.html.

The settings below have been tested successfully.

Table 4 Kerberos Settings in Cloudera Manager

PROPERTY	VALUE
KDC Type	MIT KDC
KDC Server Host	kdc-1.solarch.local
Kerberos Security Realm	KR.SOLARCH.LOCAL
Kerberos Encryption Types	aes256-cts-hmac-sha1-96

Advanced Configuration Snippet (Safety Valve) for remaining krb5.conf	SOLARCH.LOCAL = {
	kdc = dc-01.solarch.local
	admin_server = dc-01.solarch.local
	}

Additional Rules	RULE:[1:$1@$0](.*@KR.SOLARCH.LOCAL)s/@.*// RULE:[1:$1@$0](.*@SOLARCH.LOCAL)s/@.*//

CONFIRM ACCESS TO DAS HDFS USING KERBEROS

Use the steps below to authenticate as an AD user and perform a directory listing of the DAS HDFS.

```
[hduser1@mycluster1-master-0 ~]# kinit hduser1@SOLARCH.LOCAL
Password for hduser1@SOLARCH.LOCAL: <enter password for hduser1 as defined in AD>
[hduser1@mycluster1-master-0 ~]# hdfs dfs -ls /
```

CONFIGURE ECS FOR KERBEROS

Complete details for this procedure can be found in the ECS 2.2 Data Access Guide. A concise list of steps follows.

To enable Kerberos on the ECS nodes, the Ansible automation tool is used. Follow the steps below to install Ansible and prepare it to configure ECS for Kerberos.

```
[root@kdc-1 ~]#
yum-config-manager --enable base
yum-config-manager --enable extras
yum-config-manager --enable updates
yum clean all
yum install epel-release
yum install ansible
yum install sshpass
cd viprfs-client-*/playbooks
ansible-galaxy install -r requirements.txt -f
mkdir mycluster1
cp samples/* mycluster1/
cd mycluster1/
unzip /path/to/UnlimitedJCEPolicyJDK7.zip
cp /etc/krb5.conf .
```

Create a file named **inventory.txt** with the following contents:

```
[data_nodes]
```

```
ecs1-[1:4].solarch.local   ansible_ssh_pass=ChangeMe
[kdc]
kdc-1.solarch.local   ansible_ssh_pass=ChangeMe
```

Edit the file **generate-vipr-keytabs.yml** to contain the following:

```
---
###
# 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@KR.SOLARCH.LOCAL
          keytab: keytabs/_HOST@KR.SOLARCH.LOCAL.keytab
```

Edit the file **setup-vipr-kerberos.yml** to contain the following:

```
---
###
# 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@KR.SOLARCH.LOCAL
         keytab: keytabs/_HOST@KR.SOLARCH.LOCAL.keytab

     - role: vipr_jce_config
```

```
    jce_policy:
      name: unlimited
      src: UnlimitedJCEPolicy/
```

Run the Ansible playbooks.

```
[root@kdc-1 mycluster1]#
ansible-playbook -v -i inventory.txt generate-vipr-keytabs.yml
ansible-playbook -v -i inventory.txt setup-vipr-kerberos.yml
```

Confirm that vipr service principals were generated on the KDC.

```
[root@kdc-1 mycluster1]#
kadmin.local -q "list_principals" | grep vipr
vipr/ecs1-1.solarch.local@KR.SOLARCH.LOCAL
vipr/ecs1-2.solarch.local@KR.SOLARCH.LOCAL
vipr/ecs1-3.solarch.local@KR.SOLARCH.LOCAL
vipr/ecs1-4.solarch.local@KR.SOLARCH.LOCAL
```

ADD ACTIVE DIRECTORY TO ECS

Complete details for this procedure can be found in the ECS 2.2 Data Access Guide. A concise list of steps follows.

In the ECS UI, click Manager -> New Authentication Provider and enter the following values.

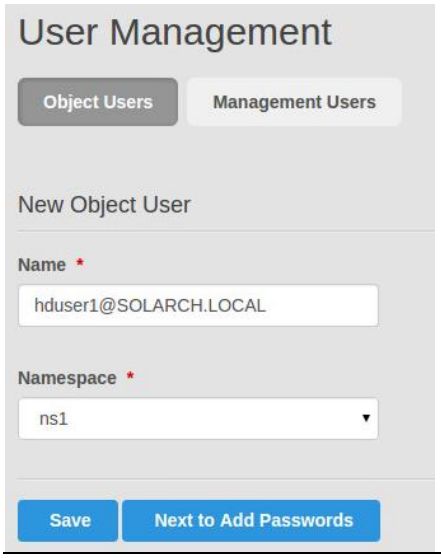
Table 5 ECS Authentication Provider Settings

PROPERTY	VALUE
Name	solarch.local
Description	solarch.local AD
Type	Active Directory
Domains	solarch.local
Server URLs	ldap://dc-01.solarch.local
Manager DN	CN=Manager,CN=Users,DC=solarch,DC=local
Search Scope	One Level
Search Base	DC=solarch,DC=local
Search Filter	userPrincipalName=%u

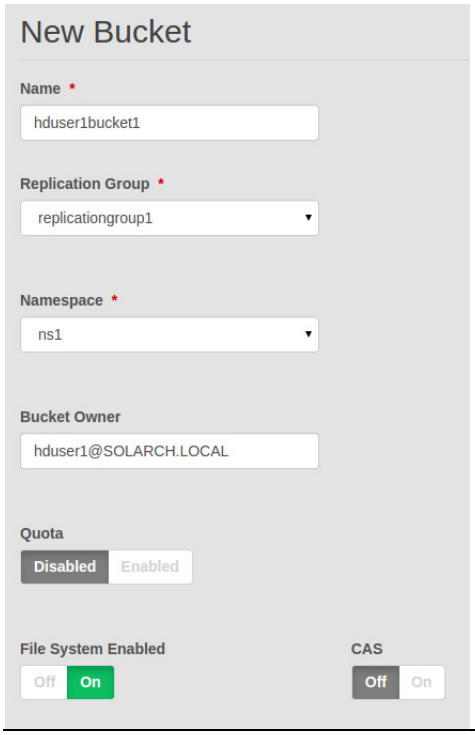
CREATE ECS USER AND BUCKET

ECS refers to AD users using the format user@REALM. For instance, the user named "hduser1" in the AD Users group of the solarch.local AD domain will be referred to as hduser1@SOLARCH.LOCAL.

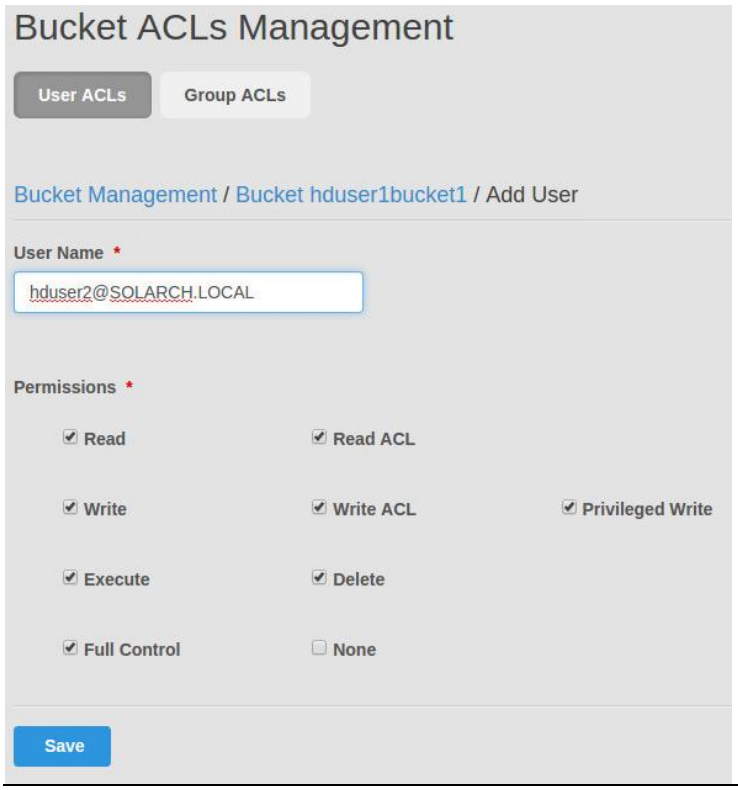
Create an ECS object user account for each AD account that will access ECS.



Next, create a bucket owned by the new object user. To allow HDFS access, you must set *File System Enabled* to On.



Next, edit the bucket ACL to allow the appropriate AD users. To ensure a secure bucket, remove all permissions from all_users and public.



Next, you must upload user metadata to the ECS bucket. Carefully follow the procedure in the **Secure the ECS bucket using metadata** and **Load metadata values to ECS using the Management API** sections of the ECS 2.2 Data Access Guide.

RECONFIGURE THE ECS CLIENT ON CDH NODES FOR KERBEROS

In Cloudera Manager, navigate to HDFS -> Configuration.

Add or replace the properties shown below in *Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml.* Do not change the other properties that were added during the initial non-Kerberos configuration.:

```
<property>
  <name>fs.viprfs.auth.identity_translation</name>
  <value>CURRENT_USER_REALM</value>
</property>
<property>
  <name>viprfs.security.principal</name>
  <value>vipr/_HOST@KR.SOLARCH.LOCAL</value>
</property>
```

Finally, use Cloudera Manager to restart all services.

CONFIRM ACCESS TO ECS USING KERBEROS

Use the steps below to authenticate as the AD user hduser1 and perform a directory listing of the secure ECS bucket.

```
[hduser1@mycluster1-master-0 ~]# kinit hduser1@SOLARCH.LOCAL
Password for hduser1@SOLARCH.LOCAL: <enter password for hduser1 as defined in AD>
```

```
[hduser1@mycluster1-master-0 ~]# hdfs dfs -ls viprfs://hduser1bucket1.ns1.ecs1/
[hduser1@mycluster1-master-0 ~]# hdfs dfs -touchz
viprfs://hduser1bucket1.ns1.ecs1/THIS_IS_hduser1bucket1
[hduser1@mycluster1-master-0 ~]# hdfs dfs -ls viprfs://hduser1bucket1.ns1.ecs1/
Found 1 items
-rw-------   1 hduser1          0 2015-08-26 19:05
viprfs://hduser1bucket1.ns1.ecs1/THIS_IS_hduser1bucket1
```

TESTING THE HADOOP INSTALLATION

To ensure that all required Hadoop components are functional, it is recommended that the functional and performance tests as documented in the related validation brief (**EMC Technical Validation of Cloudera CDH with ECS**) be performed.

TROUBLESHOOTING

ENTER DOCKER CONTAINERS

The ECS system is built using Docker containers. Most command-line operations for ECS will be performed within the object-main Docker container. To "enter" this container, SSH to the Docker container host and run the following command.

```
ecs1-1:~ # docker exec -it object-main bash
ecs1-1:/ # # Enter your command here.
ecs1-1:/ # exit
```

RESTART ECS STORAGE OS DATA SERVICES

Use the command below to restart the Storage OS Data Services on all ECS nodes. Note that it may take up to 10 minutes after this script runs for the services to be ready to accept requests. This assumes there is a file named MACHINES with the IP address or FQDN of each ECS node, one per line.

```
ecs1-1:~ # cat MACHINES | grep -v "#" | xargs -n 1 -P 0 -i ssh root@{} docker exec object-main service
storageos-dataservice restart
```

UPDATE FILES ON ECS

Use the Bash script below to copy configuration files or patches to each ECS node. Before running the script, create a directory called *template* with the desired directory structure and files.

```
#!/bin/bash
HOSTS="ecs1-1 ecs1-2 ecs1-3 ecs1-4"
tar -cvf template.tar -C template/ .
tar -tvf template.tar
for h in $HOSTS
        do
        ssh root@$h docker exec -i object-main tar --overwrite -xvf - < template.tar
        done
```

ECS LOG FILES

The following log files contain useful information regarding HDFS access on ECS. These are in the Docker object-main container.

/opt/storageos/logs/hdfssvc-error.log
/opt/storageos/logs/hdfssvc.log

S3 BROWSER

In addition to accessing the ECS bucket using HDFS, an S3-compatible browser may also be used to edit bucket objects and object ACLs. For instance, S3 Browser (http://s3browser.com/) may be used. To configure S3 Browser to access ECS, create an account with the following settings.

Table 6 S3 Browser Settings

FIELD	VALUE
Storage Type	S3 Compatible Storage
REST Endpoint	*ecs_endpoint_fqdn_or_ip*:9021
Access Key ID	(This should match the name of the object user defined in ECS)
Secret Access Key	(This should match the Object Access S3 password defined in ECS)

HADOOP DEBUG LOGGING

To troubleshoot issues with Hadoop CLI commands such as "hdfs dfs -ls", use the following commands.

```
[root@mycluster1-master-0~]#

export HADOOP_ROOT_LOGGER="TRACE,console"

export HADOOP_OPTS="-Dsun.security.krb5.debug=true"

hdfs dfs -ls $FS/ 2>&1 | tee log
```

For more details, refer to https://community.emc.com/docs/DOC-45698.

PERFORMANCE OPTIMIZATION

The table below lists the most commonly used Hadoop parameters and recommended initial values for good MapReduce performance.

Table 7 Hadoop performance-related parameters for MapReduce

LOCATION	PROPERTY	VALUE	NOTES
yarn-site	yarn.nodemanager.resource.cpu-vcores	(Number of CPU cores - 1)	Set this to the number of CPU cores on each of your Node Manager servers minus 1. By default, this is the maximum number of map or reduce tasks that will run on each Node Manager. This is a per-NodeManager setting.
yarn-site	yarn.nodemanager.resource.memory-mb	(Physical or Virtual Machine RAM minus 12 GiB)	In general, leave 12-16 GiB of RAM for use by the OS and Hadoop services. The remainder can be use by the map and reduce tasks. This is a per-NodeManager setting.
yarn-site	yarn.resourcemanager.scheduler.class	org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairSchedule	Although the default Capacity Scheduler has many more features and is more commonly used, the Fair Scheduler does a better job of running simple MapReduce jobs such as Terasort on remote Hadoop storage.
yarn-site	yarn.scheduler.maximum-allocation-mb	65536	In case the default is too small, allow a large memory allocation by the map and reduce tasks.
yarn-site	yarn.scheduler.minimum-allocation-mb	(yarn.nodemanager.resource.memory-mb / yarn.nodemanager.resource.cpu-vcores)	To achieve an even balance of CPU and memory usage for a homogeneous workload of MapReduce tasks, use this calculated value. Increase if memory errors are encountered.
yarn-site	yarn.scheduler.increment-allocation-mb	(yarn.scheduler.minimum-allocation-mb)	When using the Fair Scheduler, set this to the same value as the minimum.
mapred-site	mapreduce.map.memory.mb	(yarn.scheduler.minimum-allocation-mb)	This is a per-job setting but the default value can be specified globally.
mapred-site	mapreduce.reduce.memory.mb	(yarn.scheduler.minimum-allocation-mb)	This is a per-job setting but the default value can be specified globally.
mapred-site	mapreduce.map.java.opts	(mapreduce.map.memory.mb * 0.75 in format -Xmx1536m)	This is a per-job setting but the default value can be specified globally.
mapred-site	mapreduce.reduce.java.opts	(mapreduce.reduce.memory.mb * 0.75 in format -Xmx1536m)	This is a per-job setting but the default value can be specified globally.

For additional details, see https://community.emc.com/docs/DOC-41843.