Cloudera Runtime 7.1.6

Configuring Ozone security

Date published: 2020-04-24
Date modified: 2023-08-31

https://docs.cloudera.com/

Legal Notice

Contents

Using Ranger with Ozone

You can use Apache Ranger to secure access to your Ozone data. For Ozone to work with Ranger, you can use Cloudera Manager and enable the Ranger service instance with which you want to use Ozone.

Procedure

1. Go to the Ozone service.
2. Click the Configuration tab.
3. Search for ranger_service.
4. Select the RANGER service checkbox.
5. Enter a Reason for Change, and then click Save Changes to save the property changes.
6. Restart the cluster.

What to do next

Set up policies in Ranger for the users to have the right access permissions to the various Ozone objects such as buckets and volumes.

When using Ranger to provide a particular user with read/write permissions to a specific bucket, you must configure a separate policy for the user to have read access to the volume in addition to policies configured for the bucket.

For example,

- user/group -> Allow Read on Volume=testvol, Bucket=testbucket1.
- user/group -> Allow All (including delete on) on Volume=testvol, Bucket=testbucket1, Keys=*

To disallow users to delete volumes or buckets:

- user/group -> Allow Read on Volume=testvol, Bucket=testbucket1. Deny Delete on Volume=testvol, Bucket=testbucket1
- user/group -> Allow All (including delete on) on Volume=testvol, Bucket=testbucket1, Keys=*

Further, if Infra-Solr is managed by Ranger, the Ozone Manager principal (om) must have access to Infra-Solr. You can provide access to the Ozone Manager principal by adding om to the RANGER_AUDITS_COLLECTION Solr collection for cm_solr on Ranger.

Related Information

Apache Ranger Authorization

Kerberos configuration for Ozone

Ozone depends on Kerberos to make the clusters secure. To enable security in an Ozone cluster, you must set the parameters ozone.security.enabled to true and hadoop.security.authentication to kerberos.

Security tokens in Ozone

Ozone issues delegation and block tokens to users or client applications authenticated with the help of Kerberos such that they can perform specified operations against the cluster, as if they have kerberos tickets.

Delegation tokens

Delegation tokens allow a user or client application to impersonate a users kerberos credentials. This token is based on verification of kerberos identity and is issued by the Ozone Manager. Delegation tokens are enabled by default when security is enabled.

Block tokens

Block tokens allow a user or client application to read or write a block. This ensures that only users or client applications with the required permissions can read or write to blocks in DataNodes. Block tokens are issued to authenticated clients and signed by Ozone Manager. They are validated by the DataNode using the certificate or public key of the issuer (Ozone Manager).

S3 tokens

Users or client applications accessing Ozone using S3 APIs with S3 credential tokens. These tokens are also enabled by default when security is enabled.

Note: Ozone supports the AWS Signature Version 4 protocol.

Kerberos principal and keytab properties for Ozone service daemons

For the Kerberos-authenticated users or client applications to access Ozone, each of the Ozone components requires a Kerberos service principal name and a corresponding kerberos keytab file. You must set the corresponding in ozone-site.xml.

The following are the properties for the Kerberos service principal and the keytab file that you must set for the different Ozone components:

Storage Container Manager (SCM) properties

Property	Description
hdds.scm.kerberos.principal	The SCM service principal. You can specify this value, for example, in the following format: `scm/_HOST@REALM.COM`
hdds.scm.kerberos.keytab.file	The keytab file that the SCM daemon uses to log in as its service principal.
hdds.scm.http.auth.kerberos.principal	The service principal of the SCM HTTP server.
hdds.scm.http.auth.kerberos.keytab	The keytab file that the SCM HTTP server uses to log in as its service principal.

Ozone Manager (OM) properties

Property	Description
ozone.om.kerberos.principal	The Ozone Manager service principal. You can specify this value, for example, in the following format: `om/_HOST@REALM.COM`

Property	Description
ozone.om.kerberos.keytab.file	The keytab file that the Ozone Manager daemon uses to log in as its service principal.
ozone.om.http.auth.kerberos.principal	The service principal of the Ozone Manager HTTP server.
ozone.om.http.auth.kerberos.keytab	The keytab file that the Ozone Manager HTTP server uses to log in as its service principal.

S3 Gateway properties

Property	Description
ozone.s3g.kerberos.principal	The S3 Gateway principal. You can specify this value, for example, in the following format: `s3g/_HOST@REALM.COM`
ozone.s3g.kerberos.keytab.file	The keytab file used by S3Gateway daemon to login as its service principal. The principal name is configured with ozone.s3g.kerberos.principal.
ozone.s3g.http.auth.kerberos.principal	The server principal used by Ozone S3 Gateway server. This is typically set to HTTP/_HOST@REALM.TLD The SPNEGO server principal begins with the prefix HTTP/ by convention.
ozone.s3g.http.auth.kerberos.keytab	The keytab file used by the S3 Gateway server to login as its service principal.

Recon properties

Property	Description
ozone.recon.kerberos.principal	The Recon web user service principal. You can specify this value, for example, in the following format: `recon/_HOST@REALM.COM`
ozone.recon.kerberos.keytab.file	The keytab file used by the Recon web user.
ozone.recon.http.auth.kerberos.principal	The server principal used by Ozone Recon server. This is typically set to HTTP/_HOST@REALM.TLD The SPNEGO server principal begins with the prefix HTTP/ by convention.
ozone.recon.http.auth.kerberos.keytab	The keytab file for HTTP Kerberos authentication in Recon.

Note: Ozone *does not support* authentication based on Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) for http endpoints in the current CDP release.

Securing DataNodes

You can secure Ozone DataNodes by creating keytab files on each of the DataNodes. You must ensure that certain properties are configured in hdfs-site.xml to provide the Kerberos-authenticated users or client applications with access to the DataNodes.

Configure the following parameters in hdfs-site.xml to enable DataNode access:

Property	Description
dfs.datanode.kerberos.principal	The DataNode service principal. You can specify this value, for example, in the following format: `dn/_HOST@REALM.COM`
dfs.datanode.kerberos.keytab.file	The keytab file that the DataNode daemon uses to log in as its service principal.
hdds.datanode.http.auth.kerberos.principal	The service principal of the DataNode http server.
hdds.datanode.http.auth.kerberos.keytab	The keytab file that the DataNode http server uses to log in as its service principal.

Certificate request and approval

When a DataNode boots up, it creates a private key and sends an DataNode identity certificate request to Storage Container Manager (SCM). If the DataNode has a Kerberos keytab, SCM trusts the Kerberos credentials and automatically issues a certificate.

Configure S3 credentials for working with Ozone

For the users or client applications that use S3 APIs to access Ozone buckets, Ozone provides the AWS access key ID and AWS secret key. You can add the access key ID and secret key in the AWS config file for Ozone to ensure that a particular user or client application can get automatic access to the Ozone buckets.

Before you begin

The user or the client application accessing Ozone must be authenticated against your cluster's KDC.

Procedure

1. Display the access key ID and the secret key for a particular user or client application.

   ```
   ozone s3 getsecret
   ```

   The command communicates with Ozone, validates the user credentials through Kerberos, and generates the AWS credentials. These credentials are printed on the screen.

   Note: These S3 credentials are like Kerberos passwords that give complete access to the Ozone buckets.

2. Add the generated credentials to the AWS config file.

   The following example shows how you can add the credentials.

   ```
   aws configure set default.s3.signature_version s3v4
   aws configure set aws_access_key_id ${accessId}
   aws configure set aws_secret_access_key ${secret}
   aws configure set region us-west-1
   ```

Configuring custom Kerberos principal for Ozone

The Kerberos principal for Ozone is configured by default to use the same service principal as the default process user. However, you can change the default setting by providing a custom principal in Cloudera Manager.

About this task

Note: When you change the principal name of Ozone Manager and the Ranger plugin is enabled, you must ensure that you update the required safety valve in Ranger as well to notify the Ranger plugin about the Ozone Manager principal change. Otherwise, Ozone Manager cannot communicate with the Ranger plugin and all authorization requests fail.

Procedure

1. Go to the Cloudera Manager *Ozone service* Configuration tab.
2. Search for *Kerberos principal*.
3. Enter the custom Kerberos principals for the various Ozone roles.
4. Click Save Changes.
5. Restart the Ozone service.
6. Login to the Cloudera Manager Ranger service Ranger Web UI page using administrator credentials.
7. Edit cm_ozone to add or update the following key-value configuration parameters:
   a. tag.download.auth.users = *[***custom_ozone_manager_role_principal***]*
   b. policy.download.auth.users = *[***custom_ozone_manager_role_principal***]*

   Tip: It is recommended that you configure the conf/ranger-admin-site.xml_role_safety_valve property with ranger.plugins.ozone.serviceuser=*[***custom_ozone_manager_role_principal***]* when you set up a cluster using a template.

   For example, if the kerberos principal name of Ozone Manager is updated to ozone-om, then configure the conf/ranger-admin-site.xml_role_safety_valve property in Ranger to ranger.plugins.ozone.serviceuser=ozone-om.

Configuring Transparent Data Encryption for Ozone

Transparent Data Encryption (TDE) allows data on the disks to be encrypted-at-rest and automatically decrypted during access. For Ozone, you can enable TDE at the bucket-level.

Before you begin

- The Key Management Server must be installed and running. Ozone uses the same Key Management Server as HDFS.
- To use TDE, you (admin) must setup a Key Management Server and provide that URI to Ozone/HDFS. You can provide this configuration via hdfs-site.xml.

Property	Value
hadoop.security.key.provider.path	KMS uri.e.g. kms://http@kms-host:9600/kms

- If you are using Ranger Key Management Server, then you must ensure that the key management admin user has assigned the om service user with the Get Metadata and Generate EEK permissions.

  Further, you must ensure that any user attempting to create an encryption key has the required permissions in Ranger.

Procedure

1. Create a bucket encryption key.

```
hadoop key create enck1
```

   This command creates an encryption key for the bucket you want to protect. After the key is created, Ozone can use that key when you are reading and writing data into a bucket.

2. Assign the encryption key to a bucket.

   The following example shows how you can assign the key enck1 to the bucket encbucket1:

```
ozone sh bucket create -k enck1 /vol/encbucket1
```

   After you run this command, all data written to encbucket1 will be encrypted using enck1. During the read process, the client applications interact with the Key Management Server to read the key and decrypt it.

   The encryption of data is completely transparent to users and client applications.

Using Transparent Data Encryption from S3G

There are two ways to create an encrypted bucket that can be accessed via S3 Gateway.

Procedure

1. Create a bucket using shell under “/s3v” volume.

```
ozone sh bucket create -k enck1 /s3v/encbucket1
```

2. Create a link to an encrypted bucket under “/s3v” volume.

```
ozone sh bucket create -k enck1 /vol/encbucket1
ozone sh bucket link  /vol/encbucket1 /s3v/linkencryptedbucket
```

   Note: You cannot create an encrypted bucket using S3 APIs. It must be done using Ozone shell commands as shown above. After creating an encrypted bucket, all the keys added to this bucket using s3g will be encrypted.

   In non-secure mode, the user running the S3Gateway daemon process is the proxy user, while in secure mode the S3Gateway Kerberos principal (ozone.s3g.kerberos.principal) is the proxy user. The S3Gateway proxies all the users accessing the encrypted buckets to decrypt the key. Therefore, on security enabled cluster, during S3Gateway server startup logins using configured ozone.s3g.kerberos.keytab.file and ozone.s3g.kerberos.principal.

   You must add these two configurations to the kms-site.xml to allow the S3Gateway principal to act as a proxy for other users.In this example, “ozone.s3g.kerberos.principal” is assumed to be “s3g”.

```
<property>
<name>hadoop.kms.proxyuser.s3g.users</name>
<value>user1,user2,user3</value>
<description>
        Here the value can be all the S3G accesskey ids accessing Ozone S3

        or set to '*' to allow all the accesskey ids.
</description>
</property>
```

```
<property>
```

```
  <name>hadoop.kms.proxyuser.s3g.hosts</name>
  <value>s3g-host1.com</value>
  <description>
         This is the host where the S3Gateway is running. Set this to '*'
 to allow
         requests from any hosts to be proxied.
  </description>
</property>
```

Note: If Ranger authorization is enabled for KMS, then decrypt key permission should be given to access key id user(currently access key is kerberos principal) to decrypt the encrypted key to read/write a key in the encrypted bucket.

Configuring TLS/SSL encryption manually for Ozone

You can use Cloudera Manager to configure TLS/SSL encryption for all the Ozone role instances such as Ozone Manager, Storage Container Manager, DataNode, S3 Gateway and Recon. Cloudera recommends that you configure TLS/SSL for all the Ozone role instances to avoid errors when one role instance tries to connect with another.

Before you begin

You must ensure that your Ozone deployment meets the following requirements:

- The keystores containing certificates that are bound to the proper domain names are accessible on all the hosts on which at least one Ozone role instance is running.
- The hdfs user has read permissions to the keystore files for Ozone.
- You must specify absolute paths to the keystore file. These settings apply to all hosts on which the various Ozone role instances run. Therefore, the paths that you specify must be valid on all the hosts. In addition, the keystore file names for Ozone must be the same on all hosts.

  Consider an example where you have separate certificates for the Ozone role instances on hosts node1.exampl e.com and node2.example.com, and you have chosen to store the certificates in files with names ozone-node1.jks and ozone-node2.jks respectively. When deploying these keystores, you must provide them both with the same name on the target host, for example, ozone.jks.

Procedure

1. In Cloudera Manager, go to Ozone > Configuration .
2. Search for *tls/ssl*.

3. Enter the TLS/SSL properties for the various Ozone roles.

   Provide the values for the properties corresponding to the following fields on the Ozone configuration page:

   - Enable TLS/SSL for Ozone Manager
   - Ozone Manager TLS/SSL Server JKS Keystore File Location
   - Ozone Manager TLS/SSL Server JKS Keystore File Password
   - Ozone Manager TLS/SSL Server JKS Keystore Key Password
   - Enable TLS/SSL for Storage Container Manager
   - Storage Container Manager TLS/SSL Server JKS Keystore File Location
   - Storage Container Manager TLS/SSL Server JKS Keystore File Password
   - Storage Container Manager TLS/SSL Server JKS Keystore Key Password
   - Enable TLS/SSL for Ozone DataNode
   - Ozone DataNode TLS/SSL Server JKS Keystore File Location
   - Ozone DataNode TLS/SSL Server JKS Keystore File Password
   - Ozone DataNode TLS/SSL Server JKS Keystore Key Password
   - Enable TLS/SSL for Ozone Recon
   - Ozone Recon TLS/SSL Server JKS Keystore File Location
   - Ozone Recon TLS/SSL Server JKS Keystore File Password
   - Ozone Recon TLS/SSL Server JKS Keystore Key Password
   - Enable TLS/SSL for S3 Gateway
   - S3 Gateway TLS/SSL Server JKS Keystore File Location
   - S3 Gateway TLS/SSL Server JKS Keystore File Password
   - S3 Gateway TLS/SSL Server JKS Keystore Key Password
   - CA File Path

   Note: Ensure that you configure the ozone.prometheus.ca.file property when configuring TLS/SSL encryption for Ozone. If the location of the root CA certificate is not specified, the location of the auto-TLS CA certificate is considered as the default value for this property.

4. Click Save Changes and restart the Ozone service.

Configuration for enabling mTLS in Ozone

You can enable mutual TLS (mTLS) over gRPC for secure communication among the different elements of an Ozone cluster, such as an within an Ozone Manager (OM) High Availability (HA) deployment or among the DataNodes.

To enable gRPC, you must ensure that the value of the hdds.grpc.tls.enabled configuration property is set to true.

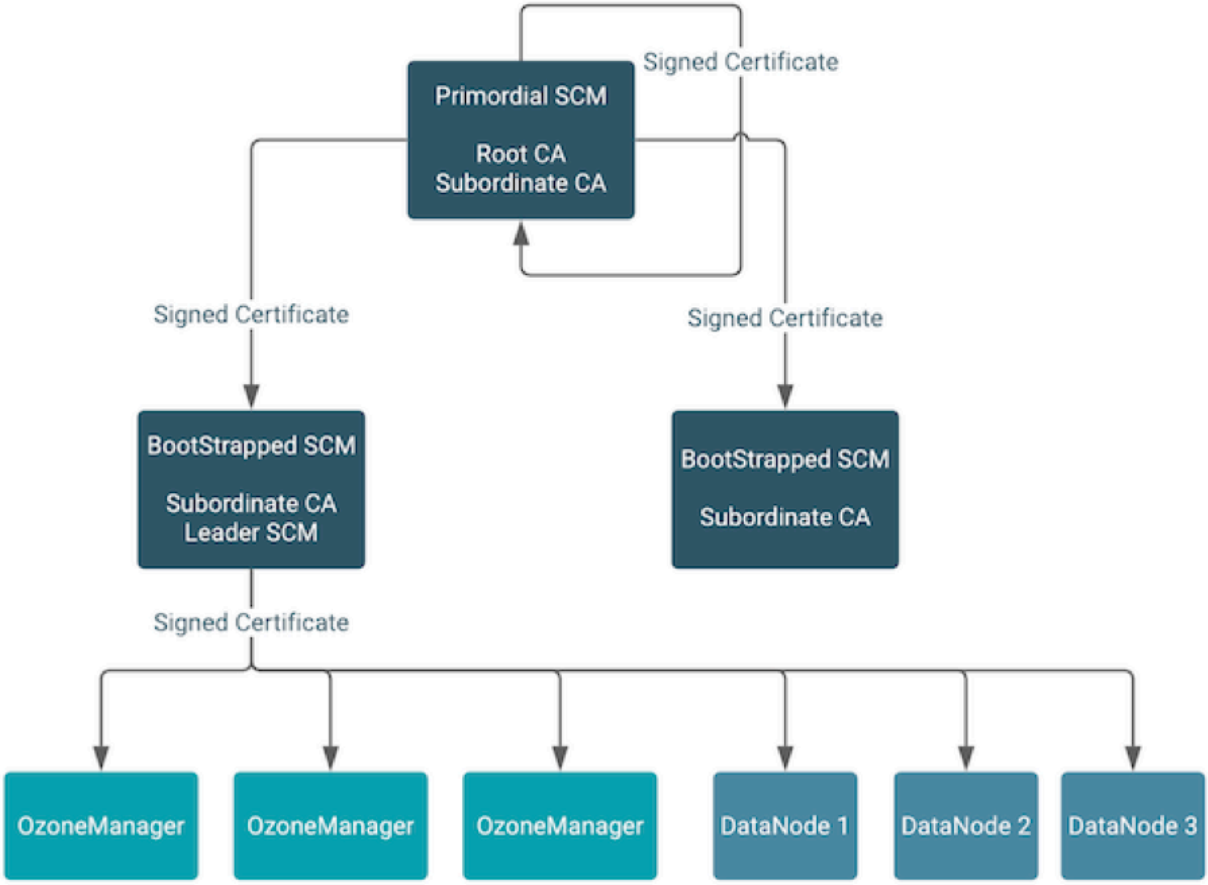
Important:

- The hdds.grpc.tls.enabled property will not be effective unless the value of the ozone.security.enabled is also set to true.
- gRPC TLS configuration for Ozone is supported only on new CDP 7.1.7 clusters and not on clusters upgraded from CDP 7.1.6 to CDP 7.1.7. If you want to enable gRPC TLS on the upgraded CDP 7.1.7 clusters, you must contact Cloudera Support for more information.

Configuring security for Storage Container Managers in High Availability

Configuring security and issuing certificates to Storage Container Managers (SCMs) along with Ozone Managers (OMs) and the DataNodes ensures secured communication in the Ozone cluster.

How the SCM issues certificates



The primordial SCM in the HA configuration starts a root Certificate Authority (CA) with self-signed certificates. The primordial SCM issues signed certificates to itself and the other bootstrapped SCMs in the HA configuration. The primordial SCM also has a subordinate CA with a signed certificate from the root CA.

When an SCM is bootstrapped, it receives a signed certificate from the primordial SCM and starts a subordinate CA of its own. The subordinate CA of any SCM that becomes the leader in the HA configuration issues signed certificates to Ozone Managers and the DataNodes in the Ozone cluster.

How Ozone renews certificates

The SSL certificates issued by the subordinate CAs within SCMs to other services expire after one year by default (hdds.x509.default.duration), and are automatically renewed once the certificate lifetime enters the renewal grace period (hdds.x509.renew.grace.duration). This renewal process is fully automated. It does not require any manual intervention or restart of the service to take effect, as the certificates are applied automatically to any SSL handshake happening after the renewal has happened, while the change in the certificate does not affect any connections that are already established and remain open even after the old certificate expires.

Note: SCM certificates are not renewed automatically, they have a 5 year lifetime period (hdds.x509.max.duration) from the initial security bootstrap of the cluster, and they require a new security bootstrap to be renewed. The Ozone team is working on to alleviate this by automating the renewal process of the SCM certificates in a future Ozone release.

Considerations for enabling SCM HA security

There are certain factors that you must consider when enabling security for Storage Container Managers (SCMs) in High Availability (HA).

- Ensure that you have enabled SCM HA by taking the required factors into consideration.
- To configure SCM HA security, ensure that the ozone.scm.ratis.enable property is set to true.
- To enable gRPC TLS for the interactions among different elements of the Ozone cluster, such as among the Ozone Manager (OM) nodes or the SCM nodes, set the following property to true: hdds.grpc.tls.enabled.

Note:

- gRPC TLS configuration for Ozone is supported only on new CDP 7.1.7 clusters and not on clusters upgraded from CDP 7.1.6 to CDP 7.1.7. If you want to enable gRPC TLS on the upgraded CDP 7.1.7 clusters, you must contact Cloudera Support for more information.

Important: Before CDP 7.1.9, Ozone's internal certificates expired after one year and CA certificates expired after five years. When the certificates expire, you must manually renew and revoke them by performing the following steps:

1. When the Ozone internal SSL certificates expire, you must remove the existing key material and certificates from the services metadata directory and allow the system to regenerate the certificates at startup. To renew the internal certificates, see Procedure to force renew internal certificates.

   Since CDP 7.1.9, general service certificates are renewed automatically without the need for a restart or without causing any service disruptions. For more information, see *Release Notes*.
2. CA certificates expire after 5 years. Cloudera is working on the automatic renewal of the CA certificates within the system without any disruption, similar to the regular certificates. In case CA certificates expire, you need to follow the same procedure that is required for certificate revocation.

   To revoke a certificate, you must remove the full trust chain to stop trusting a compromised certificate. For this, remove the SCM certificates and any other certificates from the system. During the system startup, new certificates are created and distributed. To revoke a certificate, see Certificate revocation.