Cloudera Runtime 7.0.3

Configuring Ozone security

Date published: 2019-08-21
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Working with Ozone ACLs

Ozone supports native Access Control Lists (ACLs) that can be used independently or with Apache Ranger. with Apache Ranger. If Ranger authorisation is enabled, the native ACLs are not evaluated.

Important: Ozone is available for technical preview and considered to be under development. Do not use this component in your production systems. If you have questions regarding Ozone, contact support by logging a case on the Cloudera Support Portal.

Format of an Ozone ACL

Ozone ACLs are a super set of POSIX and S3. The format of an ACL is *object:who:rights*.

object

In an ACL, an *object* can be the following:

- Volume: An Ozone volume. For example, /volume1.
- Bucket: An Ozone bucket. For example, /volume1/bucket1.
- Key: An object key or an object. For example, /volume1/bucket1/key1.
- Prefix: A path prefix for a specific key. For example, /volume1/bucket1/prefix1/prefix2.

who

In an ACL, a *who* can be the following:

- User: A user in the Kerberos domain. The user can be named or unnamed.
- Group: A group in the Kerberos domain. The group can be named or unnamed.
- World: All authenticated users in the Kerberos domain. This maps to others in the POSIX domain.
- Anonymous: Indicates that the user field should be completely ignored. This value is required for the S3 protocol to indicate anonymous users.

  Note: Ozone Manager translates any S3 user that accesses Ozone by using the AWS v4 signature protocol to the appropriate Kerberos user.

right

In an ACL, a *right* can be the following:

- Create: Create buckets in a volume and keys in a bucket.

  Note: Only administrators can create volumes.

- List: List buckets and keys. This ACL is attached to the volume and buckets which allow listing of the child objects.
- Delete: Delete a volume, a bucket, or a key.
- Read: Read the metadata of a volume or a bucket, and read the data stream and metadata of a key.
- Write: Write the metadata of a volume or a bucket, and overwrite an existing key.
- Read_ACL: Read the ACL on a specific object.
- Write_ACL: Write the ACL on a specific object.
- ALL
- NONE

APIs for working with an Ozone ACL

Ozone supports a set of APIs for working with the ACLs. The APIs are as follows:

- SetAcl: Accepts the user principal, the name and type of an Ozone object, and a list of ACLs.
- GetAcl: Accepts the name and type of an Ozone object and returns the corresponding ACLs.
- AddAcl: Accepts the name and type of the Ozone object, and the ACL to add.
- RemoveAcl: Accepts the name and type of the Ozone object, and the ACL to remove.

Related Information

Apache Ranger Authorization

Using Ranger with Ozone

You can use Apache Ranger to secure access to your Ozone data. For Ozone to work with Ranger, you must enable support for ACLs and set the ACL authorizer class inside Ozone to be Ranger authorizer.

You must add the following properties to ozone-site.xml for Ozone to work with Ranger:

- ozone.acl.enabled: Set the value of this property to true.
- ozone.acl.authorizer.class: Set the value of this property to org.apache.ranger.authorization.ozone.authorizer.RangerOzoneAuthorizer.

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.kerberos.principal	The service principal of the SCM http server.
hdds.scm.http.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`
ozone.om.kerberos.keytab.file	The keytab file that the Ozone Manager daemon uses to log in as its service principal.
ozone.om.http.kerberos.principal	The service principal of the Ozone Manager http server.
ozone.om.http.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.authentication.kerberos.principal	The S3 Gateway principal. You can specify this value, for example, in the following format: `HTTP/_HOST@EXAMPLE.COM`
ozone.s3g.keytab.file	The keytab file used by the S3 Gateway.

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.keytab.file	The keytab file that the DataNode daemon uses to log in as its service principal.
hdds.datanode.http.kerberos.principal	The service principal of the DataNode http server.
hdds.datanode.http.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
   ```

Configure 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 key-level or the bucket-level. TDE is enabled at the bucket-level when a bucket is created.

Before you begin

The Key Management Server must be installed and running. Ozone uses the same Key Management Server as HDFS.

Procedure

1. Create a bucket encryption key.

   ```
   hadoop key create encKey
   ```

   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 encKey. 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.