Cloudera Runtime 7.3.2

Using Schema Registry

Date published: 2019-08-22
Date modified: 2026-03-31

https://docs.cloudera.com/

Legal Notice

Contents

Adding a new schema

Learn how to add a new schema to Schema Registry through the Schema Registry UI.

About this task

A schema is a collection of information that describes your data. The metadata of the schema includes group, version, and description. To add a new schema to Schema Registry, provide information about the schema, select a compatibility policy, and upload the schema text from a file.

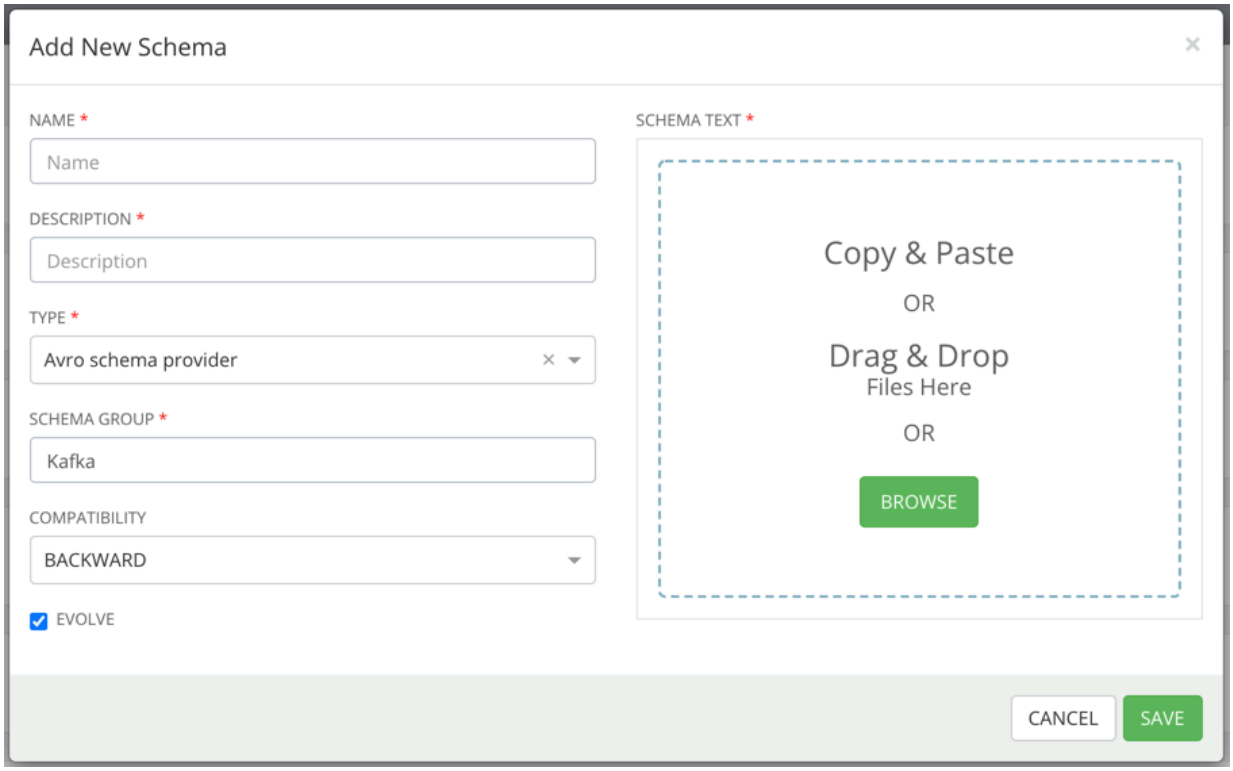
Before you begin

Ensure that you understand compatibility policies. Once selected, you cannot change the compatibility policy for a schema.

Procedure

1. From the Schema Registry UI, click the + icon.

   The Add New Schema dialog appears.



2. Add the schema metadata as follows:
   - Name

     A unique name for each schema. It is used as a key to look up schemas.
   - Description

     A short description of the schema.
   - Schema Type

     The schema format. Select one of the following formats:
     - Avro schema provider
     - JSON schema provider
   - Schema Group

     Allows you to group schemas in any logical order.
   - Compatibility

     This option appears only when the Avro schema provider option is selected. Sets the compatibility policy for the schema. Once set, this cannot be changed. Select one of the following options:
     - Backward
     - Forward
     - Both
     - None

     For more information about compatibility, see *Compatibility Policies*.
   - Schema Text
3. To allow schema to evolve over time by creating multiple versions, select the Evolve checkbox.

   Note:

   Unselecting Evolve means that you can only have one version of a schema.
4. Click Choose File to upload a new schema.

Related Information

Compatibility Policies

Querying a schema

Learn how to search for a particular schema in the Schema Registry UI.

You can use the Search box at the top of the Schema Registry UI to search for schemas by name, description, or schema text. To search, you can enter a schema name, key words, or any text string to return schemas with that value.

To return to your full list of schemas, clear the search box and press Enter on your keyboard.

Evolving a schema

Learn how to save each version of your schema as you make changes to it.

About this task

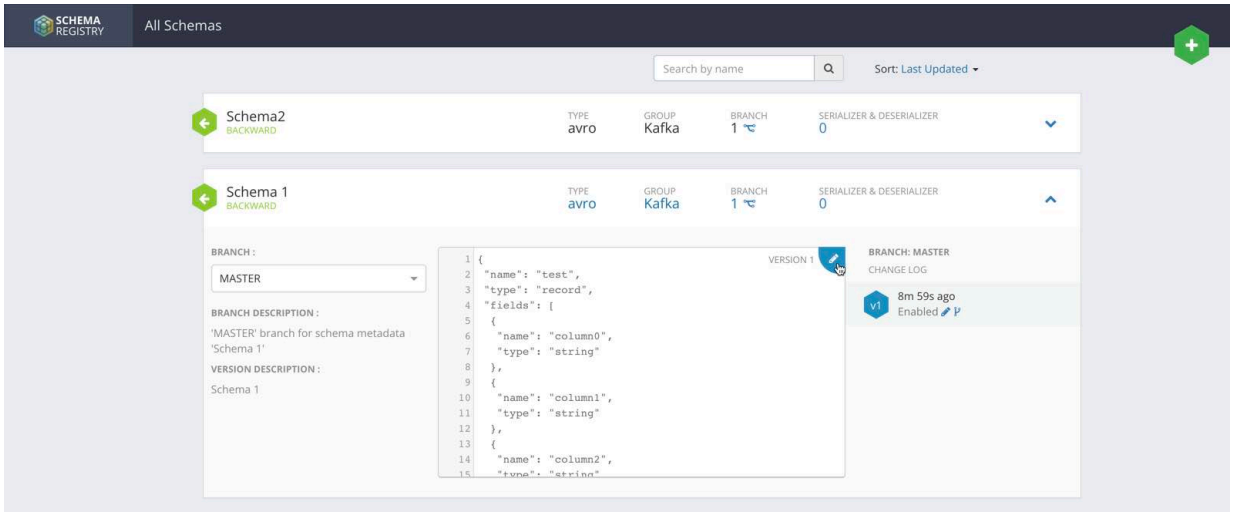
You evolve a schema when you create a new version. Schema Registry tracks the changes made to your schema, and stores each set of changes in a separate version of the schema. When multiple versions exist, you can select which version you want to use. Ensure that you understand compatibility policies, as they determine how you can evolve your schemas.

Before you begin

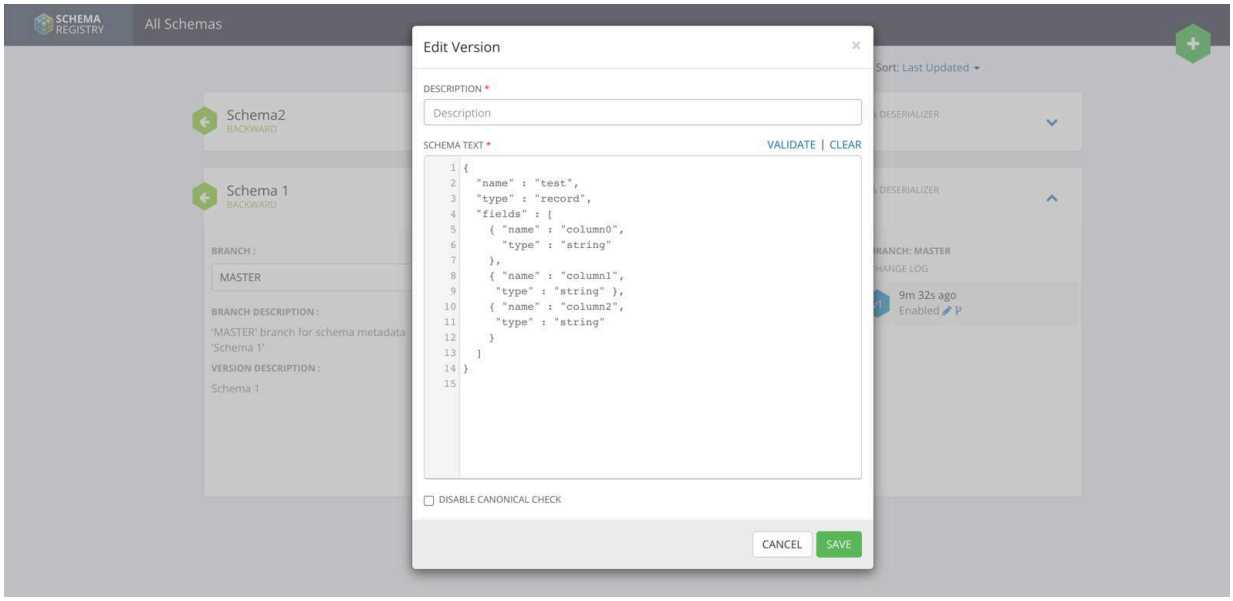
- You have selected the Evolve checkbox when initially adding the schema.
- You have saved the schema you want to evolve in a file.

Procedure

1. From the Schema Registry UI, identify the schema that you want to version.
2. Click the pencil icon to open the Edit Version dialog.



The Edit Version dialog appears.



3. Add a description of what has changed in this new version of the schema.

   You can view the description in the Schema Registry UI to easily understand what has changed in each version of the schema. Cloudera recommends that you add as much detail as you can.
4. Optional. Select the Disable Canonical Check checkbox if the schema version should be added despite being canonically similar to an existing schema version.

   The canonical check checkbox only applies to versions of schemas and not to a new schema.
5. Click Choose File to upload the schema you want to evolve.
6. Click OK.

Related Information
Compatibility Policies

Deleting a schema

You can use the Schema Registry REST API to delete schemas.

About this task

You can access the Schema Registry API Swagger documentation directly from the UI. To do this, append your URL with: /swagger/

For example: https://localhost:7790/swagger/.

Procedure

1. Find the DELETE /api/v1/schemaregistry/schemas/{schema_name} API under Schema.
2. Replace {schema_name} with the schema you wish to delete.
3. Invoke the API.

Importing Confluent Schema Registry schemas into Schema Registry

If you are migrating away from Confluent's Schema Registry, you can use the schema-import tool to import your schemas to the Schema Registry service.

About this task

Confluent Schema Registry stores all schemas in a Kafka topic usually named _schemas. To import these schemas, copy the content to a text file and upload to Schema Registry service.

Before you begin

You must have the export-import access policy with the Create permission assigned to your user account.

For more information on Schema Registry access policies, see *Schema Registry Authorization through Ranger Access Policies*.

Procedure

1. Copy the content of the Kafka topic (usually named _schemas) to a text file using the following command:

```
kafka-console-consumer --from-beginning --topic _schemas --formatter kaf
ka.tools.DefaultMessageFormatter --property print.key=true --property pr
int.value=true --bootstrap-server `hostname`:9092
```

The following is an example of a text file with the content of the Kafka topic:

```
{"keytype":"SCHEMA","subject":"Kafka-key","version":1,"magic":1}
{"subject":"Kafka-key","version":1,"id":1,"schema":"\"string\"","deleted
":false}
{"keytype":"SCHEMA","subject":"Car","version":1,"magic":1}      {"subje
ct":"Car","version":1,"id":2,"schema":"{\"type\":\"record\",\"name\":\"C
ar\",\"namespace\":\"com.cloudera\",\"fields\":[{\"name\":\"model\",\"type
```

```
\":\"string\"},{\"name\":\"color\",\"type\":\"string\",\"default\":\"blue
\"},{\"name\":\"price\",\"type\":\"string\",\"default\":\"0\"},{\"name\":
\"year\",\"type\":[\"null\",\"string\"],\"default\":null}]}","deleted":f
alse}
{"keytype":"SCHEMA","subject":"Car","version":2,"magic":1}      {"subject"
:"Car","version":2,"id":3,"schema":"{\"type\":\"record\",\"name\":\"Car\",
\"namespace\":\"com.cloudera\",\"fields\":[{\"name\":\"model\",\"type\":\"
string\"},{\"name\":\"color\",\"type\":\"string\",\"default\":\"blue\"},
{\"name\":\"price\",\"type\":\"string\",\"default\":\"0\"}]}","deleted":
false}
```

2. Upload the text file to Schema Registry by using the following endpoint: /api/v1/schemaregistry/import.

   Use the following command:

   ```
   curl -u : --negotiate --form file='@<filename>' http://`hostname`:7788/a
   pi/v1/schemaregistry/import?format=1&failOnError=true
   ```

   Where,

   file='<filename>'

   Enter the name of the file.

   format

   Can be 0 or 1. However, for schema content from Confluent, the file must be 1.

   failOnError=true

   This setting is useful if the process fails to import all the schemas when one or more schemas already exist in the target Schema Registry.

   The following is an example of the command:

   ```
   curl -u : --negotiate --form file='@schemadump.json' http://`hostname`:7
   788/api/v1/schemaregistry/import?format=1&failOnError=true
   ```

   The following is an example of the response:

   ```
   {"successCount":45,"failedCount":0,"failedIds":[]}
   ```

Related Information

Schema Registry Authorization through Ranger Access Policies

Exporting and importing schemas

Learn how to export and import schemas between different Schema Regitry instances allowing services to exchange data without the challenge of managing and sharing schemas between them.

Schemas stored in Schema Registry can be exported to a JSON file. The exported JSON file can then be imported into another Schema Registry database.

During an import, SchemaMetadata, SchemaBranch, and SchemaVersion objects are put into the database. These objects retain their ID as well as a number of other properties that are available in the JSON file used for the import. This way, serializing and deserializing protocols can continue to function without any change and Schema Registry clients can seamlessly switch between different Schema Registry instances. Both import and export operations can be done either by using curl through the command line or by using the Schema Registry API.

Exporting schemas using Schema Registry API

One of the methods of exporting schemas registered in Schema Registry into a JSON file is using the Schema Registry API. The file that you export can be used to import the exported schemas into a different Schema Registry. You can also export schemas using curl through the command line.

About this task

The following steps walk you through how you can export schemas by using the Schema Registry Swagger UI.

Before you begin

- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that Schema Registry is commissioned and running.
- Ensure that you have access to all credentials that are required to access and use Schema Registry.
- Ensure that you are logged in as a user with access to the Cloudera on cloud environment containing the cluster with Schema Registry.

Procedure

1. In Cloudera Management Console, go to Data Hub Clusters.
2. Find and select the Cloudera Data Hub cluster you want to export schemas from.
3. Under Services, click Schema Registry.
   The Schema Registry web UI opens in a new tab.
4. In the tab that has the Schema Registry web UI open, replace /ui/# at the end of the URL with /swagger.
5. Select GET /api/v1/schemaregistry/schemas/aggregated.
6. Click Try it out.
7. Click Execute.
8. Click Download found at the bottom right of the Response body pane to download the exported schemas.

Results

All schemas stored in the Schema Registry server are exported to a JSON file.

Importing schemas using Schema Registry API

One of the methods of importing schemas that were exported from one Schema Registry instance into another Schema Registry instance is using the Schema Registry API. You can also import schemas using curl through the command line.

About this task

The following steps describe how to import schemas by using the Schema Registry Swagger UI.

Important: Schema import is only recommended to an empty Schema Registry server or a Schema Registry server that has previously imported schemas. Importing schemas to a Schema Registry that contains manually added schemas is only possible if the Schema Registry instance has correctly configured ID ranges. Otherwise, importing schemas might result in overlapping ID ranges and a failed import. For more information on how to set up ID ranges, see *ID ranges in Schema Registry*.

Before you begin

- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that Schema Registry is commissioned and running.

- Ensure that you have access to all credentials that are required to access and use Schema Registry.
- Ensure that you have access to a JSON file that contains the exported schemas that you want to import. For more information on how to export schemas, see *Exporting schemas using Schema Registry API*.
- Ensure that you are logged in as a user with access to the Cloudera on cloud environment containing the cluster with Schema Registry.

Procedure

1. In Cloudera Management Console, go to Data Hub Clusters.
2. Find and select the Cloudera Data Hub cluster you want to import schemas to.
3. Under Services, click Schema Registry.

   The Schema Registry web UI opens in a new tab.
4. In the tab that has the Schema Registry web UI open, replace /ui/# at the end of the URL with /swagger.
5. Click 3. Export/Import.
6. Click POST /api/v1/schemaregistry/import.
7. Click Try it out.
8. Set format to 0.
9. Set failOnError to true.
10. Click Choose file and select a JSON file containing the schema data that you want to import.
11. Click Execute.

Results

Schemas are successfully imported into the target Schema Registry instance.

On successful import, a number of SchemaMetadata, SchemaBranch, and SchemaVersionInfo objects are added to the Schema Registry database.

Objects added to the database retain the following property values from the import file:

- Schema Metadata objects retain the ID, type, schemaGroup, name, description, compatibility, validationLevel, and evolve properties.
- SchemaBranch objects retain the ID, name, schemaMetadataName, and description properties.
- SchemaVersionInfo objects retain the ID, schemaMetadataId, name, description, version, and schemaText properties.

A successful import also results in a JSON file that contains the following:

```
{
  "successCount": 0,
  "failedCount": 0,
  "failedIds": [
    0
  ]
}
```

Where,

- successCount is the number of successfully imported SchemaVersionInfo objects.
- failedCount is the number of SchemaVersionInfo objects that failed to import.
- failedIds contains the IDs of failed SchemaVersionInfo objects.

Schemas (metadata/branch/version) that are already present in the database are skipped during an import. Skipped schemas are not counted in successCount or failedCount.

Related Information

ID ranges in Schema Registry

Exporting schemas using Schema Registry API

ID ranges in Schema Registry

You can avoid overlapping IDs for Schema Registry entities across different Schema Registry instances by setting ID ranges in each instance.

If your deployment has many Schema Registry instances in different clusters, you can ensure that each Schema Registry instance operates in a preset ID range so that there are no overlapping IDs. To achieve this, you need to set an ID range.

When an ID range is set, each Schema Registry entity (schema metadata, branches, versions, and so on) is created with an ID that falls within the configured range. Configuring a range can simplify schema export and import between Schema Registry instances where conflicting IDs can cause the import process to fail.

Consequently, having non-overlapping ID ranges allocated to each Schema Registry makes data replication easier when schema IDs are embedded in the data. This is important because replicated data can only be deserialized with compatible schemas on the replication target site.

When IDs reach the range maximum value, adding more schema fails until a new ID range is configured. The new range must be larger than the current one. The new range must also not overlap with any other ranges.

Important: Cloudera does not recommend that you import schemas into a Schema Registry's ID range. IDs assigned sequentially from the range start value might reach the imported IDs.

Setting a Schema Registry ID range

Learn how to set an ID range for schemas in Schema Registry using Cloudera Manager.

About this task

An ID range that is unique to a Schema Registry instance can be set in Cloudera Manager on the Schema Registry service's configuration page. Configuring an ID range does not alter already existing schemas.

Before you begin

- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that Schema Registry is commissioned and running.
- Ensure that you have access to all credentials that are required to access and use Schema Registry.
- Plan ahead and designate ID ranges for each Schema Registry instance. Ensure that the planned ranges do not overlap.
- Ensure that the ID range you plan to configure has enough IDs. Base the size of your range on the approximate number of schemas you expect to have.

Procedure

1. In Cloudera Manager, select the Schema Registry service.
2. Go to Configuration.
3. Find and configure the following properties:
   - ID Offset Range Minimum
   - ID Offset Range Maximum

   Configure the properties according to your planned ID ranges. For example, if you want to have an ID range of 1-100, set ID Offset Range Minimum to 1 and ID Offset Range Maximum to 100.
4. Click Save Changes.
5. Restart Schema Registry.

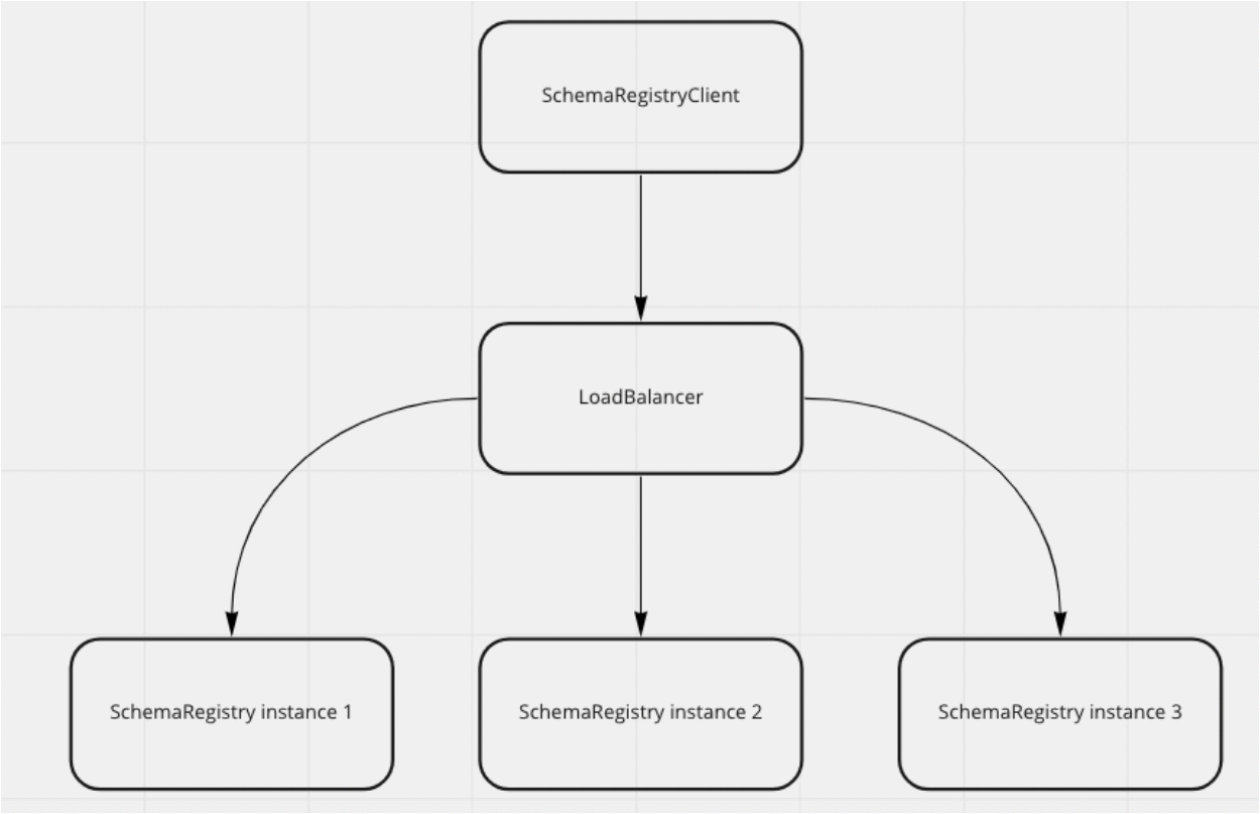
Results

When new schemas are added, either through the UI or the API, the schema IDs are allocated from the configured range.

Load balancer in front of Schema Registry instances

It is common to have multiple instances of the same application, for example Schema Registry, and have a load balancer in front of them. This can be useful for failover reasons in HA environments, and it can also help sharing the load between instances.

A load balancer can be any application (HAProxy, NGINX, and so on) that listens on a specific port and forwards requests (create schema, get schema, and so on) to Schema Registry instances.



A load balancer not only balances requests using round robin method, but also tracks application instances whether they are usable or not. Having a load balancer minimizes the need to change client-side properties with instance hostnames, because the only host that the client needs to know is the load balancer host. In case the environment is completely insecure, you do not need to perform any additional steps. But in case of SSL or Kerberos is enabled, some defending mechanisms (for example, hostname verification) can cause issues, which requires additional setup to make the load balancing work.

Note: Cloudera recommends running Schema Registry with distributed SerDes storage in High Availability mode. You can enable this by setting schema.registry.jar.storage.type to hdfs. Alternatively, you can set the storage type to local. In this case, ensure that the local storage is synchronized across nodes using a shared file system, such as NFS.

Note: Cloudera recommends using Knox as load balancer in Cloudera on cloud. It performs round robin method to load balance between multiple Schema Registry instances. Knox can be used differently than what is documented here, because it can handle authentication, so requests are not forwarded as is. In case Knox is used as load balancer, the setup mentioned in this document is not needed.

Configurations required to use load balancer with Kerberos enabled

Learn how to use load balancer in front of Schema Registry instances in an environment where Kerberos is enabled.

In this case, the clients connect to the load balancer, so they acquire a Kerberos ticket including the load balancer host. In the meantime, Schema Registry instances are logged in with their service principal including their FQDN. Because the load balancer host and the FQDN are probably not the same (if it is assumed that load balancer is not deployed to Schema Registry host), requests result in ticket mismatch to avoid man-in-the-middle attacks.

To ensure that requests forwarded by a load balancer are accepted, you need to do the following:

1. Log into Cloudera Manager and go to the Schema Registry service.
2. Set schema_registry_load_balancer_host to the host of the load balancer.
3. Restart the Schema Registry service.

This generates a new principal HTTP/<loadbalancerhost>@REALM. Each Schema Registry instance not only logs in with its FQDN related service principal, but also logs in with the load balancer related one, so forwarded requests do not fail.

Configurations required to use load balancer with SSL enabled

Learn how to use a load balancer in front of Schema Registry instances in an environment where SSL is enabled.

When SSL is enabled in your environment, the clients automatically check the SSL certificate of the Schema Registry servers and compare it with where they connected. This is called hostname verification and it fails because the clients connect to the load balancer host, but the Schema Registry server SSL certificate contains the Schema Registry server related host FQDN. To make the load-balancing work with forwarded requests, you can have multiple options to choose from:

- You can modify the server certificate and add a SAN (Subject Alternative Name) certificate including the load balancer host. This is the best option because it ensures that hostname verification only passes if the original FQDN or the load balancer host is used; otherwise it fails.
- If changing the certificates is a big effort, you can disable hostname verification on the Schema Registry client side. The Schema Registry client holds all SSL related properties in a map that contains the schema.regis try.client.ssl key in the properties. In this map, it is possible to set an entry with the hostnameVerifierClass key. The value of this key points to a class that implements the Java specific HostnameVerifier interface. An implementation to allow every hostname during hostname verification is shown in the following example:

```
public static class HostnameVerificationDisabled implements HostnameVeri
fier {

 @Override
 public boolean verify(String hostname, SSLSession session) {
   // Allowing everything is the same as disabling hostname verification.
   return true;
 }
}
```

An example setting of this property is as follows:

```
// Create map for all client properties
HashMap<String, String> clientProperties = new HashMap<>();
// Create SSL related map
HashMap<String, String> sslProperties = new HashMap<>();
// Set hostname verifier in SSL properties
sslProperties.put("hostnameVerifierClass", HostnameVerificationDisabled.clas
s.getName());

// Set other SSL related properties
```

```
sslProperties.put(...) ;
// Set SSL map on client properties map
clientProperties.put(“schema.registry.client.ssl”, sslProperties);
```

Important: This solution is less secure, because it disables an SSL client-side defending mechanism. So, Cloudera recommends to use it carefully.