EDB Postgres Hadoop Data Adapters Guide

Version 2.0

Built at 2020-10-05T20:32:04

1 What's New

The following features are added to the EDB Postgres Hadoop Data Adapter for release `2.0`:

- The data adapter now supports use of LDAP authentication.
- The data adapter now supports use of Apache Spark and Spark Thrift Server.

2 Supported Platforms

The data adapter is supported on:

- RHEL 6.5 and 7.0 (64-bit)
- CentOS 6.5 and 7.0 (64-bit)
- Debian 7 and 8
- SLES 11 and 12
- Ubuntu 14.04 LTS

The data adapter is supported on these platforms, using Advanced Server or PostgreSQL backing databases.

The Hadoop data adapter supports use of the Hadoop file system using a HiveServer2 interface or Apache Spark using the Spark Thrift Server.

3 Supported Authentication Methods

The data adapter supports NOSASL and LDAP authentication.

Using LDAP Authentication

When using the data adapter with LDAP authentication, you must first configure the Hive server or Spark server to use LDAP authentication. The configured server must provide a `hive-site.xml` file that includes the connection details for the LDAP server. For example:

```
<property>
  <name>hive.server2.authentication</name>
  <value>LDAP</value>
  <description>
    Expects one of [nosasl, none, ldap, kerberos, pam,
custom].
    Client authentication types.
      NONE: no authentication check
      LDAP: LDAP/AD based authentication
      KERBEROS: Kerberos/GSSAPI authentication
      CUSTOM: Custom authentication provider
              (Use with property
hive.server2.custom.authentication.class)
      PAM: Pluggable authentication module
      NOSASL:  Raw transport
  </description>
</property>
<property>
  <name>hive.server2.authentication.ldap.url</name>
  <value>ldap://localhost</value>
  <description>LDAP connection URL</description>
</property>
<property>
  <name>hive.server2.authentication.ldap.baseDN</name>
  <value>ou=People,dc=itzgeek,dc=local</value>
  <description>LDAP base DN</description>
</property>
```

Then, when starting the hive server, include the path to the `hive-site.xml` file in the command. For example:

```
./hive --config path_to_hive-site.xml_file --service
HiveServer2
```

Where *path_to_hive-site.xml_file* specifies the complete path to the `hive-site.xml` file.

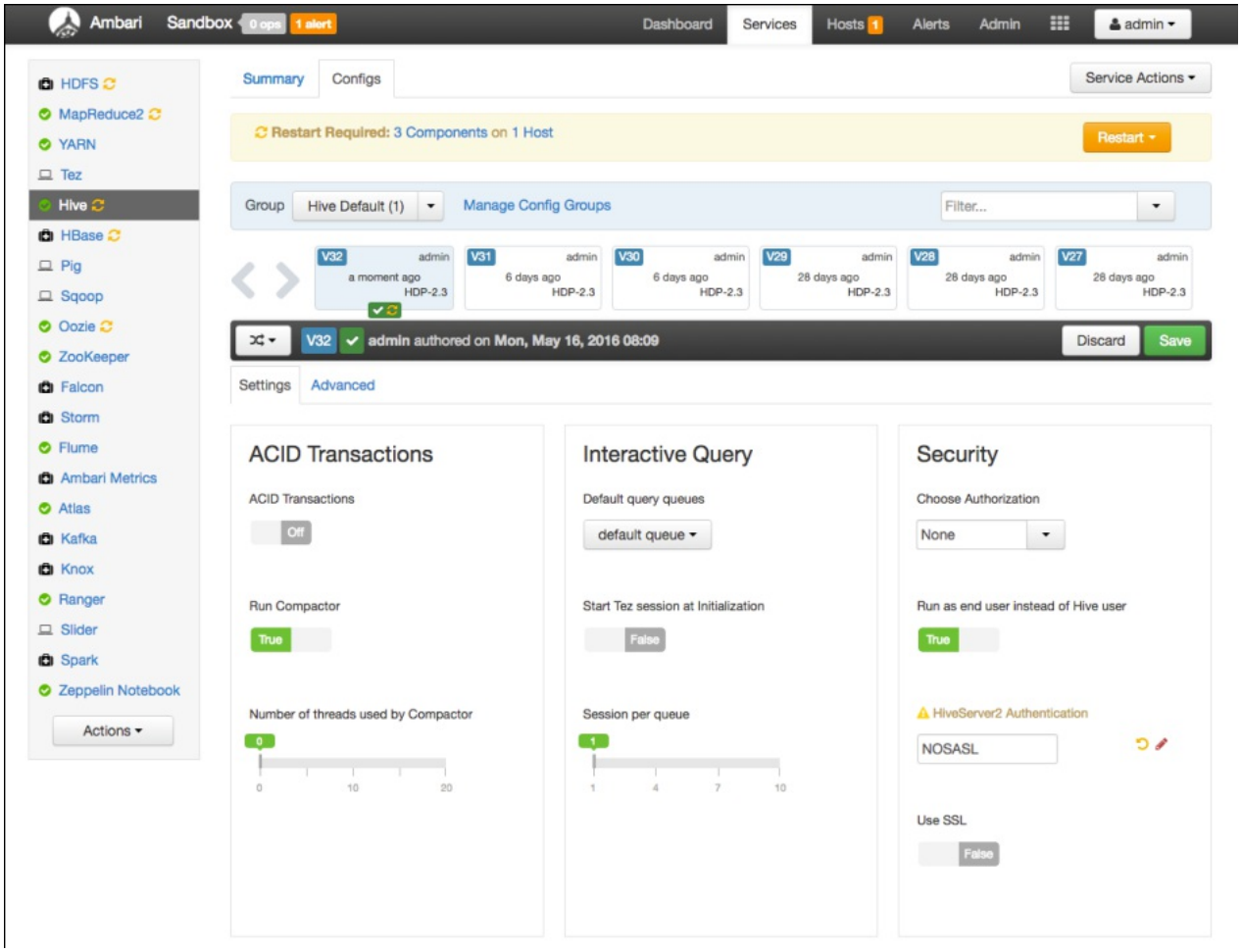
When creating the user mapping, you must provide the name of a registered LDAP user and the corresponding password as options. For details, see Create User Mapping.

Using NOSASL Authentication

When using NOSASL authentication with the data adapter, set the authorization to `None`, and the authentication method to `NOSASL` on the Hive or Spark server. For example, if you start the Hive server at the command line, include the `hive.server2.authentication` configuration parameter in the command:

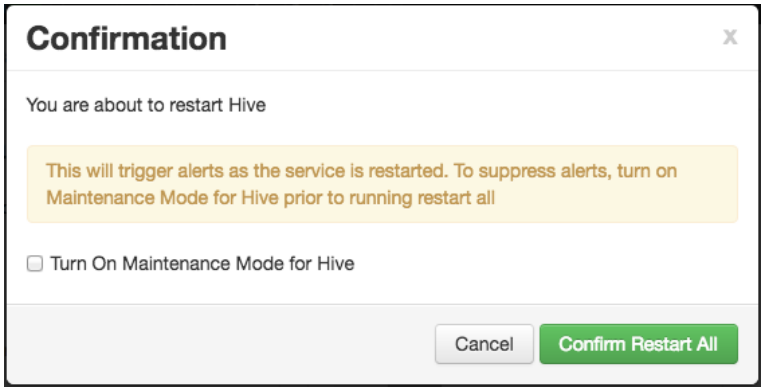
```
hive –service hiveserver2 –hiveconf
hive.server2.authentication=NOSASL
```

The following example uses the Ambari client interface to manage the Hadoop data source. After authenticating with Ambari, navigate to the `Hive` page, and select the `Configs` tab.

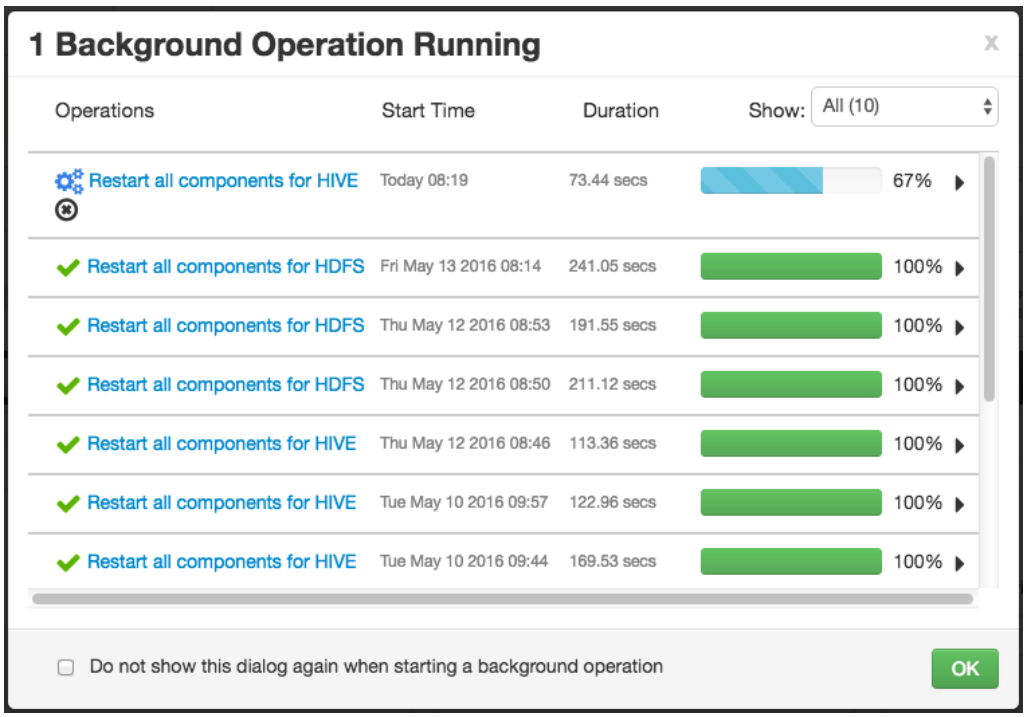


Edit the `HiveServer2 Authentication` field in the `Security` pane, specifying `NOSASL`. Click the `Save` button in the upper-right corner; if prompted, provide a comment about the change and click `Save`, and then `OK` to confirm the change.

After modifying the authentication type, you must restart the Hive server. To restart the server, select `Restart All` from the `Service Actions` drop-down listbox. When prompted, select the `Confirm Restart All` button.



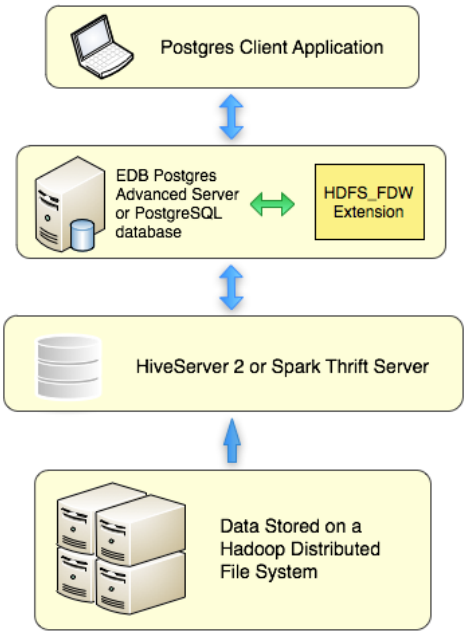
Ambari opens a dialog that will confirm the progress of the server restart.



4 Architecture Overview

Hadoop is a framework that allows you to store a large data set in a distributed file system.

The EDB Postgres Hadoop data wrapper provides an interface between a Hadoop file system and a Postgres database. The Hadoop data wrapper translates a Postgres `SELECT` statement into a query that is understood by the HiveQL or Spark SQL interface.



When possible, the data adapter asks the Hive or Spark server to peform the actions associated with the `WHERE` clause of a `SELECT` statement. Pushing down the `WHERE` clause improves performance by decreasing the amount of data moving across the network. Currently push-down is not supported for:

- Aggregate Functions (such as `AVG`, `SUM`, and `COUNT`)
- Foreign Joins
- Sorts

5 Installing the Hadoop Data Adapter

The Hadoop data adapter can be installed with a graphical installer, an RPM package, or via StackBuilder Plus. During the installation process, the installer will satisfy software prerequisites.

Prerequisites

Before installing the data adapter, install EDB Postgres Advanced Server or PostgreSQL on the host from which you will query Hadoop. After installing Postgres, modify the `postgresql.conf` file; the `postgresql.conf` file is located in:

- For an RPM server installation, `/var/lib/edb/as_version/data`
- For a graphical installation, `/opt/edb/as_version/data`

Modify the configuration file with your editor of choice, adding the `hdfs_fdw.jvmpath` parameter to the end of the configuration file, and setting the value to specify the location of the Java virtual machine (`libjvm.so`).

If your data contains columns with a type of `DATE`, you must also set the following values in the `postgresql.conf` file before using the data adapter:

```
datestyle = 'iso, dmy'
edb_redwood_date = off
```

You must also set the value of `hdfs_fdw.classpath` to indicate the location of the java class files used by the adapter; use a colon (:) as a delimiter between each path. For example:

```
hdfs_fdw.classpath=
'/usr/local/edb95/lib/postgresql/HiveJdbcClient-1.0.jar:
/home/edb/Projects/hadoop_fdw/hadoop/share/hadoop/common/hadoop
-common-2.6.4.jar:
/home/edb/Projects/hadoop_fdw/apache-hive-1.0.1-bin/lib/hive-
jdbc-1.0.1-standalone.jar'
```

Please note: the following work-around resolves an issue that will be fixed in the next minor release. If your data contains columns with a type of `DATE`, you must also set the following values in the `postgresql.conf` file before using the data adapter:

```
datestyle = 'iso, dmy'
edb_redwood_date = off
```

After setting the parameter values, restart the Postgres server. For detailed information about controlling the service on an Advanced Server host, see the EDB Postgres Advanced Server Installation Guide, available at:

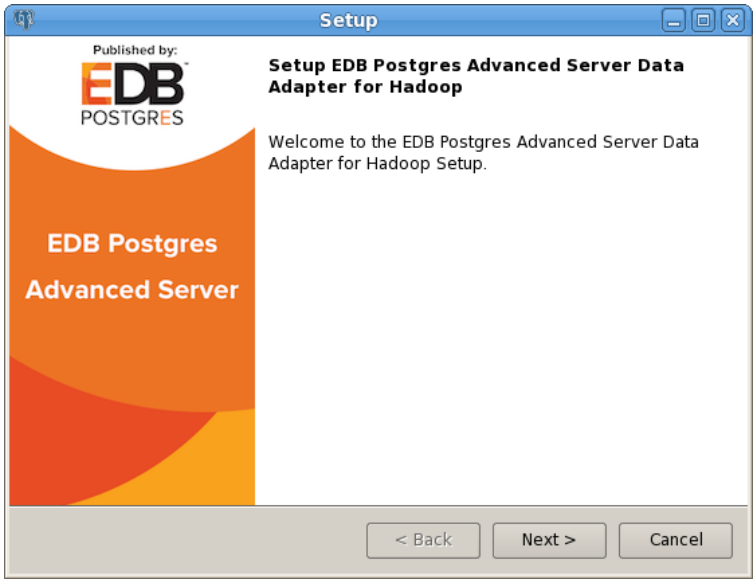
> https://www.enterprisedb.com/resources/product-documentation

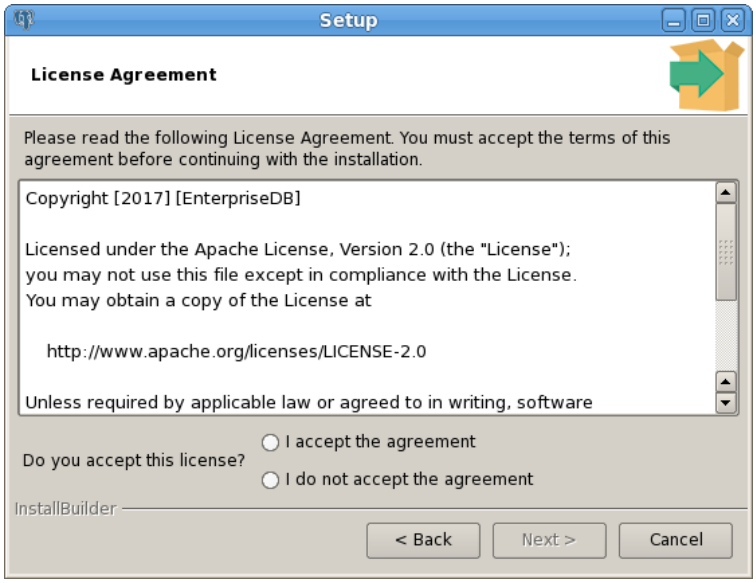
Using a Graphical Installer to Install the Data Adapter

You can download a graphical installer for the Hadoop data adapter from the EnterpriseDB website. After downloading the installer, assume superuser privileges, and invoke the installer with the command:

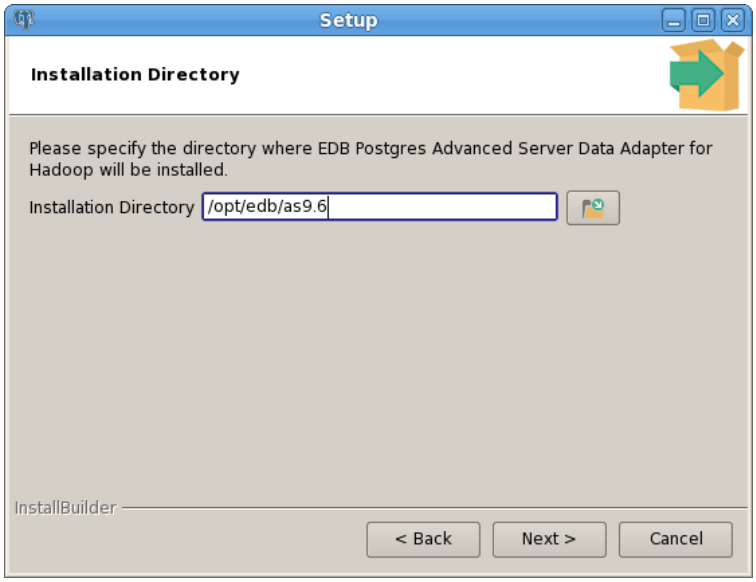
```
path_to_installer/edb_hdfs_fdw-9.x-x.x.x-linux-x64.run
```

The wizard will prompt you to select an installation language; select a language, and click the `OK` button. The setup wizard opens as shown below:

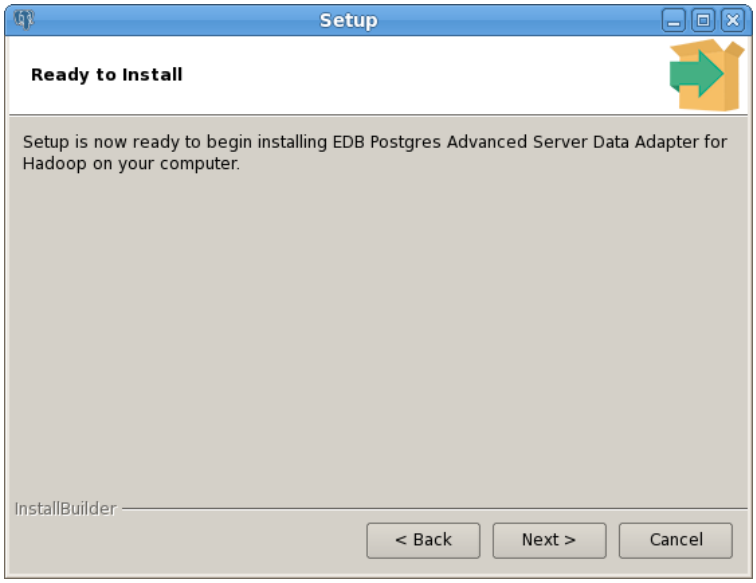




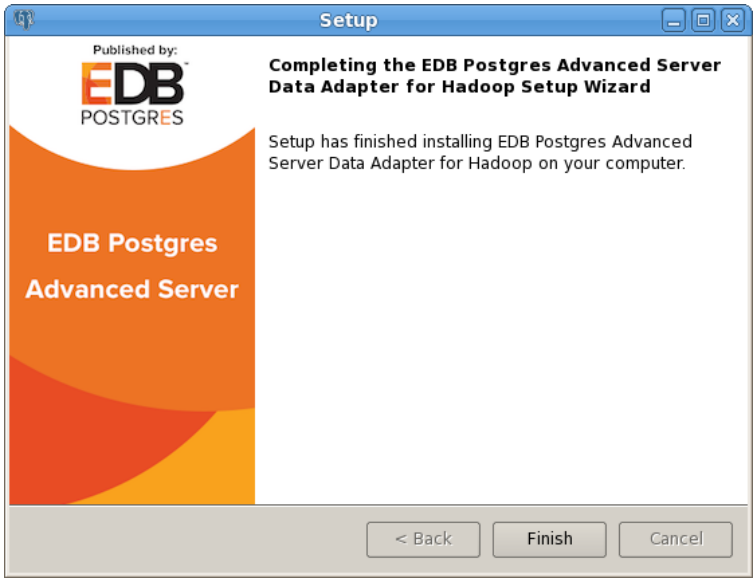
The `License Agreement` opens; accept the agreement, and click `Next` to continue, or click `Cancel` to exit the installer.



Use the `Installation Directory` dialog to specify the installation directory for the data adapter; the default location is your Postgres installation directory.



When the `Ready to Install` dialog opens, click the `Next` button to start installing the Hadoop Data Adapter. Progress bars will keep you informed of the installer's progress.



When the installation completes, the setup wizard informs you that setup has finished installing the Hadoop Data Adapter; click `Finish` to close the wizard.

Using an RPM Package to Install the Data Adapter

The RPM installation package for the Hadoop data adapter is available from the EnterpriseDB repository. Before installing the `data adapter`, you must:

- Install the `epel-release` package:

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Please note that you may need to enable the `[extras]` repository definition in the `CentOS-Base.repo` file (located in `/etc/yum.repos.d`).

You must also have credentials that allow access to the EnterpriseDB repository. For information about requesting credentials, visit:

> https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install the data adapter.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges and invoke the following command:

```
yum -y install https://yum.enterprisedb.com/edb-repo-rpms/edb-repo-latest.noarch.rpm
```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EnterpriseDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-
$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

After saving your changes to the configuration file, you can use the `yum install` command to the data adapter:

```
yum install edb-as-xx-hdfs_fdw
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter a `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

6 Configuring the Hadoop Data Adapter

Before creating the extension and the database objects that use the extension, you must modify the Postgres host, providing the location of the supporting libraries; for details, see

Prerequisites. Then, before using the data adapter, you must:

1. Use the `CREATE EXTENSION` command to create the Hadoop data adapter extension on the Postgres host.
2. Use the `CREATE SERVER` command to define a connection to the Hadoop file system.
3. Use the `CREATE USER MAPPING` command to define a mapping that associates a Postgres role with the server.
4. Use the `CREATE FOREIGN TABLE` command to define a table in the Advanced Server database that corresponds to a database that resides on the Hadoop cluster.

CREATE EXTENSION

Use the `CREATE EXTENSION` command to create the `hdfs_fdw` extension. To invoke the command, use your client of choice (for example, psql) to connect to the Postgres database from which you will be querying the Hive or Spark server, and invoke the command:

```
CREATE EXTENSION [IF NOT EXISTS] hdfs_fdw [WITH] [SCHEMA
schema_name];
```

Parameters

`IF NOT EXISTS`

Include the `IF NOT EXISTS` clause to instruct the server to issue a notice instead of throwing an error if an extension with the same name already exists.

`schema_name`

Optionally specify the name of the schema in which to install the extension's objects.

Example

The following command installs the `hdfs_fdw data` adapter:

```
CREATE EXTENSION hdfs_fdw;
```

For more information about using the `CREATE EXTENSION` command, see:

https://www.postgresql.org/docs/current/static/sql-createextension.html.

CREATE SERVER

Use the `CREATE SERVER` command to define a connection to a foreign server. The syntax is:

```
CREATE SERVER server_name FOREIGN DATA WRAPPER hdfs_fdw
    [OPTIONS (option 'value' [, ...])]
```

The role that defines the server is the owner of the server; use the `ALTER SERVER` command to reassign ownership of a foreign server. To create a foreign server, you must have `USAGE` privilege on the foreign-data wrapper specified in the `CREATE SERVER` command.

Parameters

`server_name`

> Use `server_name` to specify a name for the foreign server. The server name must be unique within the database.

`FOREIGN_DATA_WRAPPER`

> Include the `FOREIGN_DATA_WRAPPER` clause to specify that the server should use the `hdfs_fdw` foreign data wrapper when connecting to the cluster.

`OPTIONS`

> Use the `OPTIONS` clause of the `CREATE SERVER` command to specify connection information for the foreign server. You can include:

Option	Description
host	The address or hostname of the Hadoop cluster. The default value is 127.0.0.1.
port	The listener port number of the client. The default is 10000.
client_type	Specify hiveserver2 or spark as the client type. To use the ANALYZE statement on Spark, you must specify a value of spark; if you do not specify a value for client_type, the default value is hiveserver2.
auth_type	The authentication type of the client; specify LDAP or NOSASL. If you do not specify an auth_type, the data wrapper will decide the auth_type value on the basis of the user mapping: • If the user mapping includes a user name and password, the data wrapper will use LDAP authentication. • If the user mapping does not include a user name and password, the data wrapper will use NOSASL authentication.

Option	Description
connect_timeout	The length of time before a connection attempt times out. The default value is 300 seconds.
fetch_size	A user-specified value that is provided as a parameter to the JDBC API setFetchSize. The default value is 10,000.
log_remote_sql	If true, logging will include SQL commands executed on the remote hive server and the number of times that a scan is repeated. The default is false.
query_timeout	Use query_timeout to provide the number of seconds after which a request will timeout if it is not satisfied by the Hive server. Query timeout is not supported by the Hive JDBC driver.
use_remote_estimate	Include the use_remote_estimate to instruct the server to use EXPLAIN commands on the remote server when estimating processing costs. By default, use_remote_estimate is false, and remote tables are assumed to have 1000 rows.

Example

The following command creates a foreign server named `hdfs_server` that uses the `hdfs_fdw` foreign data wrapper to connect to a host with an IP address of `170.11.2.148`:

```
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER hdfs_fdw
OPTIONS (host '170.11.2.148', port '10000', client_type
'hiveserver2', auth_type 'LDAP', connect_timeout '10000',
query_timeout '10000');
```

The foreign server uses the default port (`10000`) for the connection to the client on the Hadoop cluster; the connection uses an LDAP server.

For more information about using the `CREATE SERVER` command, see:

> https://www.postgresql.org/docs/current/static/sql-createserver.html

CREATE USER MAPPING

Use the `CREATE USER MAPPING` command to define a mapping that associates a Postgres role with a foreign server:

```
CREATE USER MAPPING FOR role_name SERVER server_name
       [OPTIONS (option 'value' [, ...])];
```

You must be the owner of the foreign server to create a user mapping for that server.

Please note: the data adapter supports NOSASL and LDAP authentication. If you are creating a user mapping for a server that uses LDAP authentication, use the `OPTIONS` clause to provide the connection credentials (the username and password) for an existing LDAP user. If the server uses NOSASL authentication, omit the OPTIONS clause when creating the user mapping.

Parameters

`role_name`

> Use `role_name` to specify the role that will be associated with the foreign server.

`server_name`

> Use `server_name` to specify the name of the server that defines a connection to the Hadoop cluster.

`OPTIONS`

> Use the `OPTIONS` clause to specify connection information for the foreign server. If you are using LDAP authentication, provide a:
>
> `username`: the name of the user on the LDAP server.
>
> `password`: the password associated with the username.
>
> If you do not provide a user name and password, the data wrapper will use NOSASL authentication.

Example

The following command creates a user mapping for a role named `enterprisedb`; the mapping is associated with a server named `hdfs_server`:

```
CREATE USER MAPPING FOR enterprisedb SERVER hdfs_server;
```

If the database host uses LDAP authentication, provide connection credentials when creating the user mapping:

```
CREATE USER MAPPING FOR enterprisedb SERVER hdfs_server
OPTIONS (username 'alice', password '1safepwd');
```

The command creates a user mapping for a role named `enterprisedb` that is associated with a server named `hdfs_server`. When connecting to the LDAP server, the Hive or

Spark server will authenticate as `alice`, and provide a password of `1safepwd`.

For detailed information about the `CREATE USER MAPPING` command, see:

> https://www.postgresql.org/docs/current/static/sql-createusermapping.html

CREATE FOREIGN TABLE

A foreign table is a pointer to a table that resides on the Hadoop host. Before creating a foreign table definition on the Postgres server, connect to the Hive or Spark server and create a table; the columns in the table will map to to columns in a table on the Postgres server. Then, use the `CREATE FOREIGN TABLE` command to define a table on the Postgres server with columns that correspond to the table that resides on the Hadoop host. The syntax is:

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ OPTIONS ( option 'value' [, ... ]
) ] [ COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
  SERVER server_name [ OPTIONS ( option 'value' [, ... ] ) ]
```

where `column_constraint` is:

```
[ CONSTRAINT constraint_name ]
{ NOT NULL | NULL | CHECK (expr) [ NO INHERIT ] | DEFAULT
default_expr }
```

and `table_constraint` is:

```
[ CONSTRAINT constraint_name ] CHECK (expr) [ NO INHERIT ]
```

Parameters

`table_name`

> Specifies the name of the foreign table; include a schema name to specify the schema in which the foreign table should reside.

`IF NOT EXISTS`

> Include the `IF NOT EXISTS` clause to instruct the server to not throw an error if a table with the same name already exists; if a table with the same name exists, the server will issue a notice.

`column_name`

> Specifies the name of a column in the new table; each column should correspond to a column described on the Hive or Spark server.

`data_type`

> Specifies the data type of the column; when possible, specify the same data type for each column on the Postgres server and the Hive or Spark server. If a data type with the same name is not available, the Postgres server will attempt to cast the data type to a type compatible with the Hive or Spark server. If the server cannot identify a compatible data type, it will return an error.

`COLLATE collation`

> Include the `COLLATE` clause to assign a collation to the column; if not specified, the column data type's default collation is used.

`INHERITS (parent_table [, ... ])`

> Include the `INHERITS` clause to specify a list of tables from which the new foreign table automatically inherits all columns. Parent tables can be plain tables or foreign tables.

`CONSTRAINT constraint_name`

> Specify an optional name for a column or table constraint; if not specified, the server will generate a constraint name.

`NOT NULL`

> Include the `NOT NULL` keywords to indicate that the column is not allowed to contain null values.

`NULL`

> Include the `NULL` keywords to indicate that the column is allowed to contain null values. This is the default.

`CHECK (expr) [NO INHERIT]`

> Use the `CHECK` clause to specify an expression that produces a Boolean result that each row in the table must satisfy. A check constraint specified as a column constraint should

reference that column's value only, while an expression appearing in a table constraint can reference multiple columns.

A `CHECK` expression cannot contain subqueries or refer to variables other than columns of the current row.

Include the `NO INHERIT` keywords to specify that a constraint should not propagate to child tables.

`DEFAULT default_expr`

Include the `DEFAULT` clause to specify a default data value for the column whose column definition it appears within. The data type of the default expression must match the data type of the column.

`SERVER server_name [OPTIONS (option 'value' [, ... ] ) ]`

To create a foreign table that will allow you to query a table that resides on a Hadoop file system, include the `SERVER` clause and specify the `server_name` of the foreign server that uses the Hadoop data adapter.

Use the `OPTIONS` clause to specify the following `options` and their corresponding values:

option	value
dbname	The name of the database on the Hive server; the database name is required.
table_name	The name of the table on the Hive server; the default is the name of the foreign table.

Example

To use data that is stored on a distributed file system, you must create a table on the Postgres host that maps the columns of a Hadoop table to the columns of a Postgres table. For example, for a Hadoop table with the following definition:

```
CREATE TABLE weblogs (
 client_ip           STRING,
 full_request_date   STRING,
 day                 STRING,
 month               STRING,
 month_num           INT,
 year                STRING,
 hour                STRING,
 minute              STRING,
```

```
 second                STRING,
 timezone              STRING,
 http_verb             STRING,
 uri                   STRING,
 http_status_code      STRING,
 bytes_returned        STRING,
 referrer              STRING,
 user_agent            STRING)
row format delimited
fields terminated by '\t';
```

You should execute a command on the Postgres server that creates a comparable table on the Postgres server:

```
CREATE FOREIGN TABLE weblogs
(
 client_ip                TEXT,
 full_request_date        TEXT,
 day                      TEXT,
 Month                    TEXT,
 month_num                INTEGER,
 year                     TEXT,
 hour                     TEXT,
 minute                   TEXT,
 second                   TEXT,
 timezone                 TEXT,
 http_verb                TEXT,
 uri                      TEXT,
 http_status_code         TEXT,
 bytes_returned           TEXT,
 referrer                 TEXT,
 user_agent               TEXT
)
SERVER hdfs_server
        OPTIONS (dbname 'webdata', table_name 'weblogs');
```

Include the `SERVER` clause to specify the name of the database stored on the Hadoop file system (`webdata`) and the name of the table (`weblogs`) that corresponds to the table on the Postgres server.

For more information about using the `CREATE FOREIGN TABLE` command, see:

> https://www.postgresql.org/docs/current/static/sql-createforeigntable.html

Data Type Mappings

When using the foreign data wrapper, you must create a table on the Postgres server that mirrors the table that resides on the Hive server. The Hadoop data wrapper will automatically convert the following Hive data types to the target Postgres type:

Hive	PostgreSQL
BCHAR	CHAR
BIGINT	INTEGER
BOOLEAN	BOOLEAN
BYTEA	BYTEA
CHAR	CHAR
DATE	DATE
FLOAT4	FLOAT
FLOAT8	FLOAT
INT	INTEGER
INT4	INT8
INT8 (BIGINT)	INTEGER
NAME	NAME
SMALLINT	INTEGER
STRING	TEXT
TEXT	TEXT
TIME	DATETIME
TIMESTAMP with timezone	DATETIME
TIMESTAMP without timezone	DATETIME
TINYINT	INTEGER
VARCHAR	VARCHAR

7 Using the Hadoop Data Adapter

After configuring the data adapter, and creating a table on the Postgres server that mirrors the table on the distributed file system, you can seamlessly access the data from a Postgres server.

To query data on the Hadoop file system, connect to your Postgres database with your choice of client (for example, the PEM client or EDB-PSQL). When you query the table that resides on the Postgres host, the queries will be re-directed to the Hadoop host.

Query Pushdown

When possible, the data adapter pushes down the `WHERE` predicate and target column list to the Hive server. Supporting operations (aggregates, foreign joins, and sorts) are performed by the database server. Remote push down provides better performance and improves network traffic by distributing processing to the remote server when possible, decreasing the amount of data that must travel to the database server.

You can demonstrate command push-down by including the `EXPLAIN` clause in a query of a foreign table:

```
EXPLAIN VERBOSE SELECT DISTINCT client_ip IP, count(*) FROM
weblogs WHERE uri = '/demo' GROUP BY IP HAVING count(*) > 10;
                            QUERY PLAN
------------------------------------------------------------
--
 HashAggregate  (cost=21.00..23.00 rows=200 width=32)
   Output: client_ip, (count(*))
   Group Key: weblogs.client_ip, count(*)
   ->  HashAggregate  (cost=17.50..20.00 rows=200 width=32)
         Output: client_ip, count(*)
         Group Key: weblogs.client_ip
         Filter: (count(*) > 10)
         ->  Foreign Scan on public.weblogs
(cost=100.00..10.00 rows=1000 width=32)
               Output: client_ip, full_request_date, day,
month, month_num, year, hour, minute, second, timezone,
http_verb, uri, http_status_code, bytes_returned, referrer,
user_agent
               Remote SQL: SELECT client_ip FROM
fdw_db.weblogs WHERE ((uri = '/demo'))
(10 rows)
```

As noted near the end of the query plan, the `WHERE` clause is performed on the `Remote` server.

Identifying the Data Adapter Version

The `HDFS_FDW` data adapter includes a function that you can use to identify the currently installed version of the `.so` file for the data adapter. To use the function, connect to the Postgres server, and enter:

```
SELECT hdfs_fdw_version();
```

The function returns the version number:

```
hdfs_fdw_version
-----------------
20003
```