

Setting Up HiveServer2

HiveServer2

- [HiveServer2](#)
 - [How to Configure](#)
 - [Configuration Properties in the hive-site.xml File](#)
 - [Running in HTTP Mode](#)
 - [Cookie Based Authentication](#)
 - [Optional Global Init File](#)
 - [Logging Configuration](#)
 - [How to Start](#)
 - [Usage Message](#)
 - [Authentication/Security Configuration](#)
 - [Configuration](#)
 - [Impersonation](#)
 - [Integrity/Confidentiality Protection](#)
 - [SSL Encryption](#)
 - [Setting up SSL with self-signed certificates](#)
 - [Selectively disabling SSL protocol versions](#)
 - [Pluggable Authentication Modules \(PAM\)](#)
 - [Setting up HiveServer2 job credential provider](#)
 - [Scratch Directory Management](#)
 - [Configuration Properties](#)
 - [ClearDanglingScratchDir Tool](#)
 - [Web UI for HiveServer2](#)
 - [Python Client Driver](#)
 - [Ruby Client Driver](#)

[HiveServer2](#) (HS2) is a server interface that enables remote clients to execute queries against Hive and retrieve the results (a more detailed intro [here](#)). The current implementation, based on Thrift RPC, is an improved version of [HiveServer](#) and supports multi-client concurrency and authentication. It is designed to provide better support for open API clients like JDBC and ODBC.

- The Thrift interface definition language (IDL) for HiveServer2 is available at <https://github.com/apache/hive/blob/trunk/service/ift/TCLIService.thrift>.
- Thrift documentation is available at <http://thrift.apache.org/docs/>.

This document describes how to set up the server. How to use a client with this server is described in the [HiveServer2 Clients](#) document.

Version

 Introduced in Hive version 0.11. See [HIVE-2935](#).

How to Configure

Configuration Properties in the `hive-site.xml` File

```
hive.server2.thrift.min.worker.threads – Minimum number of worker threads, default 5.
hive.server2.thrift.max.worker.threads – Maximum number of worker threads, default 500.
hive.server2.thrift.port – TCP port number to listen on, default 10000.
hive.server2.thrift.bind.host – TCP interface to bind to.
```

See [HiveServer2 in the Configuration Properties](#) document for additional properties that can be set for HiveServer2.

Optional Environment Settings

```
HIVE_SERVER2_THRIFT_BIND_HOST – Optional TCP host interface to bind to. Overrides the configuration file setting.
HIVE_SERVER2_THRIFT_PORT – Optional TCP port number to listen on, default 10000. Overrides the configuration file setting.
```

Running in HTTP Mode

HiveServer2 provides support for sending Thrift RPC messages over HTTP transport (Hive 0.13 onward, see [HIVE-4752](#)). This is particularly useful to support a proxying intermediary between the client and the server (for example, for load balancing or security reasons). Currently, you can run HiveServer2 in either TCP mode or the HTTP mode, but not in both. For the corresponding JDBC URL check this link: [HiveServer2 Clients -- JDBC Connection URLs](#). Use the following settings to enable and configure HTTP mode:

| Setting | Default | Description |
|---|---------|---|
| hive.server2.transport.mode | binary | Set to http to enable HTTP transport mode |

| | | |
|---|------------|---|
| hive.server2.thrift.http.port | 10001 | HTTP port number to listen on |
| hive.server2.thrift.http.max.worker.threads | 500 | Maximum worker threads in the server pool |
| hive.server2.thrift.http.min.worker.threads | 5 | Minimum worker threads in the server pool |
| hive.server2.thrift.http.path | cliservice | The service endpoint |

Cookie Based Authentication

[HIVE-9709](#) and [HIVE-9710](#) introduced cookie based authentication for HiveServer2 in HTTP mode. The HiveServer2 parameters (`hive.server2.thrift.http.cookie.*`) associated with this change can be found [here](#).

Optional Global Init File

A global init file can be placed in the configured [hive.server2.global.init.file.location](#) location (Hive 0.14 onward, see [HIVE-5160](#), [HIVE-7497](#), and [HIVE-8138](#)). This can be either the path to the init file itself, or a directory where an init file named ".hiverc" is expected.

The init file lists a set of commands that will run for users of this HiveServer2 instance, such as register a standard set of jars and functions.

Logging Configuration

HiveServer2 operation logs are available for Beeline clients (Hive 0.14 onward). These parameters configure logging:

- [hive.server2.logging.operation.enabled](#)
- [hive.server2.logging.operation.log.location](#)
- [hive.server2.logging.operation.verbose](#) (Hive 0.14 to 1.1)
- [hive.server2.logging.operation.level](#) (Hive 1.2 onward)

How to Start

```
$HIVE_HOME/bin/hiveserver2
```

OR

```
$HIVE_HOME/bin/hive --service hiveserver2
```

Usage Message

The `-h` or `--help` option displays a usage message, for example:

```
$HIVE_HOME/bin/hive --service hiveserver2 -H
Starting HiveServer2
usage: hiveserver2
  -H, --help                Print help information
  --hiveconf <property=value>  Use value for given property
```

Authentication/Security Configuration

HiveServer2 supports Anonymous (no authentication) with and without SASL, Kerberos (GSSAPI), pass through LDAP, Pluggable Custom Authentication and Pluggable Authentication Modules (PAM, supported Hive 0.13 onwards).

Configuration

Authentication mode:

`hive.server2.authentication` – Authentication mode, default `NONE`. Options are `NONE` (uses plain SASL), `NOSASL`, `KERBEROS`, `LDAP`, `PAM` and `CUSTOM`.

Set following for `KERBEROS` mode:

`hive.server2.authentication.kerberos.principal` – Kerberos principal for server.

`hive.server2.authentication.kerberos.keytab` – Keytab for server principal.

Set following for `LDAP` mode:

`hive.server2.authentication.ldap.url` – LDAP URL (for example, `ldap://hostname.com:389`).

`hive.server2.authentication.ldap.baseDN` – LDAP base DN. (Optional for AD.)

`hive.server2.authentication.ldap.Domain` – LDAP domain. (Hive 0.12.0 and later.)

See [User and Group Filter Support with LDAP Atn Provider in HiveServer2](#) for other LDAP configuration parameters in Hive 1.3.0 and later.

Set following for `CUSTOM` mode:

`hive.server2.custom.authentication.class` – Custom authentication class that implements the `org.apache.hive.service.auth.PasswordAuthenticationProvider` interface.

For `PAM` mode, see details in [section on PAM](#) below.

Impersonation

By default HiveServer2 performs the query processing as the user who submitted the query. But if the following parameter is set to `false`, the query will run as the user that the `hiveserver2` process runs as.

`hive.server2.enable.doAs` – Impersonate the connected user, default `true`.

To prevent memory leaks in unsecure mode, disable file system caches by setting the following parameters to `true` (see [HIVE-4501](#)):

`fs.hdfs.impl.disable.cache` – Disable HDFS filesystem cache, default `false`.

`fs.file.impl.disable.cache` – Disable local filesystem cache, default `false`.

Integrity/Confidentiality Protection

Integrity protection and confidentiality protection (beyond just the default of authentication) for communication between the Hive JDBC driver and HiveServer2 are enabled (Hive 0.12 onward, see [HIVE-4911](#)). You can use the [SASL QOP](#) property to configure this.

- This is only when Kerberos is used for the HS2 client (JDBC/ODBC application) authentication with HiveServer2.
- `hive.server2.thrift.sasl.qop` in `hive-site.xml` has to be set to one of the valid [QOP](#) values ('auth', 'auth-int' or 'auth-conf').

SSL Encryption

Support is provided for SSL encryption (Hive 0.13 onward, see [HIVE-5351](#)). To enable, set the following configurations in `hive-site.xml`:

`hive.server2.use.SSL` – Set this to `true`.

`hive.server2.keystore.path` – Set this to your keystore path.

`hive.server2.keystore.password` – Set this to your keystore password.

Note



When `hive.server2.transport.mode` is binary and `hive.server2.authentication` is `KERBEROS`, SSL encryption did not work until Hive 2.0. Set `hive.server2.thrift.sasl.qop` to `auth-conf` to enable encryption. See [HIVE-14019](#) for details.

Setting up SSL with self-signed certificates

Use the following steps to create and verify self-signed SSL certificates for use with HiveServer2:

1. Create the self signed certificate and add it to a keystore file using: `keytool -genkey -alias example.com -keyalg RSA -keystore keystore.jks -keysize 2048` Ensure the name used in the self signed certificate matches the hostname where HiveServer2 will run.
2. List the keystore entries to verify that the certificate was added. Note that a keystore can contain multiple such certificates: `keytool -list -keystore keystore.jks`
3. Export this certificate from keystore.jks to a certificate file: `keytool -export -alias example.com -file example.com.crt -keystore keystore.jks`

4. Add this certificate to the client's truststore to establish trust: `keytool -import -trustcacerts -alias example.com -file example.com.crt -keystore truststore.jks`
5. Verify that the certificate exists in truststore.jks: `keytool -list -keystore truststore.jks`
6. Then start HiveServer2, and try to connect with beeline using: `jdbc:hive2://<host>:<port>/<database>;ssl=true;sslTrustStore=<path-to-truststore>;trustStorePassword=<truststore-password>`

Selectively disabling SSL protocol versions

To disable specific SSL protocol versions, use the following steps:

1. Run `openssl ciphers -v` (or the corresponding command if not using openssl) to view all protocol versions.
2. In addition to 1, an additional step of going over the HiveServer2 logs may be required to see all the protocols that the node running HiveServer2 is supporting. For that, search for "SSL Server Socket Enabled Protocols:" in the HiveServer2 log file.
3. Add all the SSL protocols that need to be disabled to `hive.ssl.protocol.blacklist`. Ensure that the property in `hiveserver2-site.xml` does not override that in `hive-site.xml`.

Pluggable Authentication Modules (PAM)

Warning

JPAM library that is used to provide the PAM authentication mode can cause HiveServer2 to go down if a user's password has expired. This happens because of segfault/core dumps from native code invoked by JPAM. Some users have also reported crashes during logins in other cases as well. Use of LDAP or KERBEROS is recommended.

Support is provided for PAM (Hive 0.13 onward, see [HIVE-6466](#)). To configure PAM:

- Download the **JPAM** native library for the relevant architecture.
- Unzip and copy `libjpam.so` to a directory (`<libjmap-directory>`) on the system.
- Add the directory to the `LD_LIBRARY_PATH` environment variable like so: `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<libjmap-directory>`
- For some PAM modules, you'll have to ensure that your `/etc/shadow` and `/etc/login.defs` files are readable by the user running the HiveServer2 process.

Finally, set the following configurations in `hive-site.xml`:

`hive.server2.authentication` – Set this to **PAM**.

`hive.server2.authentication.pam.services` – Set this to a list of comma-separated PAM services that will be used. Note that a file with the same name as the PAM service must exist in `/etc/pam.d`.

Setting up HiveServer2 job credential provider

Starting Hive 2.2.0 onwards (see [HIVE-14822](#)) Hiveserver2 supports job specific hadoop credential provider for MR and Spark jobs. When using encrypted passwords via the Hadoop Credential Provider, HiveServer2 needs to forward enough information to the job configuration so that jobs launched across cluster can read those secrets. Additionally, HiveServer2 may have secrets that the job should not have such as the Hive Metastore database password. If your job needs to access such secrets, like S3 credentials, then you can configure them using the configuration steps below:

1. Create a job-specific keystore using Hadoop Credential Provider API at a secure location in HDFS. This keystore should contain the encrypted key /value pairs of the configurations needed by jobs. Eg: in case of S3 credentials the keystore should contain `fs.s3a.secret.key` and `fs.s3a.access.key` with their corresponding values.
2. The password to decrypt the keystore should be set as a HiveServer2 environment variable called `HIVE_JOB_CREDSTORE_PASSWORD`
3. Set `hive.server2.job.credential.provider.path` to URL pointing to the type and location of keystore created in (1) above. If there is no job-specific keystore, HiveServer2 will use the one set using `hadoop.credential.provider.path` in `core-site.xml` if available.
4. If the password using environment variable set in step 2 is not provided, HiveServer2 will use `HADOOP_CREDSTORE_PASSWORD` environment variable if available.
5. HiveServer2 will now modify the job configuration of the jobs launched using MR or Spark execution engines to include the job credential provider so that job tasks can access the encrypted keystore with the secrets.

`hive.server2.job.credential.provider.path` – Set this to your job-specific hadoop credential provider. Eg: `jceks://hdfs/user/hive/secret/jobcreds.jceks`.

`HIVE_JOB_CREDSTORE_PASSWORD` – Set this HiveServer2 environment variable to your job specific Hadoop credential provider password set above.

Scratch Directory Management

HiveServer2 allows the configuration of various aspects of scratch directories, which are used by Hive to store temporary output and plans.

Configuration Properties

The following are the properties that can be configured related to scratch directories:

- [hive.scratchdir.lock](#)
- [hive.exec.scratchdir](#)

- [hive.scratch.dir.permission](#)
- [hive.start.cleanup.scratchdir](#)

ClearDanglingScratchDir Tool

The tool **cleardanglingscratchdir** can be run to clean up any dangling scratch directories that might be left over from improper shutdowns of Hive, such as when a virtual machine restarts and leaves no chance for Hive to run the shutdown hook.

```
hive --service cleardanglingscratchdir [-r] [-v] [-s scratchdir]
-r          dry-run mode, which produces a list on console
-v          verbose mode, which prints extra debugging information
-s          if you are using non-standard scratch directory
```

The tool tests if a scratch directory is in use, and if not, will remove it. This relies on HDFS write locks to detect if a scratch directory is in use. An HDFS client opens an HDFS file (`$(scratchdir)/inuse.lock`) for writing and only closes it at the time that the session is closed. **cleardanglingscratchdir** will try to open `$(scratchdir)/inuse.lock` for writing to test if the corresponding HiveCli/HiveServer2 is still running. If the lock is in use, the scratch directory will not be cleared. If the lock is available, the scratch directory will be cleared. Note that it might take NameNode up to 10 minutes to reclaim the lease on scratch file locks from a dead HiveCli/HiveServer2, at which point **cleardanglingscratchdir** will be able to remove it if run again.

Web UI for HiveServer2

Version

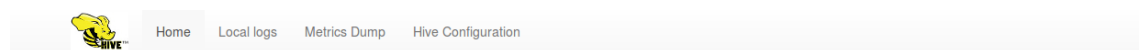


Introduced in Hive 2.0.0. See [HIVE-12338](#) and its sub-tasks.

A Web User Interface (UI) for HiveServer2 provides configuration, logging, metrics and active session information. The Web UI is available at port 10002 (127.0.0.1:10002) by default.

- [Configuration properties](#) for the Web UI can be [customized in hive-site.xml](#), including `hive.server2.webui.host`, `hive.server2.webui.port`, `hive.server2.webui.max.threads`, and others.
- [Hive Metrics](#) can be viewed by using the "Metrics Dump" tab.
- [Logs](#) can be viewed by using the "Local logs" tab.

The interface is currently under development with [HIVE-12338](#).



HiveServer2

Active Sessions

| User Name | IP Address | Operation Count | Active Time (s) | Idle Time (s) |
|-----------------------------|------------|-----------------|-----------------|---------------|
| anonymous | 127.0.0.1 | 1 | 99 | 3 |
| Total number of sessions: 1 | | | | |

Queries

| User Name | Query | State | Elapsed Time (s) |
|----------------------------|---|---------|------------------|
| anonymous | select count(*) from store_sales join store on ss_store_sk = s_store_sk | RUNNING | 6 |
| Total number of queries: 1 | | | |

Software Attributes

| Attribute Name | Value | Description |
|------------------------|---|---|
| Hive Version | 2.0.0-SNAPSHOT, rd4eba26021cfcc47ab5cd75f9664bea27f469a3f | Hive version and revision |
| Hive Compiled | Thu Nov 19 08:08:09 PST 2015, jxiang | When Hive version was compiled and by whom |
| HiveServer2 Start Time | Thu Nov 19 08:15:49 PST 2015 | Date stamp of when this HiveServer2 was started |

Python Client Driver

A Python client driver for HiveServer2 is available at <https://github.com/BradRuderman/pyhs2> (thanks, Brad). It includes all the required packages such as SASL and Thrift wrappers.

The driver has been certified for use with Python 2.6 and newer.

To use the [pyhs2](#) driver:

```
pip install pyhs2
```

and then:

```
import pyhs2

with pyhs2.connect(host='localhost',
                  port=10000,
                  authMechanism="PLAIN",
                  user='root',
                  password='test',
                  database='default') as conn:
    with conn.cursor() as cur:
        #Show databases
        print cur.getDatabases()

        #Execute query
        cur.execute("select * from table")

        #Return column info from query
        print cur.getSchema()

        #Fetch table results
        for i in cur.fetch():
            print i
```

You can discuss this driver on the user@hive.apache.org mailing list.

Ruby Client Driver

A Ruby client driver is available on github at <https://github.com/forward3d/rbhive>.