Installation

OpenTSDB may be compiled from source or installed from a package. Releases can be found on Github.

Runtime Requirements

To actually run OpenTSDB, you'll need to meet the following:

- A Linux system (or Windows with manual building)
- Java Runtime Environment 1.6 or later
- HBase 0.92 or later
- GnuPlot 4.2 or later

Installation

First, you need to setup HBase. If you are brand new to HBase and/or OpenTSDB we recommend you test with a stand-alone instance as this is the easiest to get up and running. The best place to start is to follow the Apache Quick Start guide. Alternatively you could try a packaged distribution such as Cloudera's CDH, Hortonworks HDP or MapR.

Before proceeding with OpenTSDB, make certain that Zookeeper is accessible. One method is to simply telnet to the proper port and execute the `stats` command.

```
root@host:~# telnet localhost 2181
Trying 127.0.0.1...
Connected to myhost.
Escape character is '^]'.
stats
Zookeeper version: 3.4.3-cdh4.0.1--1, built on 06/28/2012 23:59 GMT
Clients:

Latency min/avg/max: 0/0/677
Received: 4684478
Sent: 4687034
Outstanding: 0
Zxid: 0xb00187dd0
Mode: leader
Node count: 127182
Connection closed by foreign host.
```

If you can't connect to Zookeeper, check IPs and name resolution. HBase can be finicky.

If HBase is running, you can choose to install OpenTSDB from a package (available under Releases in Github) or from source using GIT or a source tarball.

Compiling From Source

Compilation requirements include:

- A Linux system
- Java Development Kit 1.6 or later
- GnuPlot 4.2 or later
- Autotools
- Make
- Python
- Git
- An Internet connection

Download the latest version using `git clone` command or download a release from the site or Github. Then just run the `build.sh` script. This script helps run all the processes needed to compile OpenTSDB:

it runs `./bootstrap` (only once, when you first check out the code), followed by `./configure` and `make`. The output of the build process is put into a `build` folder and JARs required by OpenTSDB will be downloaded.

```
git clone git://github.com/OpenTSDB/opentsdb.git
cd opentsdb
./build.sh
```

If compilation was successfully, you should have a tsdb jar file in `./build` along with a `tsdb` script. You can now execute command-line tool by invoking `./build/tsdb` or you can run `make install` to install OpenTSDB on your system. Should you ever change your mind, there is also `make uninstall`, so there are no strings attached.

If you need to distribute OpenTSDB to machines without an Internet connection, call `./build.sh dist` to wrap the build directory into a tarball that you can then copy to additional machines.

Source Layout

There are four main branches in the GIT repo.

- **master** This is the current release of OpenTSDB. It has been marked as stable and in between releases, only bug-fixes are applied. This is always suitable for running in production.
- **maintenance** This was the previous release of OpenTSDB and may have bug fixes applied in between releases.
- **next** The current version of OpenTSDB under development. This version is suitable for development environments and may have new features as well as bug fixes. If a release candidate has been cut, this branch will only contain bug fixes and is suitable for a staging environment.
- **put** The next version of OpenTSDB that may have new features when a release candidate is present in the **next** branch.

Debian Package

You can generate a Debian package by calling `sh build.sh debian`. The package will be located at `./build/opentsdb-2.x.x/opentsdb-2.x.x_all.deb`. Then simply distribute the package and install it as you regularly would. For example `dpkg -i opentsdb-2.0.0_all.deb`.

The Debian package will create the following directories:

- /etc/opentsdb - Configuration files
- /tmp/opentsdb - Temporary cache files
- /usr/share/opentsdb - Application files
- /usr/share/opentsdb/bin - The "tsdb" startup script that launches a TSD or command line tools
- /usr/share/opentsdb/lib - Java JAR library files
- /usr/share/opentsdb/plugins - Location for plugin files and dependencies
- /usr/share/opentsdb/static - Static files for the GUI
- /usr/share/opentsdb/tools - Scripts and other tools
- /var/log/opentsdb - Logs

Installation includes an init script at `/etc/init.d/opentsdb` that can start, stop and restart OpenTSDB. Simply call `service opentsdb start` to start the tsd and `service opentsdb stop` to gracefully shutdown. Note after install, the tsd will not be running so that you can edit the configuration file. Edit the config file, then start the TSD.

The Debian package also creates an `opentsdb` user and group for the TSD to run under for increased security. TSD only requires write permission to the temporary and logging directories. If you can't use the default locations, please change them in `/etc/opentsdb/opentsdb.conf` and `/etc/opentsdb/logback.xml` respectively and apply the proper permissions for the `opentsdb` user.

If you install OpenTSDB for the first time, you'll need to create the HBase tables using the script located at `/usr/share/opentsdb/tools/create_table.sh`. Follow the steps below.

Create Tables

If this is the first time that you are running OpenTSDB with your HBase instance, you first need to create the necessary HBase tables. A simple script is provided to create the proper tables with the ability to enable or disable compression. Execute:

```
env COMPRESSION=NONE HBASE_HOME=path/to/hbase-0.94.X ./src/create_table.sh
```

where the `COMPRESSION` value is either `NONE`, `LZO`, `GZIP` or `SNAPPY`. This will create four tables: `tsdb`, `tsdb-uid`, `tsdb-tree` and `tsdb-meta`. If you're just evaluating OpenTSDB, don't worry about compression for now. In production and at scale, make sure you use a valid compression library as it will save on storage tremendously.

Start a TSD

OpenTSDB 2.3 works off a configuration file that is shared between the daemon and command line tools. If you compiled from source, copy the `./src/opentsdb.conf` file to a proper directory as documented in Configuration and edit the following, required settings:

- **tsd.http.cachedir** - Path to write temporary files to
- **tsd.http.staticroot** - Path to the static GUI files found in `./build/staticroot`
- **tsd.storage.hbase.zk_quorum** - If HBase and Zookeeper are not running on the same machine, specify the host and port here.

With the config file written, you can start a tsd with the command:

```
./build/tsdb tsd
```

Alternatively, you can also use the following commands to create a temporary directory and pass in only command line flags:

```
tsdtmp=${TMPDIR-'/tmp'}/tsd    # For best performance, make sure
mkdir -p "$tsdtmp"             # your temporary directory uses tmpfs
./build/tsdb tsd --port=4242 --staticroot=build/staticroot --cachedir="$tsdtmp"
--zkquorum=myhost:2181
```

At this point you can access the TSD's web interface through http://127.0.0.1:4242 (if it's running on your local machine).

Note

The **Cache Directory** stores temporary files generated when a graph is requested via the built-in GUI. These files should be purged periodically to free up space. OpenTSDB doesn't clean up after itself at this time but there is a script that should be run as a cron at least once a day located at `tools/clean_cache.sh`.

Upgrading from 1.x

OpenTSDB 2.3 is fully backwards compatible with 1.x data. We've taken great pains to make sure you can download 2.3, compile, stop your old TSD and start the new one. Your existing tools will read and write to the TSD without a problem. 2.3 introduces two new tables to HBase schema for storing meta-data. From the directory where you downloaded the source (or the tools directory if installed with the Debian package), execute:

```
env COMPRESSION=NONE HBASE_HOME=path/to/hbase-0.94.X ./src/upgrade_1to2.sh
```

where `COMPRESSION` is the same as your existing production table compression format.

While you can start a 2.3 TSD with the same command line options as a 1.0 TSD, we highly recommend that you create a configuration file based on the config included at `./src/opentsdb.conf`. Or if you install from a package, you'll want to edit the included default config. The config file includes many more options than are accessible via command line and the file is shared with CLI tools. See

Configuration for details.

You do not have to upgrade all of your TSDs to 2.3 at the same time. Some users upgrade their read-only TSDs first to gain access to the full HTTP API and test the new features. Later on you can upgrade the write-only TSDs at leisure. You can also perform a rolling upgrade without issues. Simply stop traffic to one TSD, upgrade it, restore traffic, and continue on until you have upgraded all of your TSDs.

If you do perform a rolling upgrade where you have multiple TSDs, heed the following warning:

Warning

Do not write **Annotations** or **Data point with Millisecond Timestamps** while you run a mixture of 1.x and 2.x. Because these data are stored in the same rows as regular data points, they can affect compactions and queries.

Before upgrading to 2.x, you may want to upgrade all of your TSDs to OpenTSDB 1.2. This release is fully forwards compatible in that it will ignore annotations and millisecond timestamps and operate as expected. With 1.2 running, if you accidentally record an annotation or millisecond data point, your 1.2 TSDs will operate normally.

Upgrading from 2.x to a Later 2.x

In general, upgrading within a single major release branch is simply a matter of updating the binaries or package and restarting a TSD. Within a branch we'll maintain settings, APIs and schema. However new features may be added with each minor version that include new configuration settings with useful defaults.

Note

The exception so far has been the introduction of salted rows in 2.2.0. Disabled by default, using this feature requires creating a new HBase table with a new set of pre-splits and modifying the configuration of every TSD to use the new table with salting enabled. The schema for salted and unsalted tables is incompatible so if users have a lot of data in a previous table, it may be best to leave a few TSDs running to query against the old table and new TSDs to write to and read from the new salted table. For smaller amounts of data, the scan tool can be used to export and re-import your data.

Note

Likewise with 2.3, the introduction of new backends (Bigtable or Cassandra) requires setting up new storage tables and migrating data.

Downgrading

Because we've worked hard to maintain backwards compatibility, you can turn off a 2.x TSD and restart your old 1.x TSD. The only exceptions are if you have written annotations or milliseconds as you saw in the warning above. In these cases you must downgrade to 1.2 or later. You may also delete the `tsdb-tree` and `tsdb-meta` tables if you so desire.

Downgrades within a major version are idempotent.

Warning

If you wrote data using a salted table or changed the UID widths for metrics, tag keys or tag values then you cannot downgrade. Create a new table and export the data from the old table, then re-write the data to the new table using the older TSD version.