**Cassandra: Install Cassandra**

**Step 1 of 9**

**Step 1**

**Install the Binary Distribution**

While there are a number of options available for installing Cassandra on various operating systems, let's start your journey by downloading the Apache distribution from [the Apache website](https://cassandra.apache.org/) so you can get a good look at what's inside:

wget https://dlcdn.apache.org/cassandra/4.1.0/apache-cassandra-4.1.0-bin.tar.gz

You can unpack the compressed file using any regular ZIP utility. On Unix-based systems such as Linux or macOS, gzip extraction utilities should be preinstalled. On some platforms, you might have to extract the ZIP file and the TAR file in separate steps, but you can do it in a single step on this Linux machine:

tar xvzf apache-cassandra-4.1.0-bin.tar.gz

Now you have a folder on your filesystem called *apache-cassandra-x.x.x*.

**Step 2**

**Investigate the Contents**

Now let's explore the files under the expanded folder *apache-cassandra-x.x.x*, using the file explorer to the upper right:

* apache-cassandra-4.1.0/NEWS.txt includes the release notes describing features included in the current and prior releases
* apache-cassandra-4.1.0/CHANGES.txt is similar but focuses on bug fixes. You'll want to make sure to review these files whenever you are upgrading to a new version so you know what changes to expect.
* apache-cassandra-4.1.0/LICENSE.txt contains the Apache 2.0 license
* apache-cassandra-4.1.0/NOTICE.txt contains copyright notices for Cassandra and included software

Next, look around in the directories and see what's there:

* *bin*: Contains the executables to run Cassandra as well as clients, including the query language shell (cqlsh). It also has scripts to run the nodetool, which is a utility for inspecting a cluster to determine whether it is properly configured, and to perform a variety of maintenance operations. The directory also contains several utilities for performing operations on SSTables, the files in which Cassandra stores its data on disk.
* *conf*: Contains the files for configuring your Cassandra instance. The configuration files you might use most frequently include the *cassandra.yaml* file, which is the primary configuration for running Cassandra, and the *logback.xml* file, which lets you change the logging settings to suit your needs. Additional files can be used to configure Java Virtual Machine (JVM) settings, the network topology, metrics reporting, archival and restore commands, and triggers.
* *doc*: Includes a getting started guide, an architectural overview, and instructions for configuring and operating Cassandra.
* *javadoc*: The documentation website generated using Java's JavaDoc tool. It's helpful if you want to see how the code is laid out.
* *lib*: The external libraries that Cassandra needs to run. For example, it uses two different JSON serialization libraries, the Google collections project, and several Apache Commons libraries.
* *pylib*: Python libraries that are used by cqlsh.
* *tools*: Tools that are used to maintain your Cassandra nodes.

Exemple :

cat apache-cassandra-4.1.0/NEWS.txt

#### Step 3

# Start the Cassandra Daemon

Navigate into the Cassandra directory:

cd apache-cassandra-4.1.0

Run the following command to start your server:

bin/cassandra -R

The -R is required when running Cassandra from a superuser account. You could also use the -f switch to tell Cassandra to stay in the foreground instead of running as a background process. In either case, the logs will append to the logs/system.log file.

#### Step 4

# Examine the Log File

In a clean installation, you should see quite a few log statements as the server gets running. The exact syntax of logging statements will vary depending on the release you're using, but there are a few highlights you can look for. Search for cassandra.yaml:

grep -m 1 -C 1 "cassandra.yaml" logs/system.log

These log statements indicate the location of the cassandra.yaml file containing the configured settings. The Node configuration statement lists out the settings read from the config file.

Now search for JVM:

grep -m 1 -C 1 "JVM" logs/system.log

These log statements provide information describing the JVM being used, including memory settings.

Next, search for the versions in use—Cassandra version, CQL version, and Native protocol supported versions:

grep -m 1 -A 2 "Cassandra version" logs/system.log

You can also find statements where Cassandra is initializing internal data structures, such as caches:

grep -m 4 "CacheService.java" logs/system.log

You can also search for terms like JMX, gossip, and listening:

grep -m 1 "JMX" logs/system.log

grep -m 1 "gossip" logs/system.log

grep -m 1 "listening" logs/system.log

These log statements indicate the server is beginning to initiate communications with other servers in the cluster and expose publicly available interfaces. By default, the management interface via the Java Management Extensions (JMX) is disabled for remote access.

Finally, search for state jump:

grep -m 1 -C 1 "state jump" logs/system.log

#### Step 5

# Connect with cqlsh

Now that you have a Cassandra installation up and running, let's give it a quick try to make sure everything is set up properly. You'll use the cqlsh to connect to your server and have a look around:

bin/cqlsh

Because you did not specify a node to which you wanted to connect, the shell helpfully checks for a node running on the local host and finds the node you started earlier. The shell also indicates that you're connected to a Cassandra server cluster called “Test Cluster.” That's because this cluster of one node at localhost is set up for you by default.

Let's take a quick tour of cqlsh to learn what kinds of commands you can send to the server. You'll see how to use the basic environment commands and how to do a round trip of inserting and retrieving some data.

The cqlsh commands are all case insensitive. For these examples, we adopt the convention of uppercase to be consistent with the way the shell describes its own commands in help topics and output.

#### Step 6

# The HELP Command

To get help for cqlsh, type HELP or ? to see the list of available commands:

HELP

To get additional documentation about a particular command, type HELP <command>.

Many cqlsh commands may be used with no parameters, in which case they print out the current setting. Examples include CONSISTENCY, EXPAND, and PAGING.

You can use the SHOW VERSION command to see the versions of Cassandra and the CQL Native Protocol in use:

SHOW VERSION

You may have noticed that this version info is printed out when cqlsh starts.

#### Step 7

# The DESCRIBE Command

To learn more about the cluster you're working in, type:

DESCRIBE CLUSTER

To see which keyspaces are available in the cluster, issue this command:

DESCRIBE KEYSPACES

#### Step 8

# cqlsh Command History

You might have noticed that you can navigate through commands you've executed previously in cqlsh with the up and down arrow keys. If not, you can give it a try now.

When you're done, exit cqlsh using the EXIT command:

EXIT;

Your command history is stored in a file called cqlsh＿history, which is located in a hidden directory called .cassandra within your home directory. If you're interested, you can open this file using the file browser to the upper right.

This acts like your bash shell history, listing the commands in a plain-text file in the order Cassandra executed them. Nice!

**Step 9**

**Stop Cassandra**

Now that you've successfully started a Cassandra server, you might be wondering how to stop it. You might have noticed the stop-server command in the *bin* directory. Let's try running that command:

bin/stop-server

So you see that the server has not been stopped, but instead, you are directed to read the script. If you examine this file using the editor to the upper right, you'll learn that the way to stop Cassandra is to kill the JVM process that is running Cassandra. The file suggests a couple of different techniques by which you can identify the JVM process and kill it.

The first technique is to start Cassandra using the -p option, which provides Cassandra with the name of a file to which it should write the process identifier (PID) upon starting up. This is arguably the most straightforward approach to making sure you kill the right process.

However, because you did not start Cassandra with the -p option, you'll need to find the process yourself and kill it. The script suggests using pgrep to locate processes for the current user containing the term “cassandra,” but it's just as simple to use pkill:

user=`whoami`

pkill -u $user -f cassandra