# Apache Cassandra 快速入门指南（Quick Start）

我们在这篇文章简单介绍了 Apache Cassandra 是什么，以及有什么值得关注的特性。本文将简单介绍 Apache Cassandra 的安装以及简单使用，可以帮助大家快速了解 Apache Cassandra。

我们到 Apache Cassandra 的官方网站下载最新版本的 Cassandra，在本文写作时最新版本的 Cassandra 为 3.11.4。Apache Cassandra 可以在 Linux、Unix、Mac OS 以及 Windows 上进行安装，为了简便起见，本文以 CentOS 为例进行介绍。


如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

## 下载、安装并启动 Cassandra

因为本文只是简单介绍 Apache Cassandra 的使用，所以本文仅安装单机版的 Cassandra，在生产环境下应该部署成分布式模式。可以使用下面的命令下载和解压相关的压缩文件：

```
$ wget http://mirror.bit.edu.cn/apache/cassandra/3.11.4/apache-cassandra-3.11.4-bin.tar.gz
$ tar -zxf apache-cassandra-3.11.4-bin.tar.gz
$ cd apache-cassandra-3.11.4
```

在 apache-cassandra-3.11.4 目录下有很多文件：

```
[iteblog@www.iteblog.com apache-cassandra-3.11.4]# ll
total 528
drwxr-xr-x 2 iteblog iteblog   4096 Apr  2 21:12 bin
-rw-r--r-- 1 iteblog iteblog   4832 Feb  3 06:09 CASSANDRA-14092.txt
-rw-r--r-- 1 iteblog iteblog 366951 Feb  3 06:09 CHANGES.txt
drwxr-xr-x 3 iteblog iteblog   4096 Apr  2 21:12 conf
drwxr-xr-x 4 iteblog iteblog   4096 Apr  2 21:12 doc
drwxr-xr-x 2 iteblog iteblog   4096 Apr  2 21:12 interface
drwxr-xr-x 3 iteblog iteblog   4096 Apr  2 21:12 javadoc
drwxr-xr-x 4 iteblog iteblog   4096 Apr  2 21:12 lib
-rw-r--r-- 1 iteblog iteblog  11609 Feb  3 06:09 LICENSE.txt
-rw-r--r-- 1 iteblog iteblog 112586 Feb  3 06:09 NEWS.txt
-rw-r--r-- 1 iteblog iteblog   2811 Feb  3 06:09 NOTICE.txt
drwxr-xr-x 3 iteblog iteblog   4096 Apr  2 21:12 pylib
drwxr-xr-x 4 iteblog iteblog   4096 Apr  2 21:12 tools
```

各个文件或目录介绍如下：

- bin：这个目录下包含了启动 Cassandra 以及客户端相关操作的可执行文件，包括 query language shell（cqlsh）以及命令行界面（CLI）等客户端。同时还包含运行 nodetool 的相关脚本，操作 SSTables 的工具等等。
- conf：这个目录下面包含了 Cassandra 的配置文件。必须包含的配置文件包括：assandra.yaml 以及 logback.xml，这两个文件分别是运行 Cassandra 必须包含的配置文件以及日志相关配置文件。同时还包含 Cassandra 网络拓扑配置文件等。
- doc：这个目录包含 CQL 相关的 html 文档。
- interface：这个文件夹下面只包含一个名为 cassandra.thrift 的文件。这个文件定义了基于 Thrift 语法的 RPC API，这个 Thrift 主要用于在 Java, C++, PHP, Ruby, Python, Perl, 以及 C# 等语言中创建相关客户端，但是在 CQL 出现之后，Thrift API 在 Cassandra 3.2 版本开始标记为 deprecated，并且会在 Cassandra 4.0 版本删除。
- javadoc：这个文件夹包含使用 JavaDoc 工具生成的 html 文档。
- lib：这个目录包含 Cassandra 运行时需要的所有外部库。
- pylib：这个目录包含 cqlsh 运行时需要使用的 Python 库。
- tools：这个目录包含用于维护 Cassandra 节点的相关工具。
- NEWS.txt：这个文件包含当前及之前版本的 release notes 相关信息。
- CHANGES.txt：这个文件主要包含一些 bug fixes 信息。

## 启动 Cassandra

上面已经简单介绍了 Cassandra 发行包里面的一些文件和目录用途。因为我们主要简单介绍 Cassandra 的使用，所以我们使用默认的配置。下面我们来启动 Cassandra 服务，具体如下：

```
[iteblog@www.iteblog.com apache-cassandra-3.11.4]# bin/cassandra
```

运行上面命令会在命令行里面输出一堆的日志，但是我们如何判断 cassandra 服务已经启动了呢？答案是使用 nodetool 工具，如下：

```
[iteblog@www.iteblog.com apache-cassandra-3.11.4]# bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens       Owns (effective)  Host ID                               Rack
```

UN 127.0.0.1 160.88 KiB 256 100.0% 49f4470f-396b-4d50-bcd6-3da2d9370167 r
ack1

如果我们看到上面的信息，那么我们的测试服务已经正常启动了。而且会在 apache-cassandra-3.11.4 目录下生成 data 和 logs 两个目录。

## 使用 CQL Shell

上面我们已经启动了 Cassandra 服务，我们可以使用 CQL Shell
来进行一些操作。从名字就可以看出，CQL（Cassandra Query Language）其实和我们熟悉的 SQL 很类似，我们可以通过它使用类似 SQL 的语言来和 Cassandra
进行交互。需要注意的是，CQL 和 SQL 是不兼容的，CQL 缺少 SQL 的一些关键功能，比如 JOIN 等，这个在 Cassandra 下不能实现；同时，CQL 也不是 SQL 的子集。为了使用 CQL
，可以使用下面命令：

```
[iteblog@www.iteblog.com apache-cassandra-3.11.4]# bin/cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

在启动 cqlsh 的时候我们并没有指定需要连接的节点以及端口，这种情况下 cqlsh
会自动探测本机及相关端口，因为我们在前面已经启动了 Cassandra 服务，所以 cqlsh
可以正确连接到这个集群。从上面的命令可以看出 cqlsh 连接到名为 Test Cluster
的集群，这是由 conf/cassandra.yaml 文件里面的 cluster_name 参数决定的，默认值为 Test Cluster。

当然，我们也可以在启动 cqlsh 的时候指定节点和相应的端口，如下：

```
[iteblog@www.iteblog.com apache-cassandra-3.11.4]# bin/cqlsh localhost 9042
```

上面的命令执行效果和不指定一样。我们也可以将节点和端口相关的信息保存到环境变量 $CQLSH_HOST 和 $CQLSH_PORT
里面，这个在我们需要经常连接到特定节点的情况下非常有用。更多关于 cqlsh
命令支持的参数可以使用 bin/cqlsh -help。

## 基本的 cqlsh 命令

cqlsh 支持很多操作 Cassandra 的基本命令，我们可以在 cqlsh 里面使用 HELP 或？命令查看所有支持的命令：

```
cqlsh> HELP

Documented shell commands:
===========================
CAPTURE  CLS      COPY  DESCRIBE  EXPAND  LOGIN   SERIAL  SOURCE   UNICODE
CLEAR    CONSISTENCY  DESC  EXIT    HELP    PAGING  SHOW    TRACING

CQL help topics:
================
AGGREGATES            CREATE_KEYSPACE         DROP_TRIGGER     TEXT
ALTER_KEYSPACE        CREATE_MATERIALIZED_VIEW  DROP_TYPE      TIME
ALTER_MATERIALIZED_VIEW  CREATE_ROLE          DROP_USER        TIMESTAMP
ALTER_TABLE           CREATE_TABLE            FUNCTIONS        TRUNCATE
ALTER_TYPE            CREATE_TRIGGER          GRANT            TYPES
ALTER_USER            CREATE_TYPE             INSERT           UPDATE
APPLY                 CREATE_USER             INSERT_JSON      USE
ASCII                 DATE                    INT              UUID
BATCH                 DELETE                  JSON
BEGIN                 DROP_AGGREGATE          KEYWORDS
BLOB                  DROP_COLUMNFAMILY       LIST_PERMISSIONS
BOOLEAN               DROP_FUNCTION           LIST_ROLES
COUNTER               DROP_INDEX              LIST_USERS
CREATE_AGGREGATE      DROP_KEYSPACE           PERMISSIONS
CREATE_COLUMNFAMILY   DROP_MATERIALIZED_VIEW  REVOKE
CREATE_FUNCTION       DROP_ROLE               SELECT
CREATE_INDEX          DROP_TABLE              SELECT_JSON
```

如果需要查看特定命令的帮助，可以使用 HELP <command>。需要注意的是，很多 cqlsh 命令并不接收相关的参数，当我们使用这些命令时，其输出为当前的设置，比如 CONSISTENCY, EXPAND 和 PAGING 命令，如下：

```
cqlsh> CONSISTENCY
Current consistency level is ONE.
cqlsh> EXPAND
Expanded output is currently disabled. Use EXPAND ON to enable.
cqlsh> PAGING
Query paging is currently enabled. Use PAGING OFF to disable
```

Page size: 100

## 在 cqlsh 里面查看环境变量

我们可以使用 DESCRIBE
命令，来查看一些集群的一些环境变量的值。下面命令查看当前集群的情况

```
cqlsh> DESCRIBE CLUSTER;

Cluster: Test Cluster
Partitioner: Murmur3Partitioner
```

DESCRIBE CLUSTER 显示了集群的名字以及采用的 Partitioner ，Cassandra 1.2 版本开始默认为 Murmur3Partitioner，其他可选的 Partitioner 有 RandomPartitioner（Cassandra 1.2 版本之前默认的 Partitioner）、OrderPreservingPartitioner 以及 ByteOrderedPartitioner 等。

如果我们需要查看集群里面可用的 keyspaces，可以使用下面命令：

```
cqlsh> DESCRIBE KEYSPACES;

system_traces  system_schema  system_auth  system  system_distributed
```

上面命令将系统自带的 keyspaces 都显示出来了，如果我们自己创建了 keyspaces，也会在这里面显示。

可以使用下面命令查看 cqlsh、Cassandra 以及 protocol 的版本：

```
cqlsh> SHOW VERSION;
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
```

事实上，这些版本在启动 cqlsh 的时候就显示了。

## 通过 cqlsh 创建 keyspace

Cassandra 里面的 keyspace 和关系型数据库里面的 database 概念类似的，一个 keyspace 可以包含一个或多个 tables 或 column families。当我们启动 cqlsh 时没有指定 keyspace，那么命令提示符为 cqlsh>，我们可以使用 CREATE KEYSPACE 命令来创建 keyspace，具体如下：

```
cqlsh> CREATE KEYSPACE iteblog_keyspace WITH replication = {'class': 'SimpleStrategy', 'replica
tion_factor': 1};
cqlsh>
```

上面命令创建了名为 iteblog_keyspace 的 keyspace；并且采用 SimpleStrategy 进行副本复制，因为我们这个测试集群只有单个节点，所以这里设置的副本因子（replication factor）为 1。如果是生产环境，千万别把副本因子设置为 1，比较常见的副本因子为 3。其他可选的副本复制策略出了 SimpleStrategy 还有 NetworkTopologyStrategy 和 OldNetworkTopologyStrategy，具体什么含义这里还不深入介绍，后面会起单独一篇文章进行详细介绍。

注意，cqlsh 自带了命令提示功能，当我们输入 CREATE KEYSPACE iteblog_keyspace 时，按上键盘上的 Tab 键，cqlsh 会自动给我们不全到 CREATE KEYSPACE iteblog_keyspace WITH replication = {'class': '，这时候我们再按 Tab 键，会显示出支持的所有副本复制策略。具体大家可以去试试。

创建完 keyspace 之后，我们可以使用 DESCRIBE KEYSPACE 命令来查看这个 keyspace：

```
cqlsh> DESCRIBE KEYSPACE  iteblog_keyspace;

CREATE KEYSPACE iteblog_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_fa
ctor': '1'}  AND durable_writes = true;
```

现在我们可以使用 USE 命令来切换到这个 keyspace ：

```
cqlsh> USE iteblog_keyspace;
cqlsh:iteblog_keyspace>
```

从上面的输出可以看出，keyspace 已经切换到 iteblog_keyspace 了。

## 通过 cqlsh 创建表

接下来，我们通过 cqlsh 来创建一张表：

```
cqlsh> use iteblog_keyspace;
cqlsh:iteblog_keyspace> CREATE TABLE iteblog_user (first_name text , last_name text, PRIMARY
 KEY (first_name)) ;
```

通过上面的命令，我们在 iteblog_keyspace 下面创建了一张名为 iteblog_user 的表。其中包含了 first_name 和 last_name 两个字段，类型都是 text，并且 first_name 是这张表的 PRIMARY KEY。当然，我们也可以通过下面命令在 iteblog_keyspace 里面建表：

```
cqlsh> CREATE TABLE iteblog_keyspace.iteblog_user(first_name text , last_name text, PRIMARY
KEY (first_name)) ;
```

效果和上面一样。我们可以使用 DESCRIBE TABLE 命令查看建表语句：

```
cqlsh:iteblog_keyspace> DESCRIBE TABLE iteblog_user;

CREATE TABLE iteblog_keyspace.iteblog_user (
    first_name text PRIMARY KEY,
    last_name text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStra
tegy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.L
Z4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

cqlsh:iteblog_keyspace>
```

DESCRIBE TABLE 命令将建表语句以格式化的形式显示出来。除了我们制定的设置，还包含了许多默认的设置，这里我们先不纠结这些设置的含义。

## 通过 cqlsh 往表里面读写数据

到现在，我们已经创建好 keyspace 和 table
了，我们可以往表里面插入一些数据，看下一切是不是正常。

```
cqlsh:iteblog_keyspace> INSERT INTO iteblog_user (first_name, last_name) VALUES ('iteblog', 'Hadoop');
cqlsh:iteblog_keyspace> INSERT INTO iteblog_user (first_name, last_name) VALUES ('Zhang', 'San');
cqlsh:iteblog_keyspace> INSERT INTO iteblog_user (first_name) VALUES ('Wu');
```

上面语句我们往 iteblog_user 表里面插入三条数据，其中最后一条数据只指定了 key，last_name 没有值。现在我们可以使用 SELECT COUNT 语句查看上面的数据是否插入成功

```
cqlsh:iteblog_keyspace> SELECT COUNT(*) FROM iteblog_user;

 count
-------
     3

(1 rows)

Warnings :
Aggregation query used without partition key
```

可以看出 iteblog_user 表里面已经有3条数据了。我们可以使用下面命令将这条数据查询出来：

```
cqlsh:iteblog_keyspace> SELECT * FROM iteblog_user;

 first_name | last_name
------------+-----------
   iteblog |    Hadoop
        Wu |      null
     Zhang |       San

(3 rows)
cqlsh:iteblog_keyspace> SELECT * FROM iteblog_user WHERE first_name='iteblog';
```

```
 first_name | last_name
------------+-----------
   iteblog |    Hadoop

(1 rows)
cqlsh:iteblog_keyspace>
```

可以看出，first_name 为 Wu 对应的 last_name 没数据直接显示 null 了，在 Cassandra 里面的这个代表对应的列没有数据，在底层存储是不占用空间的，而在常见的关系型数据库里面是占一定空间的。

注意，在 cqlsh 里面查询数据如果超过 10,000 行，那么只会显示 10,000，这是 cqlsh 的限制。

我们可以使用 DELETE 命令删除一些列，比如我们删除 last_name 列，

```
cqlsh:iteblog_keyspace> DELETE last_name FROM iteblog_user WHERE first_name='iteblog';
cqlsh:iteblog_keyspace> SELECT * FROM iteblog_user WHERE first_name='iteblog';

 first_name | last_name
------------+-----------
   iteblog |      null

(1 rows)
cqlsh:iteblog_keyspace>
```

可以看出 last_name 列已经成功被删除了。

我们也可以删除一整行数据，如下：

```
cqlsh:iteblog_keyspace> DELETE FROM iteblog_user WHERE first_name='iteblog';
cqlsh:iteblog_keyspace> SELECT * FROM iteblog_user WHERE first_name='iteblog';

 first_name | last_name
------------+-----------

(0 rows)
cqlsh:iteblog_keyspace>
```

可以看到 key 为 iteblog 的数据已经成功被删除了。

## insert/update 相当于 upsert

如果我们插入数据对应的 key 在 Cassandra 已经存在了，这时候 Cassandra
并不会在原来数据位置上修改数据，而是会新写入一份数据，旧的数据会被 Cassandra 删除。

```
cqlsh:iteblog_keyspace> INSERT INTO iteblog_user (first_name, last_name) VALUES ('Wu', 'Shi');
cqlsh:iteblog_keyspace> SELECT * FROM iteblog_user;

 first_name | last_name
------------+-----------
         Wu |      Shi
      Zhang |       San

(2 rows)
```

可以看见，key 为 Wu 的数据对应的 last_name 已经有值了。

如果我们使用 UPDATE 命令往表里面更新不存在的数据会发生什么呢？答案是会插入新的数据。

```
cqlsh:iteblog_keyspace> SELECT * FROM iteblog_user;

 first_name | last_name
------------+-----------
         Wu |      Shi
      Zhang |       San

(2 rows)
cqlsh:iteblog_keyspace> UPDATE iteblog_user SET last_name = 'Si' WHERE first_name = 'Li';
cqlsh:iteblog_keyspace> SELECT * FROM iteblog_user;

 first_name | last_name
------------+-----------
         Wu |      Shi
      Zhang |       San
         Li |        Si

(3 rows)
cqlsh:iteblog_keyspace>
```

可见，key 为 Li 的数据被插入到表中了，更新之前不存在。

## 清空或删除表

如果我们确实想清空一张表，我们也可以使用 TRUNCATE 命令；使用 DROP TABLE 命令可以删除一张表。

```
cqlsh:iteblog_keyspace> TRUNCATE iteblog_user;
cqlsh:iteblog_keyspace> DROP TABLE iteblog_user;
```

到目前为止，我们已经学会了 cqlsh 的一些简单的命令。后面我们将介绍 Cassandra 底层的数据模型，敬请关注。

细心的同学可能已经发现我们在 cqlsh 里面移动键盘里面的上下键可以看到过去敲过的命令。这是因为 Cassandra 会在用户的 home 目录下生成名为 .cassandra 的文件夹，里面有个 cqlsh_history 文件，里面以文本形式记录了我们执行的命令，这个和 Linux 的 .bash_history 文件类似。

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接: 【】（）