Apache Pig & Hive Installation Guide

Complete Setup for WSL2 Ubuntu Environment

Document Version: 1.0
Last Updated: November 25, 2025
Scope: Apache Pig 0.17.0 & Apache Hive 3.1.3 on WSL2 Ubuntu
Purpose: Master reference for complete environment rebuilding

---

Table of Contents

Phase 1: Download and Install Software

Context

This phase downloads Apache Pig and Apache Hive binaries from official Apache repositories and extracts them to system directories.

Key Decision: We use Hive 3.1.3 (from Archive) instead of the latest Hive 4.0 because version 4.0 removes MapReduce support, which is essential for your current environment.

Step 1: Install Apache Pig (0.17.0)

1.1 Download Apache Pig Binary

```
wget -c https://downloads.apache.org/pig/pig-0.17.0/pig-0.17.0.tar.gz
```

Context:

- `-c` flag resumes interrupted downloads
- Downloads 0.17.0 stable release from Apache's main repository

1.2 Extract and Move to System Directory

```
tar -xzvf pig-0.17.0.tar.gz
sudo mv pig-0.17.0 /usr/local/pig
```

Context:

- `tar -xzvf` extracts the compressed archive
- Moves to `/usr/local/pig` for system-wide access
- Requires `sudo` privileges for `/usr/local/` directory

Step 2: Install Apache Hive (3.1.3 - Archive)

2.1 Download Apache Hive Binary from Archive

```
wget https://archive.apache.org/dist/hive/hive-3.1.3/apache-hive-3.1.3-bin.tar.gz
```

Context:

- Uses **Archive link** instead of current releases
- Hive 3.1.3 maintains full MapReduce compatibility
- Latest Hive versions remove MapReduce, causing incompatibility with WSL2 environments

2.2 Extract and Move to System Directory

```
tar -xzvf apache-hive-3.1.3-bin.tar.gz
sudo mv apache-hive-3.1.3-bin /usr/local/hive
```

Context:

- Creates `/usr/local/hive` installation directory
- Ensures both tools are in standard system locations for easy PATH management

---

Phase 2: Environment Variables

Context

Ubuntu needs to know the installation paths of Pig and Hive. This section configures your shell environment to automatically locate these tools and their dependencies.

Step 3: Configure ~/.bashrc File

3.1 Open the Configuration File

```
nano ~/.bashrc
```

Context:

- `~/.bashrc` is the shell configuration file loaded on every terminal session
- `nano` opens a simple text editor
- Changes here persist across sessions

3.2 Add Environment Variables

Append these lines to the **end** of `~/.bashrc`:

```
# --- PIG VARIABLES ---
export PIG_HOME=/usr/local/pig
export PATH=$PATH:$PIG_HOME/bin
export PIG_CLASSPATH=$HADOOP_HOME/etc/hadoop

# --- HIVE VARIABLES ---
export HIVE_HOME=/usr/local/hive
export PATH=$PATH:$HIVE_HOME/bin
export HIVE_CONF_DIR=$HIVE_HOME/conf
```

Variable Explanations:

Variable	Purpose	Value
`PIG_HOME`	Root directory of Pig installation	`/usr/local/pig`
`PIG_CLASSPATH`	Hadoop configuration path for Pig	`$HADOOP_HOME/etc/hadoop`
`HIVE_HOME`	Root directory of Hive installation	`/usr/local/hive`
`HIVE_CONF_DIR`	Hive configuration directory	`$HIVE_HOME/conf`
`PATH`	System executable search path	Appends bin directories

Important: Ensure your `HADOOP_HOME` variable already exists. If not, add it before the Pig/Hive variables.

3.3 Apply Changes Immediately

```
source ~/.bashrc
```

Context:

- `source` reloads the configuration file without restarting the terminal
- Changes take effect immediately in the current session

Verification:

```
echo $PIG_HOME
echo $HIVE_HOME
```

Both should return their respective installation paths.

Phase 3: Critical Configuration Fixes

Context

This phase addresses compatibility issues between Hadoop 3.3.6, Hive 3.1.3, and WSL2. These patches prevent crashes, memory errors, and job failures.

Step 4: The Guava Library Fix (Hive)

Problem

Hive 3.1.3 bundles Guava 19.0, but Hadoop 3.3.6 requires Guava 27.0. This version mismatch causes:

- ClassNotFoundException crashes
- Runtime library conflicts
- Metastore initialization failures

Solution

```
rm $HIVE_HOME/lib/guava-19.0.jar
cp $HADOOP_HOME/share/hadoop/common/lib/guava-27.0-jre.jar $HIVE_HOME/lib/
```

Context:

- Removes the conflicting older Guava version from Hive
- Copies the correct Guava version from Hadoop's library
- Ensures both systems use the same library version

Verification:

```
ls -la $HIVE_HOME/lib/guava*.jar
```

Should show `guava-27.0-jre.jar` only.

Step 5: Configure hive-site.xml

Problem

Hive needs to know where to store metadata. Without this configuration, it cannot initialize the metastore.

Solution: Create and Configure the File

5.1 Create the Configuration File

```
nano $HIVE_HOME/conf/hive-site.xml
```

5.2 Insert Configuration Content

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:derby:;databaseName=metastore_db;create=true</value>
    </property>
    <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/user/hive/warehouse</value>
    </property>
    <property>
        <name>hive.metastore.schema.verification</name>
        <value>false</value>
    </property>
</configuration>
```

Configuration Explanations:

Property	Value	Purpose
`javax.jdo.option.ConnectionURL`	`jdbc:derby:;databaseName=metastore_db;create=true`	Derby database connection string; creates DB automatically
`hive.metastore.warehouse.dir`	`/user/hive/warehouse`	HDFS directory for storing Hive table data
`hive.metastore.schema.verification`	`false`	Disables strict schema checking (essential for first-time setup)

Context:

- Derby is an embedded SQL database suitable for local Hive installations
- Creates metastore database in your home directory
- Simplifies setup without requiring external database servers

---

Step 6: Configure yarn-site.xml (The WSL2 Fix)

Problem

WSL2 environments have limited memory reporting. Without these fixes:

- "Job Failed" errors appear without root cause
- Virtual memory checks cause unnecessary failures
- YARN cannot locate required libraries (ClassNotFoundException)

Solution: Update YARN Configuration

6.1 Retrieve Your Hadoop Classpath

Run this command and **copy the entire output string**:

```
hadoop classpath
```

Example Output (yours will be longer):

```
/usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/*:/usr/local/hadoop/share/hadoop/common/*:/usr/local/hadoop/sh
```

6.2 Edit yarn-site.xml

```
nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

6.3 Add/Update These Properties

Insert these inside the `<configuration>` tags:

```
<property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
</property>
<property>
    <name>yarn.nodemanager.pmem-check-enabled</name>
    <value>false</value>
</property>
<property>
    <name>yarn.application.classpath</name>
    <value>PASTE_YOUR_LONG_CLASSPATH_STRING_HERE</value>
</property>
```

Property Explanations:

Property	Value	Purpose	WSL2 Impact
`yarn.nodemanager.vmem-check-enabled`	`false`	Disables virtual memory validation	WSL2 reports incorrect vMem; disabling prevents false failures
`yarn.nodemanager.pmem-check-enabled`	`false`	Disables physical memory validation	WSL2 memory constraints differ from Linux; prevents premature job termination
`yarn.application.classpath`	Your hadoop classpath	Tells YARN where all libraries are located	Resolves ClassNotFoundException errors

Critical Step: Replace `PASTE_YOUR_LONG_CLASSPATH_STRING_HERE` with the actual output from Step 6.1. This is essential.

Step 7: Configure mapred-site.xml

Problem

MapReduce workers don't know where to find Hadoop binaries, causing:

- "HADOOP_MAPRED_HOME not found" errors
- Map/Reduce task failures
- Container launch failures

Solution

7.1 Edit Configuration File

```
nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

7.2 Add These Properties

Insert inside the `<configuration>` tags:

```
<property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
```

Property Explanations:

Property	Purpose
`yarn.app.mapreduce.am.env`	Environment for ApplicationMaster process
`mapreduce.map.env`	Environment for Map tasks
`mapreduce.reduce.env`	Environment for Reduce tasks

Context:

- All three specify the same Hadoop path: `/usr/local/hadoop`
- Ensures consistency across all MapReduce processes
- Each process inherits these environment variables at startup

---

Phase 4: Initialization

Context

This phase starts Hadoop services and initializes the Hive metastore database. These steps must be executed in order.

Step 8: Start Hadoop Services

```
start-dfs.sh
start-yarn.sh
```

Context:

- `start-dfs.sh` starts the Distributed File System (NameNode and DataNode)
- `start-yarn.sh` starts the resource manager and node managers
- Hadoop must be running before creating HDFS directories

Verification:

```
jps
```

Expected Output:

```
1234 NameNode
5678 DataNode
9012 ResourceManager
3456 NodeManager
7890 Jps
```

All five processes should appear.

Step 9: Create HDFS Directories

Context

HDFS directories store Hive data and temporary files. These must exist with proper permissions before Hive can operate.

9.1 Create Hive Warehouse Directory

```
hdfs dfs -mkdir -p /user/hive/warehouse
```

Context:

- `-p` creates parent directories if they don't exist
- `/user/hive/warehouse` is the standard location for Hive table data

9.2 Create Temporary Directory

```
hdfs dfs -mkdir -p /tmp
```

Context:

- `/tmp` is used by Hive and Pig for intermediate processing
- Must be writable by all users

9.3 Set Write Permissions on Warehouse

```
hdfs dfs -chmod g+w /user/hive/warehouse
```

Context:

- `g+w` grants write permission to the group
- Allows Hive metastore to write table data

9.4 Set Full Permissions on /tmp

```
hdfs dfs -chmod -R 777 /tmp
```

Context:

- `-R` recursively applies to all subdirectories
- `777` grants read/write/execute to owner, group, and others
- Prevents "Permission Denied" errors during job execution

Verification:

```
hdfs dfs -ls -la /
```

Should show both `/user` and `/tmp` directories.

Step 10: Initialize Hive Metastore

Important Precondition

Run this command from your home directory (not from `/usr/local/hive` or any other location).

```
cd ~
```

10.1 Clear Any Corrupted Previous Database

```
rm -rf metastore_db
```

Context:

- Removes any existing metastore database that might be corrupted
- Derby creates a new clean database on initialization
- Safe to run even if the directory doesn't exist

10.2 Initialize Hive Schema

```
schematool -dbType derby -initSchema
```

Context:

- `schematool` is a Hive utility that manages database schemas
- `-dbType derby` specifies using Derby (embedded database)
- `-initSchema` creates the initial schema for Hive

Expected Output:

```
Initialization script completed
schemaTool completed
```

If you see errors like "Database already exists":

- Run `rm -rf metastore_db` again
- Try `schematool -dbType derby -initSchema` again

Verification:

```
ls -la metastore_db
```

Should show a directory containing Derby database files.

Phase 5: Final Verification (Testing)

Context

This phase runs integrated tests that verify both Pig and Hive are functioning correctly and can access data through HDFS.

Step 11: Create and Upload Test Data

11.1 Create Local Test Data File

```
nano students.txt
```

11.2 Insert Test Data

```
101,John,Engineering
102,Alice,Medical
103,Bob,Engineering
104,Sarah,Arts
```

Context:

- Comma-separated values (CSV) format
- Fields: Student ID, Name, Department
- 4 records for comprehensive testing

11.3 Upload to HDFS

```
hdfs dfs -mkdir -p /input_data
hdfs dfs -put students.txt /input_data/
```

Verification:

```
hdfs dfs -cat /input_data/students.txt
```

Should display the test data contents.

Step 12: Test Apache Pig

12.1 Start Pig Interactive Mode

```
pig
```

Expected Prompt:

```
grunt>
```

12.2 Execute Pig Latin Commands

At the `grunt>` prompt, type these commands:

```
students = LOAD '/input_data/students.txt' USING PigStorage(',') AS (id:int, name:chararray, dept:chararray);
engineers = FILTER students BY dept == 'Engineering';
DUMP engineers;
```

Command Breakdown:

Command	Purpose
`LOAD`	Reads data from HDFS

Command	Purpose
`USING PigStorage(',')`	Specifies comma as field delimiter
`AS (id:int, name:chararray, dept:chararray)`	Defines schema with data types
`FILTER`	Selects rows matching a condition
`DUMP`	Displays results to console

12.3 Verify Output

Expected Output:

```
(101,John,Engineering)
(103,Bob,Engineering)
```

Context:

- Only Engineering department records should appear
- IDs 101 and 103 match the filter condition

12.4 Exit Pig

```
quit
```

Step 13: Test Apache Hive

13.1 Start Hive CLI

Important: Run from your home directory (where `metastore_db` was created):

```
cd ~
hive
```

Expected Prompt:

```
hive>
```

13.2 Create Hive Table

```
CREATE TABLE student_records (id INT, name STRING, dept STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

Command Breakdown:

Clause	Purpose
`CREATE TABLE student_records`	Creates table named student_records
`(id INT, name STRING, dept STRING)`	Defines three columns with data types
`ROW FORMAT DELIMITED`	Specifies row format
`FIELDS TERMINATED BY ','`	Specifies comma as field separator

12.3 Load Data into Table

```
LOAD DATA INPATH '/input_data/students.txt' INTO TABLE student_records;
```

Context:

- `INPATH` specifies the HDFS path to the data file
- Data is moved from `/input_data/` to the warehouse directory
- `INTO TABLE` appends to the table

12.4 Query the Data

```
SELECT * FROM student_records WHERE dept = 'Arts';
```

12.5 Verify Output

Expected Output:

```
104     Sarah   Arts
```

Context:

- Only the Arts department record appears
- Note: Hive outputs tab-separated (not comma-separated like input)
- This verifies successful table creation, data loading, and querying

12.6 Exit Hive

```
quit;
```

Troubleshooting Quick Reference

Issue	Probable Cause	Solution
`pig: command not found`	PIG_HOME not in PATH	Verify `source ~/.bashrc` was run
`hive: command not found`	HIVE_HOME not in PATH	Verify `source ~/.bashrc` was run
`Guava ClassNotFoundException`	Old guava-19.0.jar still in Hive	Re-run Step 4 (Guava fix)
`Job Failed` errors in Pig	YARN memory checks blocking jobs	Verify yarn-site.xml changes from Step 6
`HADOOP_MAPRED_HOME not found`	mapred-site.xml not configured	Re-run Step 7
`metastore_db already exists`	Corrupted Derby database	Run `rm -rf ~/metastore_db` and reinit
`Permission Denied` on /tmp	Incorrect directory permissions	Re-run `hdfs dfs -chmod -R 777 /tmp`
Hive stuck during schema init	Running from wrong directory	Ensure you're in `~` (home directory)

Summary Checklist

After completing all phases, verify:

- [ ] Pig 0.17.0 installed in `/usr/local/pig`
- [ ] Hive 3.1.3 installed in `/usr/local/hive`
- [ ] PIG_HOME and HIVE_HOME in `~/.bashrc`
- [ ] Guava library replaced (guava-27.0-jre.jar in Hive/lib)
- [ ] hive-site.xml created with Derby configuration
- [ ] yarn-site.xml updated with memory checks disabled and classpath set
- [ ] mapred-site.xml updated with HADOOP_MAPRED_HOME paths
- [ ] HDFS directories created: `/user/hive/warehouse`, `/tmp`
- [ ] Hive metastore initialized successfully (schemaTool completed)
- [ ] Pig test executed successfully (2 Engineering records returned)
- [ ] Hive test executed successfully (1 Arts record returned)

Document Archive Information

Configuration Files Modified:

- `~/.bashrc` (Environment variables)
- `$HIVE_HOME/conf/hive-site.xml` (Hive configuration)
- `$HADOOP_HOME/etc/hadoop/yarn-site.xml` (YARN configuration)
- `$HADOOP_HOME/etc/hadoop/mapred-site.xml` (MapReduce configuration)

Directories Created:

- `/usr/local/pig` (Pig installation)
- `/usr/local/hive` (Hive installation)
- `~/metastore_db` (Hive metastore database)
- `/user/hive/warehouse` (HDFS Hive warehouse)
- `/tmp` (HDFS temporary directory)

Versions Used:

- Apache Pig: 0.17.0
- Apache Hive: 3.1.3 (Archive version, maintains MapReduce)

- Apache Hadoop: 3.3.6 (assumed pre-installed)
- Guava Library: 27.0-jre (synchronized with Hadoop)

Keep this document for reference and environment rebuilding.