# HowToSetupYourDevelopmentEnvironment

This page describes how to get your environment setup and is IDE agnostic.

## Requirements

- Java 7 or 8 (Branch 2) or Java 8 (trunk)
- Maven 3.3 or later
- Your favorite IDE
- Protobuf 2.5.0

## Setup Your Development Environment in Linux

The instructions below talk about how to get an environment setup using the command line to build, control source, and test. These instructions are therefore IDE independent. Take a look at EclipseEnvironment for instructions on how to configure Eclipse to build, control source, and test. If you prefer IntelliJ IDEA, then take a look here

- Choose a good place to put your code. You will eventually use your source code to run Hadoop, so choose wisely. For example `~/code/hadoop`.
- Get the source. This is documented in HowToContribute. Put the source in `/code/hadoop (or whatever you chose) so that you have {{{/code/hadoop/hadoop-common`
- cd into `hadoop-common`, or whatever you named the directory
- attempt to run `mvn install`. To build without tests: `mvn install -DskipTests`
- If you get any strange errors (other than JUnit test failures and errors), then consult the *Build Errors* section below.
- follow GettingStartedWithHadoop to learn how to run Hadoop.
- If you run in to any problems, refer to the *Runtime Errors* below, along with the troubleshooting document here: TroubleShooting

## Setup Your Development Environment in OSX

The Linux instructions match, except that:

XCode is needed for the command line compiler and other tools.

protobuf 2.5.0 needs to be built by hand, as macports and homebrew no longer ship that version.

Follow the instructions in the building from source Installing protoc 2.5.x compiler on mac *but change the URL for the protobuf archive to https://github.com/google/protobuf/releases/download/v2.5.0/protobuf-2.5.0.tar.gz*.

To verify that protobuf is correctly installed, the command `protoc --version` must print out the string `libprotoc 2.5.0`.

## Run HDFS in pseudo-distributed mode from the dev tree

Build the packaging from the top level. This will build the distribution in an exploded format that we can run directly (i.e. no need to untar):

```
mvn clean package -Pdist -DskipTests
```

```
export HADOOP_COMMON_HOME=$(pwd)/$(ls -d hadoop-common-project/hadoop-common/target/hadoop-common-*-SNAPSHOT)
export HADOOP_HDFS_HOME=$(pwd)/$(ls -d hadoop-hdfs-project/hadoop-hdfs/target/hadoop-hdfs-*-SNAPSHOT)
export PATH=$HADOOP_COMMON_HOME/bin:$HADOOP_HDFS_HOME/bin:$PATH
```

Set the `fs.default.name` to local HDFS:

```
cat > $HADOOP_COMMON_HOME/etc/hadoop/core-site.xml  << EOF
<?xml version="1.0"?><!-- core-site.xml -->
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost/</value>
  </property>
</configuration>
EOF
```

You can now run HDFS daemons. E.g.:

```
# Format the namenode
hdfs namenode -format
# Start the namenode
hdfs namenode
# Start a datanode
hdfs datanode
```

Note that the `start-dfs.sh` script will not work with this set up, since it assumes that HADOOP_COMMON_HOME and HADOOP_HDFS_HOME are the same directory.

# Run MapReduce in pseudo-distributed mode from the dev tree

Build the packaging from the top level. This will build the distribution in an exploded format that we can run directly (i.e. no need to untar):

```
mvn clean package -Pdist -DskipTests
```

```
export HADOOP_COMMON_HOME=$(pwd)/$(ls -d hadoop-common-project/hadoop-common/target/hadoop-common-*-SNAPSHOT)
export HADOOP_HDFS_HOME=$(pwd)/$(ls -d hadoop-hdfs-project/hadoop-hdfs/target/hadoop-hdfs-*-SNAPSHOT)
export HADOOP_MAPRED_HOME=$(pwd)/$(ls -d hadoop-yarn-project/target/hadoop-yarn-*-SNAPSHOT)
export HADOOP_YARN_HOME=$HADOOP_MAPRED_HOME
export PATH=$HADOOP_COMMON_HOME/bin:$HADOOP_HDFS_HOME/bin:$HADOOP_MAPRED_HOME/bin:$PATH
```

Configure YARN to start a MR shuffle service:

```
cat > $HADOOP_YARN_HOME/conf/yarn-site.xml  << EOF
<?xml version="1.0"?>
<configuration>
    <!-- Site specific YARN configuration properties -->
    <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce.shuffle</value>
    </property>
    <property>
      <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
      <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
</configuration>
EOF
```

Run YARN daemons:

```
yarn resourcemanager
yarn nodemanager
```

Run a MR job:

```
cd hadoop-mapreduce-project
ant examples -Dresolvers=internal
cd ..
export HADOOP_CLASSPATH=$YARN_HOME/modules/*
mkdir in
cp BUILDING.txt in/
hadoop jar hadoop-mapreduce-project/build/hadoop-mapreduce-examples-*.jar wordcount -Dmapreduce.job.user.
name=$USER in out
```

# Build Errors

### OS/X + Java 7 build failure

```
Exception in thread "main" java.lang.AssertionError: Missing tools.jar at: /Library/Java/JavaVirtualMachines/jdk1.7.0_45.jdk/Contents/Home/Classes/classes.jar. Expression: file.exists()
```

This happens because one of the modules used in the Hadoop build expects `classes.jar` to be in a location it no longer is on Oracle Java 7+ on OS/X. See HADOOP-9350

## Runtime Errors

### ERROR dfs.NameNode: java.io.IOException: javax.security.auth.login.LoginException: Login failed: id: cannot find name for group ID

This was a specific error that one developer received. The error has to do with a permission issue when running Hadoop. At runtime, Hadoop SSHes into all nodes, including itself in a single-node setup. This error occurred because some LDAP groups that the developer belonged to were not found at Hadoop runtime. If you get an error like this, then there can be a number of things wrong, but very generally you have a permissions error.

### DataNode process appearing then disappearing on worker niode

When transitioning from a single-node cluster to a multi-node cluster, one of your nodes may appear to be up at first and then will go down immediately. Check the datanode logs of the node that goes down, and look for a connection refused error. If you get this connection refused error, then this means that your slave is having difficulties finding the master. I did two things to solve this problem, and I'm not sure which one, if not both, solved it. First, erase all of your hadoop temporary data and the namenode on all masters and slaves. Reformat the namenode. Second, make sure all of your master and slave hosts in the conf files (slaves, masters, hadoop-site.xml) refer to full host names (ex: host.domain.com instead of host).