How To:

Compile and Run Hadoop Programs using Eclipse

Eclipse a widely used IDEs for developing Java projects. We recommend you use Eclipse in the labs, as it provides a range of features that will help you develop your project; the documentation we are providing assumes that you will use Eclipse, and provides support for this. If you choose to use another IDE, we may have more difficulty supporting you in labs.

Importing a project into Eclipse

The first step with Eclipse is either creating a new project or importing an existing one. For this project, we will import an existing project.

First of all, ensure you are in the workspace you want to do your project work in; eclipse will create a default workspace when first run. We have provided you with existing projects that will be the starting points for your lab exercises. Please consult the exercise specifications for their locations, and copy the projects into your Eclipse workspace.

In Eclipse, go to 'File > Import > General> Existing Projects into Workspace'. Under 'Select root directory', browse to the existing directory. It should automatically select project. Then, click finish. You should now see the project in your Package Explorer. (Ensure you are in the Java perspective - 'Window > Open Perspective > Java | Other > Java').

Setting up Ant for use with Eclipse

We will use Ant to compile and run Hadoop jobs. The document *How To Build and Run MapReduce Projects Using Ant* will take you through the components of an Ant build script as well as how to run the script from your terminal.

To use Ant from Eclipse, go to 'Run > External Tools > External Tools Configuration'. You should now see the External Tools Configurations Window. This window will manage your ant build scripts. Right click on 'Ant Build', and select 'New'.

In your new run configuration, name your 'Ant Build' whatever you want, such as 'BuildWordcount'. Under the tab 'Main', ensure 'Buildfile' refers to the file *build.xml* in the project that you have just imported to your workspace. If this text is not present, you can either type it in, or click 'Browse Workspace' and browse to the *build.xml* file. Note that the value in <property name="hadoop-location" value="/opt/hadoop-3.3.3/bin/" /> must refer to the installed version of Hadoop. In the lab, this is */opt/hadoop-3.3.3/bin/.*

You may also need to define JAVA_HOME in the Environment tab. In the lab machines, the following should work for Java 11, but you can use whatever version of Java you have been using on the lab machines: export JAVA_HOME="/usr/lib/jvm/java-11-openjdk".

Next, under the 'Targets' tab, choose which target you wish to run with your configuration. For this example, check only 'run'. If you have read *How To Build and Run MapReduce Projects Using Ant*, you will know that this target cleans the project, compiles the classes, builds jar file and runs the jar file as a hadoop local job.

You can now apply these settings, and run, if you so choose. You can define as many configurations as you like (one for running, one for just building, etc). Once defined you can run these from the main Eclipse window using the ‘Run External Tools’ button (looks like a green circle with white triangle with a toolbox in the bottom corner). Clicking on the button runs the last configuration, and the black triangle to the right allows you to select any configuration you have defined.

When a hadoop job is run, the results are written to files in an *output* directory that is a subdirectory of that containing the project. Note that this *output* directory may not be automatically visible in Eclipse, and may need a manual refresh.

Other useful tidbits

There are a few other parts of Eclipse that may be useful to you that are briefly discussed below, however, you may want to look these up in more detail yourself.

Configuring your project’s build path

Your project’s build path should already be set up, but just for your information if you want to add more libraries to your build path, just follow this short tutorial.

First of all, right click on the relevant project in your package explorer pane in the main Eclipse Window. Now go to ‘Build Path > Configure Build path’. In this window, the tab that will be of most interest to you is the ‘Libraries’ tab. You should able to see some jar files already in this pane. There are some buttons on the right.

- Add JARs...
  This button is the main one you will be interested in. Any new libraries you wish to add should be placed in the ‘lib’ folder in your project folder. When you click ‘Add JARs...’ you should browse to this JAR file and add it to the build path.
- Add External JARs...
  This button is similar to the last except it will add any JAR file outside of the workspace.
- Add Variable...
  This button adds CLASSPATH variables.
- Add Library...
  This button adds preconfigured libraries to the build path. The AWS SDK may be of use to you later.
- Add Class Folder...
  Adds a folder containing class files in a similar way to how one might add a JAR file to the build path.
- Add External Class Folder...
  As with JAR files, this adds a class folder that is outside the workspace.

Browsing your project’s objects

One of the most useful aspects of Eclipse is being able to browse your classes by objects, methods and variables. For example, open up any class file into the main Eclipse window. Now, holding ‘ctrl’ (or ‘cmd’ if you are on a Mac), mouse over any variable or object and they should underline.

Clicking on a variable will take you to the first place that variable was declared. This can be useful if you are unsure what a variable is or how it was declared. However, more usefully, clicking on an object or method will open up the class that contains that object or method. You can then read exactly how it works to see if that is the method or object you wish to use.

The Eclipse Console / Problems pane

Think of the Eclipse console as the output that would be printed to your terminal. When you run a java program or your ant script in Eclipse the output is printed to your console. It is as simple as that.

The problems pane, again, is similar to the standard output of your terminal window, however, it shows the current issues with your code, be that warnings (yellow) or compile issues (red). Clicking one of those problems will take you straight to the problem in the code. Ensuring there are no red problems should mean that your code will compile.