Lab Exercise 3 – C++ Build using Bazel

(Single Package, Multiple Targets)

While a single target is sufficient for small projects, you may want to split larger projects into multiple targets and packages. This allows for fast incremental builds – that is, Bazel only rebuilds what's changed – and speeds up your builds by building multiple parts of a project at once. This stage of the tutorial adds a target, and the next adds a package.

This is the directory you are working with for Stage 2:

```
├──NobleProg2
│  ├── main
│  │   ├── BUILD
│  │   ├── hello-world.cc
│  │   ├── hello-greet.cc
│  │   └── hello-greet.h
│  └── WORKSPACE
```

Take a look below at the BUILD file in the cpp-tutorial/stage2/main directory:

```
cc_library(
   name = "hello-greet",
   srcs = ["hello-greet.cc"],
   hdrs = ["hello-greet.h"],
)

cc_binary(
   name = "hello-world",
   srcs = ["hello-world.cc"],
   deps = [
      ":hello-greet",
   ],
)
```

With this BUILD file, Bazel first builds the hello-greet library (using Bazel's built-in cc_library rule), then the hello-world binary. The deps attribute in the hello-world target tells Bazel that the hello-greet library is required to build the hello-world binary.

Before you can build this new version of the project, you need to change directories, switching to the cpp-tutorial/stage2 directory by running:

```
cd ../NobleProg2
```

Now you can build the new binary using the following familiar command:

```
bazel build //main:hello-world
```

Once again, Bazel produces something that looks like this:

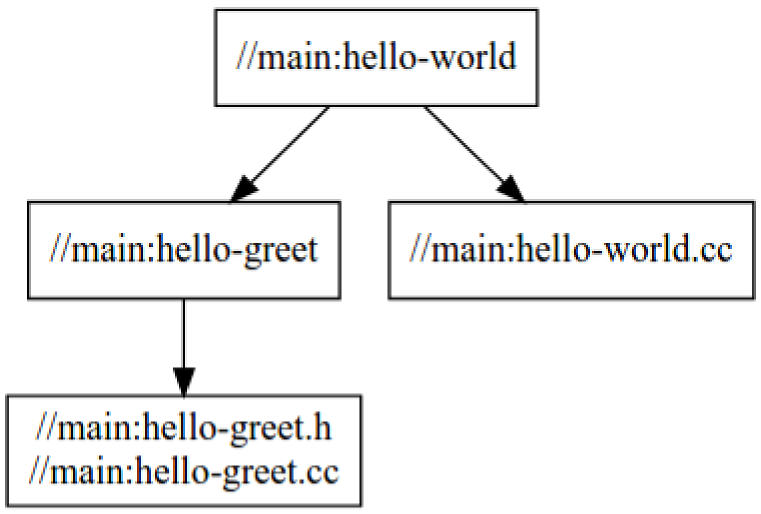
```
INFO: Found 1 target...
Target //main:hello-world up-to-date:
  bazel-bin/main/hello-world
INFO: Elapsed time: 2.399s, Critical Path: 0.30s
```

Now you can test your freshly built binary, which returns another “Hello world”:

```
bazel-bin/main/hello-world
```

If you now modify hello-greet.cc and rebuild the project, Bazel only recompiles that file.

Looking at the dependency graph, you can see that hello-world depends on an extra input named hello-greet:



Summary: NobleProg 2

You've now built the project with two targets. The hello-world target builds one source file and depends on one other target (//main:hello-greet), which builds two additional source files. In the next section, take it a step further and add another package.