

MASS C++ Developer's Guide

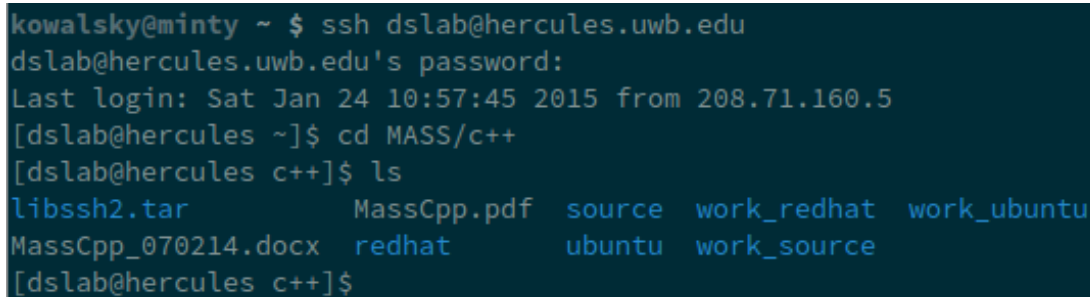
Updated January 25, 2015

1 SETUP

1.1 LOCATION AND PROJECT SETUP

MASS C++ can be found on the Dslab computer and the Linux Lab computers. Currently, the RedHat Linux machine Hercules is being used for development.

The master copy of MASS C++ can be found in the folder `~/MASS/c++`, as shown in figure 1.

A terminal window showing a user named kowalsky@minty connecting via SSH to dslab@hercules.uwb.edu. The user enters the password and is prompted for the last login time (Sat Jan 24 10:57:45 2015 from 208.71.160.5). The user then navigates to the directory ~/MASS/c++ and lists the contents. The output shows several files and directories: libssh2.tar, MassCpp.pdf, source, work_redhat, work_ubuntu, MassCpp_070214.docx, redhat, ubuntu, and work_source.

```
kowalsky@minty ~ $ ssh dslab@hercules.uwb.edu
dslab@hercules.uwb.edu's password:
Last login: Sat Jan 24 10:57:45 2015 from 208.71.160.5
[dslab@hercules ~]$ cd MASS/c++
[dslab@hercules c++]$ ls
libssh2.tar      MassCpp.pdf  source  work_redhat  work_ubuntu
MassCpp_070214.docx  redhat      ubuntu  work_source
[dslab@hercules c++]$
```

Figure 1 Location of MASS C++

The codebase is broken up into several folders. The `work_` folders are currently unused and present only for historical reasons. Development occurs with git branches based on the source directory. The `redhat` and `ubuntu` folders contain Makefiles specific to those systems.

1.2 GETTING A COPY OF THE CODE FOR DEVELOPMENT

To get a local copy of MASS C++, git clone the repository on your machine as shown:

```
git clone ssh://dslab@hercules.uwb.edu:~/git/MASS_CPP.git
```

2 MAKING CHANGES

2.1 MAKING THE CHANGE

If you are unfamiliar with Git, checkout the Git training files for more indepth discussion on how to use in in development. The training files can be found in the `~/Training/GitTraining` folder.

MASS C++ is using the Git Flow model for updates. To add a feature, branch off of the develop branch. You can then push your branch to origin to test it in Jenkins.

Here is the basic workflow in the command line, which should be very close to any GUI Git application's process.

```
git checkout -b <MY_BRANCH_NAME>

git add <CHANGED_FILES>

git commit

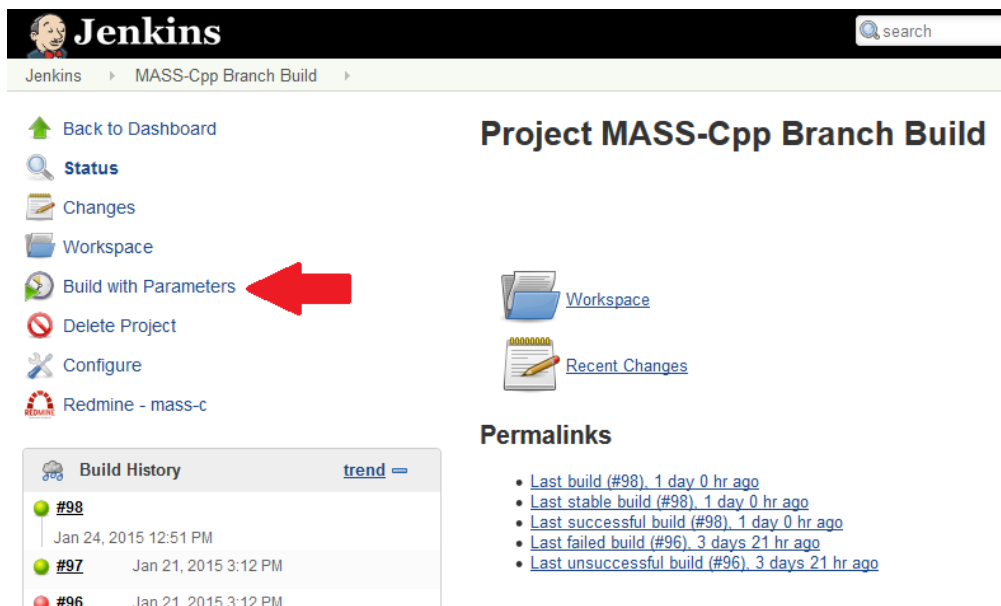
git push origin <MY_BRANCH_NAME>
```

2.2 BUILDING THE CHANGE

There are two ways to build your project.

2.2.1 Method 1: Jenkins

The easiest is to go to [Jenkins MASS-Cpp Branch Build](#). Select Build with Parameters. Then enter the name of your branch.



Build Number	Status	Timestamp
#98	Success	Jan 24, 2015 12:51 PM
#97	Success	Jan 21, 2015 3:12 PM
#96	Failure	Jan 21, 2015 3:12 PM

Figure 2 Building with Parameters

If the build is green, then it compiled successfully. If not, compilation has failed.

2.2.2 Method 2: Linux Lab

To build a particular branch, you will need to clone the repository into a folder so you can checkout the correct branch you need to build.

For the Linux Lab machines, you can do the following:

```
ssh dslab@uw1-320-lab.uwb.edu

mkdir <MY_NAME>

cd <MY_NAME>
```

```
git clone ssh://dslab@hercules.uwb.edu:/~git/MASS_CPP.git
```

You will now be able to checkout your branch. To build it, enter the ubuntu directory and type

```
make
```

This will build the MASS library.

Warning: This only builds your changes. Runtime errors will not be detected. Do not assume that you have not broken anything until you run your branch on one of the Linux machines!

Note: When building MASS C++, use the redhat makefile if building on Hercules, and use the Ubuntu makefile if building on the Linux lab machines. The Ubuntu machines are preferred for testing development.

2.3 TESTING THE CHANGE

To run MASS with a sample program, you will need to have setup a clone of the repository as shown in Method 2. In the ubuntu/samples folder, type

```
sh compile.sh
```

```
sh run.sh
```

This will run MASS with a test program so you can catch obvious runtime errors.

3 RELEASING A NEW VERSION

3.1 TAGGING IN GIT FOR RELEASE

3.2 COPY TO THE RELEASE DIRECTORY