



CASAL2 Contributors Manual

C. Marsh and S. Rasmussen

Contents

1	Introduction	1
2	Creating a local repository	2
2.1	Git Username and Profile	2
2.2	Git Software	2
2.3	Cloning a repository	2
3	Maintaining and contributing to a forked repository	5
3.1	Checking the status of the forked repository	5
3.2	Updating the forked repository to incorporate changes committed to the master . .	5
4	Setting up CASAL2 BuildSystem	9
4.1	Overview	9
4.2	Building on Windows	10
4.2.1	Prerequisite Software	10
4.2.2	Pre-Build Requirements	11
4.2.3	Building CASAL2	11
4.3	Building on Linux	12
4.3.1	Prerequisite Software	12
4.3.2	Building CASAL2	12
4.4	Troubleshooting	13
4.4.1	Third-party Libraries	13
4.4.2	Main Code Base	13
5	CASAL2 build rules	14
5.1	CASAL2 coding practice and style	14
5.2	Unit tests	14
5.3	Reporting (Optional)	15
5.4	Update manual	16
5.5	Builds to pass before merging changes	16
6	An example of adding a new process	17
7	Merging changes form a forked repository to a master repository	18

1. Introduction

The Contributors Manual provides an overview for users who wish to access the source code, compile their own version of CASAL2, or who wish to contribute to CASAL2 source code. CASAL2 is open source and the development team encourages users to submit and contribute changes, bug fixes, or enhancements. This manual is intended to provide a guide for users who wish to access the source code for CASAL2, build and compile CASAL2, or contribute to the development of CASAL2.

The CASAL2 source code is hosted on github, and can be found at <https://github.com/NIWAFisheriesModelling/CASAL2>.

A Microsoft Windows bundle which includes a binary, manual, examples and other help guides can be downloaded at <ftp://ftp.niwa.co.nz/Casal2/windows/Casal2.zip> for the Microsoft Windows version, ftp://ftp.niwa.co.nz/Casal2/windows/Casal2_setup.exe for the Microsoft Windows installer. The Linux bundle which includes a binary, manual, examples and other help guides can be downloaded at <ftp://ftp.niwa.co.nz/Casal2/linux/Casal2.tar.gz>.

If you have any questions or require more information, please contact the CASAL2 development team at casal2@niwa.co.nz.

To maintain the quality and integrity of the code base and ensure that any changes are reliable and accurate, the development team has created this manual to assist contributors in understanding how to access the source code, how to compile and build, and the guidelines for submitting code changes to the CASAL2 Development Team. We want this process to be as pain free as possible to encourage contributions.

At the beginning of each section there will be a list of points that the section will go into at more detail. If you have experience with some aspects you can skip but if you run into trouble you can always come back to check this guide. This manual covers basics such as setting up a github profile, to using github, and all the way to compiling and modifying source code.

2. Creating a local repository

This section will cover the following points

1. Registering a github username
2. Download git software
3. Fork the master repository

2.1. Git Username and Profile

The first step is to create a username and profile on github (if you do not already have one). Creating a username and profile on github is free and easy to do.

Go to <https://www.github.com> to register a username and set up a profile if you do not already have one. See the help at github for more information. Once you have set up a github account you need download the git software that you use to “communicate” with repositories (places where the software source code is stored) on github.

2.2. Git Software

You will need to acquire a git client in order to clone a copy of the source code and link this to the repository.

CASAL2 also requires a command line version of git in order to compile. The CASAL2 build environment requires git in order to evaluate the version of the code used at compile time to include into the executable and manual when being built.

One package that allows this on a Microsoft Windows platform is tortoisegit (see <https://tortoisegit.org/download>) to pull, push and commit changes to git repositories. However there are many other clients that could be used to achieve the same functionality.

2.3. Cloning a repository

The publicly available CASAL2 code is in the master repository. Only the CASAL2 Development Team have permission to add, delete, or change code in the master repository. Other contributors can either add, delete, or change code either by forking the master repository’ or by maintaining a local version of the repository. Forking a copy on github can be done at <https://github.com/NIWAFisheriesModelling/CASAL2> by selecting the fork button in the top right of the page circled in Figure 2.1

This will create a copy of the CASAL2 repository under your profile at the point of the fork. To check that you have successfully forked the repository, go to your git profile and you should see a CASAL2 repository under your repositories, shown in Figure 2.2,

An important point is that the forked repository will not automatically keep up to date with the master repository. So if the master changes, you will want to keep your forked repository up to date. This can easily be done and is explained in the next section.

2 Creating a local repository

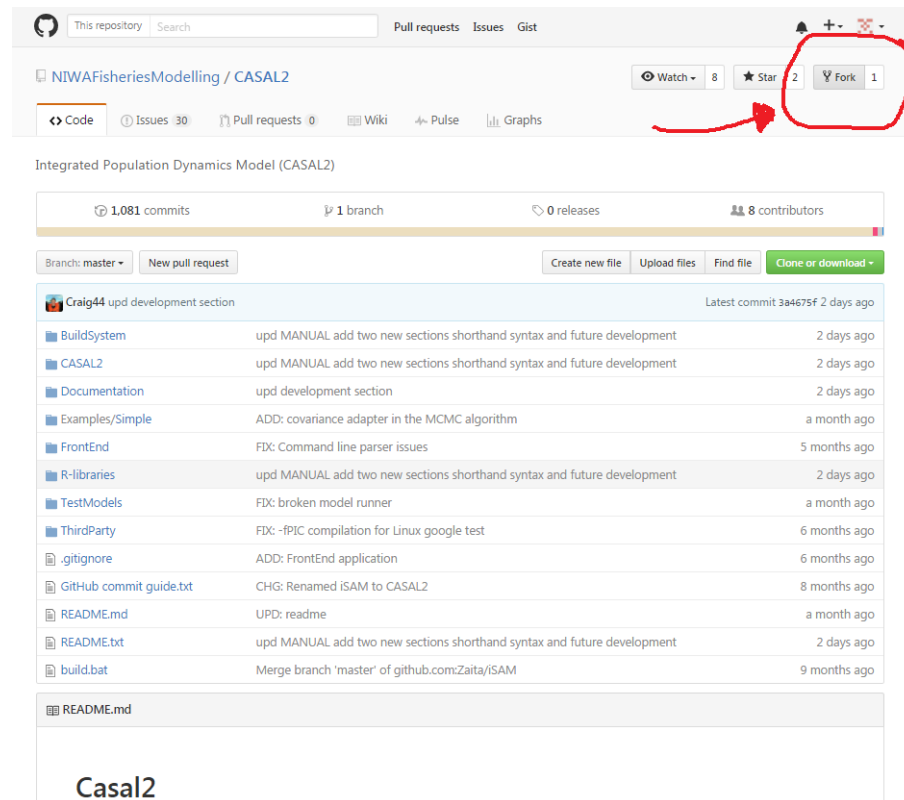


Figure 2.1: Creating a forked repository

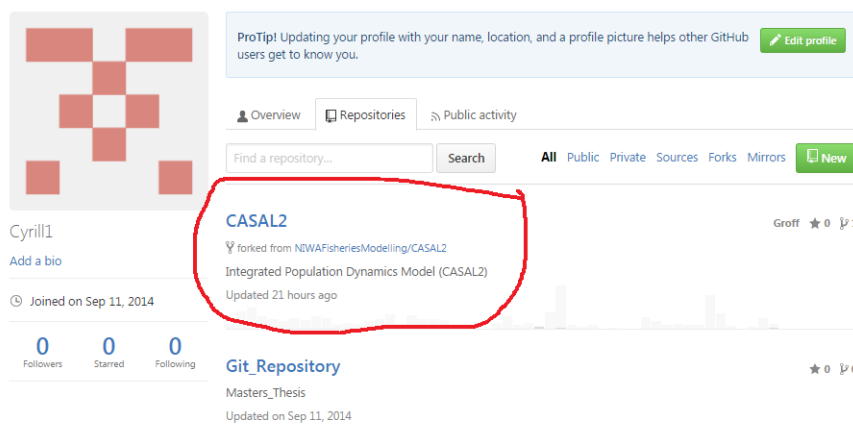


Figure 2.2: Fork success

3. Maintaining and contributing to a forked repository

This section covers the following details

1. Checking the status of the forked repository compared to the master repository
2. Updating forked repository
3. Making changes to your forked repository

3.1. Checking the status of the forked repository

The status of the forked repository as compared to the master repository can be seen on github (see Figure 3.1)

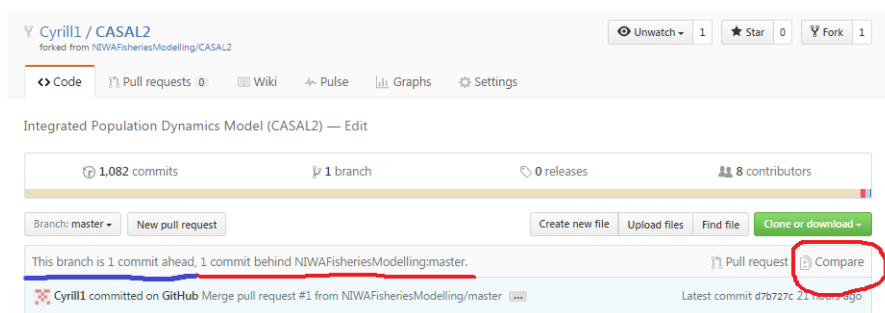


Figure 3.1: Fork status

This line underlined in blue indicates how many commits have occurred on the forked repo since forking the repo or updating the forked repo. The line underlined in red indicates the commits that have occurred to the master since you forked the repo or last updated it. In the above example, we have made one commit to our forked repo (indicated by the blue) and the forked version is one commit behind the master (indicated by the red).

3.2. Updating the forked repository to incorporate changes committed to the master

To update your forked repository to be the same as the master, click the 'compare' button (circled in red on the right of Figure 3.1). This will bring up a page that will tell you of the changes that have occurred to the master repository.

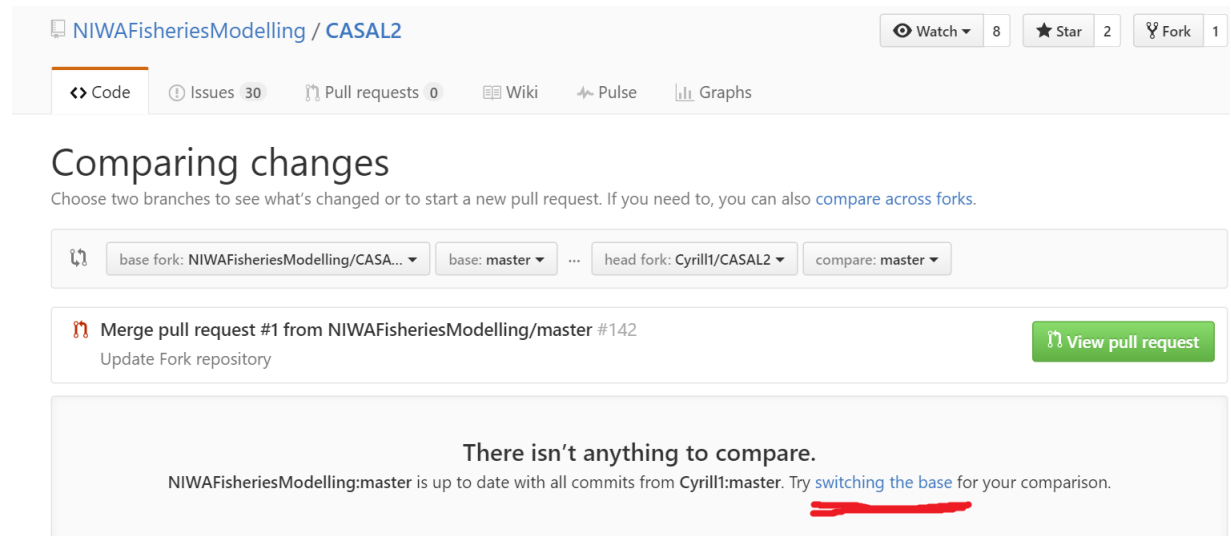


Figure 3.2

If the following figure appears (Figure 3.2) telling you "there isn't anything to compare" click on the switching the base underlined in red.

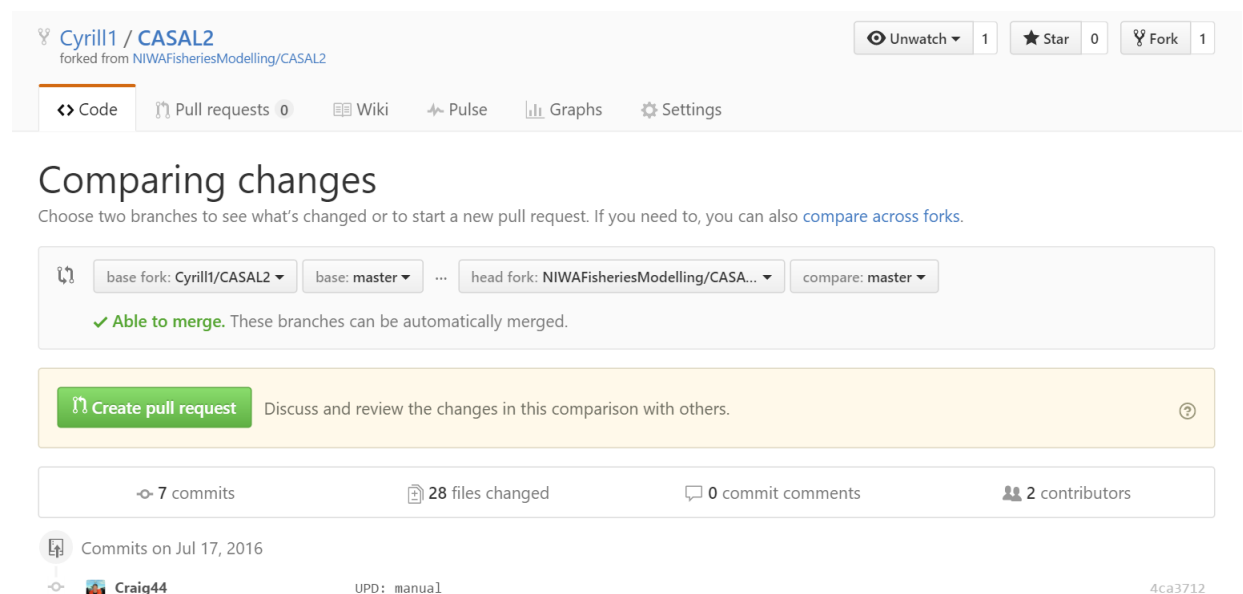


Figure 3.3

You can see that there are 7 commits and 28 files changed on the master that we want to merge into our forked repo. You can see that there is now a "create pull request" and a green message saying "Able to merge". This indicates you will be able to update your forked repo with ease. Click on the "create pull request" button. This will open a git pull request (Figure 3.4) we suggest adding a nice title for the merge like "Updating Craig's Forked Repository"

The screenshot shows the 'Update from original' form in GitHub. At the top, it displays the base fork (CyrilH/CASAL2), the base branch (master), and the head fork (NIWAFisheriesModelling/CASA...). A green checkmark indicates 'Able to merge'. The form has a 'Write' tab and a 'Preview' tab. Below the tabs is a large text area for a comment, with a placeholder 'Leave a comment'. Below the text area is a button 'Create pull request'. To the right of the form, there are sections for 'Labels' (None yet), 'Milestone' (No milestone), and 'Assignees' (No one—assign yourself).

Figure 3.4

Once you write a reasonable title and click create pull request if there are no conflicts, you can merge the changes from the master into your forked repo by clicking "merge pull request" form Figure 3.5.

The screenshot shows the 'Merge pull request' form in GitHub. At the top, it displays a green checkmark and the text 'This branch has no conflicts with the base branch'. Below this is a green button 'Merge pull request'. To the right of the button, it says 'You can also open this in GitHub Desktop or view command line instructions.' Below the button is a large text area for a comment, with a placeholder 'Leave a comment'. Below the text area is a button 'Close pull request' and a green button 'Comment'. To the right of the form, there are sections for 'Milestone' (No milestone), 'Assignees' (No one—assign yourself), '3 participants' (with a list of participants), 'Notifications' (with an 'Unsubscribe' button), and 'Lock conversation'.

Figure 3.5

Once you have clicked that you will be told that the merge was a success as shown in Figure 3.6.

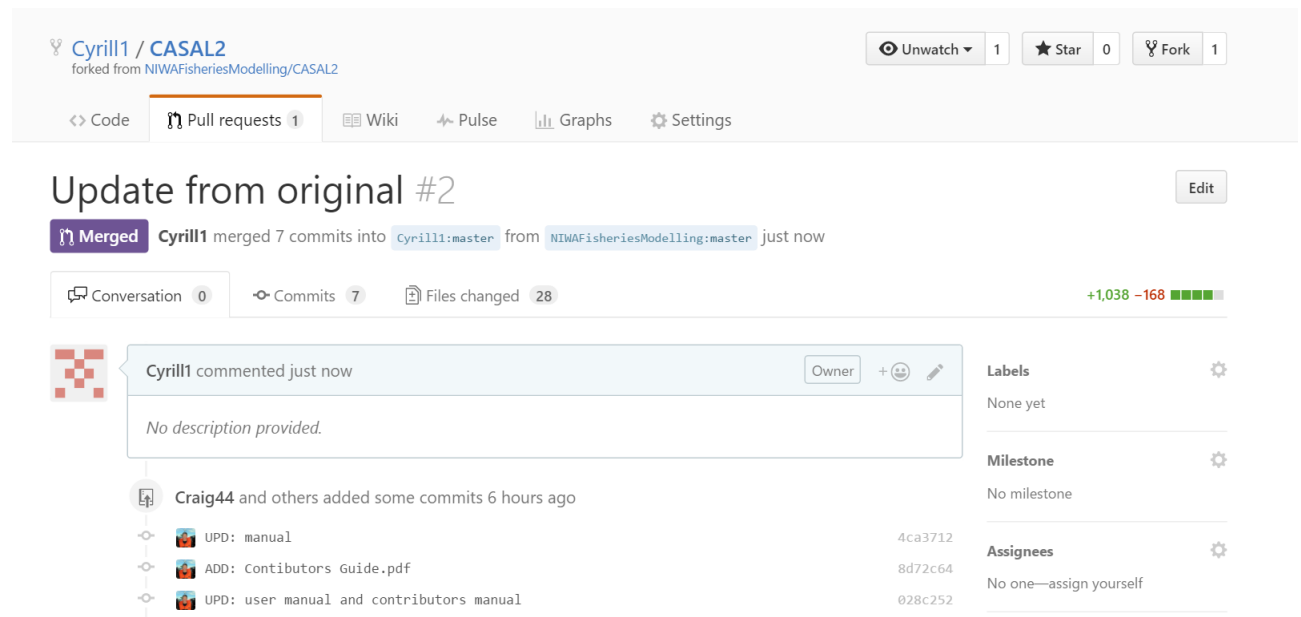


Figure 3.6

Now you have successfully updated your remote forked repository. To incorporate these changes into your local repository on your computer you just need to do a pull and the code changes should be incorporated in your next build.

4. Setting up CASAL2 BuildSystem

This section describes how to set up the environment on your local machine that will allow you to build and compile CASAL2. The build environment can be on either Microsoft Windows or Linux systems. At present the CASAL2 build system supports Microsoft Windows 7+ and Linux (with GCC/G++ 4.9.0+). Apple OSX or other platforms are not currently supported.

4.1. Overview

The build system is made up of a collection of python scripts that do various tasks. These are located in `CASAL2/BuildSystem/buildtools/classes/`. Each python script has its own set of functionality and undertakes one set of actions.

The top level of the build system can be found at `CASAL2/BuildSystem/`. In this directory you can run `doBuild.bat help` from the command line in Microsoft Windows systems or `./doBuild.sh help` from a terminal in Linux systems.

The build system will take one or two parameters depending on what style of build you'd like to achieve. These commands allow the building of various stand-alone binaries, shared libraries, and the documentation. Note that you will need additional software installed on your system in order to build CASAL2. These requirements are described later.

A summary of all of the `doBuild` arguments can be found using the command `doBuild help` in the `BuildSystem` directory.

The current arguments to `doBuild` are:

- `debug`: Build standalone debug executable
- `release`: Build standalone release executable
- `test`: Build standalone unit tests executable
- `documentation`: Build the user manual
- `thirdparty`: Build all required third party libraries
- `thirdpartylean`: Build minimal third party libraries
- `clean`: Remove any previous debug/release build information
- `cleanall`: Remove all previous build information
- `archive`: Build a zipped archive of the application. The application is built using shared libraries so a single `casal2` executable is created.
- `check`: Do a check of the build system
- `modelrunner`: Run the test suite of models
- `installer`: Build an installer package
- `deb`: Create Linux `.deb` installer
- `library`: Build shared library for use by front end application
- `frontend`: Build single CASAL2 executable with all minimisers and unit tests

Valid Build Parameters: (thirdparty only)

- `<library name>`: Target third party library to build or rebuild

Valid Build parameters: (debug/release only)

- `betadiff`: Use BetaDiff auto-differentiation (from CASAL)
- `cppad`: Use CppAD auto-differentiation
- `adolc`: Use ADOLC auto-differentiation in compiled executable

Valid Build parameters: (library only)

- `adolc`: Build ADOLC auto-differentiation library
- `betadiff`: Build BetaDiff auto-differentiation library (from CASAL)
- `cppad`: Build CppAD auto-differentiation library
- `test`: Build Unit Tests library
- `release`: Build release library

The outputs from the build system commands will be placed in subfolders of CASAL2/BuildSystem/bin/<operatingsystem>/<build_type>

For example:

```
CASAL2/BuildSystem/windows/debug
CASAL2/BuildSystem/windows/library_release
CASAL2/BuildSystem/windows/thirdparty/
CASAL2/BuildSystem/linux/library_release
```

4.2. Building on Windows

4.2.1. Prerequisite Software

The building of CASAL2 requires additional build tools and software, including git version control, GCC compiler, LaTeX compiler, and an Windows package builder. CASAL2 requires specific implementations and versions in order to build.

C++ and Fortran Compiler

Source: tdm-gcc (MingW64) from <http://www.tdm-gcc.tdragon.net/>.

CASAL2 is designed to compile under GCC on Microsoft Windows and Linux platforms. While it may be possible to build the package using different compilers, the CASAL2 Development Team does provide any assistance or recommendations. We recommend using 64-bit TDM-GCC version 5.1.0. Ensure you have the gfortran and openmp options installed as a part of TDM-GCC otherwise CASAL2 will not compile. **Note:** A common error that can be made is having a different GCC compiler in your path when attempting to compile. For example, rtools includes a version of the GCC compiler. We recommend removing these from your path prior to compiling.

GIT Version Control

Source: Command line GIT from <https://www.git-scm.com/downloads>.

CASAL2 automatically adds a version number based on the GIT version of the latest commit to its repository. The command line version of GIT is used to generate a version number for the compiled binaries, R libraries, and the manuals.

MiKTeX Latex Processor

Source: Portable version from [urlhttp://www.miktex.org/portable](http://www.miktex.org/portable).

The main user documentation for CASAL2 is a PDF manual generated from LaTeX. The LaTeX syntax sections of the documentation are generated, in part, directly from the code. In order to regenerate the user documentation, you will need the MiKTeX LaTeX compiler.

Inno Setup Installer Builder (optional)

Source: Inno Setup 5 from <http://www.jrsoftware.org/isdl.php>

If you wish to build a Microsoft Windows compatible Installer for CASAL2 then you will need the Inno Setup 5 application installed on the machine. The installation path must be `C:\ProgramFiles(x86)\InnoSetup5\` in order for the build scripts to find and use it.

4.2.2. Pre-Build Requirements

Prior to building CASAL2 you will need to ensure you have both G++ and GIT in your path. You can check both of these by typing:

```
g++ --version
```

```
git --version
```

This also allows you to check that there are no alternative versions of a GCC compiler that may confuse the CASAL2 build.

It's worth checking to ensure GFortran has been installed with the G++ compiler by typing:

```
gfortran --version
```

If you wish to build the documentation bibtex will also need to be in the path:

```
bibtex -version
```

4.2.3. Building CASAL2

The build process is relatively straightforward. You can run `doBuild check` to see if your build environment is ready.

1. Get a copy (clone) of the forked code on your local machine, mentioned in Section 2:
2. Navigate to the BuildSystem folder in `CASAL2/BuildSystem`
3. You need to build the third party libraries with:
 - `doBuild thirdparty`
4. You need to build the binary you want to use:
 - `doBuild release`

5. You can build the documentation if you want:

- `doBuild documentation`

4.3. Building on Linux

This guide has been written against a fresh install of Ubuntu 15.10. With Ubuntu we use `apt-get` to install new packages. You'll need to be familiar with the package manager for your distribution to correctly install the required prerequisite software.

4.3.1. Prerequisite Software

Compiler G++

Ubuntu 15.10 comes with G++ 15.10, gfortran is not installed though so we can install it with: `sudo apt-get install gfortran`.

GIT Version Control

Git isn't installed by default but we can install it with `sudo apt-get install git`

CASAL2 automatically adds a version number based on the GIT version of the latest commit to its repository. The command line version of GIT is used to generate a version number for the compiled binaries, R libraries, and the manuals.

CMake

CMake is required to build multiple third-party libraries and the main code base. You can do this with `sudo apt-get install cmake`

Python2 Modules

There are a couple of Python2 modules that are required to build CASAL2. These can be installed with `sudo apt-get install python-dateutil`

You may also need to install **datetime**, **re** and **distutils**. **Texlive** Latex Processor. No supported latex processors are installed with Ubuntu by default. You can install a suitable latex process with:

```
sudo apt-get install texlive-binaries  sudo apt-get install texlive-latex-base
sudo apt-get install texlive-latex-recommended      sudo apt-get install
texlive-latex-extra
```

Alternatively you can install the complete package: `sudo apt-get install texlive-full`

4.3.2. Building CASAL2

The build process is relatively straightforward. You can run `./doBuild.sh check` to see if your build environment is ready.

1. Get a copy (clone) of the forked code on your local machine, mentioned in Section 2:
2. Navigate to the BuildSystem folder in CASAL2/BuildSystem
3. You need to build the third party libraries with:

- `./doBuild.sh thirdparty`

4. You need to build the binary you want to use:

- `./doBuild.sh release`

5. You can build the documentation if you want:

- `./doBuild.sh documentation`

4.4. Troubleshooting

4.4.1. Third-party Libraries

Its possible there will be build errors or issues building the third-party libraries. If you encounter an error then its worth checking the log files. Each third-party build system stores a log of everything its doing. The files will be named

- `casal2_unzip.log`
- `casal2_configure.log`
- `casal2_make.log`
- `casal2_build.log`
- ...etc.,.

Some of the third-party libraries require very specialised environments for compiling under GCC on Windows. These libraries are packaged with MSYS (MinGW Linux style shell system). The log files for these will be found in `ThirdParty/<libraryname>/msys/1.0/<libraryname>/`

e.g: `ThirdParty/adolc/msys/1.0/adolc/ADOL-C-2.5.2/casal2_make.log`

4.4.2. Main Code Base

If the unmodified code base does not compile, the most likely cause is an issue with the third-party libraries not being built. Ensure they have been built correctly. As they are outside the control of the Development Team, problems can arise that may require the developers of the third party libraries to resolve first. Contact the CASAL2 development team at `casal2@niwa.co.nz` for help.

5. CASAL2 build rules

This section describes the standards and specifications for contributors to have their contributions included within the CASAL2 codebase.

5.1. CASAL2 coding practice and style

CASAL2 is written in C++ and follows the Google C++ style guide (see <https://google.github.io/styleguide/cppguide.html>). The guide is long and comprehensive, so we don't necessarily recommend that you read or understand all of its content. However, the Development Team would like you to follow the Google style of code layout and structure.

This means using good indentations for functions and loops, sensible (human readable) variable names but noting the use of the characters '_' on the end of class variables defined in the .h files.

Annotate your code. For readability we encourage you to put lots of comments in your code. This helps others read what you intended.

On top of annotating your code we encourage developers to add descriptive logging statements (print messages) in the code. You will see in the source code this already. The purpose of this is to allow a descriptive summary of the actions being done in the model for debugging purposes. By using these, it becomes easier to identify issues and errors.

You can also output the text and equations in these logs that would normally be too detailed for general use, as this may allow users to verify the exact equations or processes that have been implemented.

There are different levels of logging in CASAL2 listed below.

- LOG_MEDIUM()
- LOG_FINE()
- LOG_FINEST()
- LOG_TRACE()

To run CASAL2 in log mode piping out any LOG_FINEST and coarser logs (LOG_MEDIUM and LOG_FINE) you can use the following command,

```
casal2 -r --loglevel finest > MyReports.csl2 2> log.out
```

This will output all the logged information to log.out.

5.2. Unit tests

One of the key focusses in the CASAL2 development is an emphasis on software integrity — this is to help ensure that the results from implemented models are consistent and error free. As part of this, we use unit tests to check the individual components of the code base, as well as tests that run entire models in order to validate that the outputs are what we expect them to be.

CASAL2 uses:

- Google testing framework

- Google mocking framework

When adding unit tests, they need to be developed and tested outside of CASAL2 first, for example in **R** or another program like CASAL2 e.g. CASAL or Stock Synthesis. This gives confidence that the test does not contain a calculation or other error.

An example of how to add a unit test for a process is shown in Section 6

5.3. Reporting (Optional)

Currently CASAL2 has reports that are **R** compatible, i.e., all output reports produced by CASAL2 can be read into **R** using the standard **CASAL2 R** package. If you create a new report or modify an old one, you must follow the standard so that the report is **R** compatible.

All reports must start with, `*label (type)` and end with, `*end`

Depending on what type of information you wish to report, will depend on the syntax you need to use. For example

{d} (Dataframe)

Report a dataframe

```
*estimates (estimate_value)
values {d}
process[Recrutiment_BOP].r0 process[Recrutiment_ENLD].r0
2e+006 8e+006
*end
```

{m} (Matrix)

Report a matrix

```
*covar (covariance_matrix)
Covariance_Matrix {m}
2.29729e+010 -742.276 -70160.5
-110126 -424507 -81300
-36283.4 955920 -52736.2
*end
```

{L} (List)

Report a List

```
*weight_one (partition_mean_weight)
year: 1900
ENLD.EN.notag {L}
mean_weights {L}
0.0476604 0.111575 0.199705
end {L}
age_lengths {L}
12.0314 16.2808 20.0135
end {L}
end {L}
*end
```

5.4. Update manual

The syntax sections of the user manual are automatically generated from the source code. This means contributors will need to add or modify the remaining sections of the the user manual to document their changes. This is a requirement of contributed or suggested code changes, and is important for end users to be able to use the new or modified functionality.

5.5. Builds to pass before merging changes

Once you have made changes to the code, you must run the following builds before your changes can be considered for inclusion in the the main code base.

build the unittest version see Section 4 for how to build unittest depending on you're system.

Run the standard and new unit tests to check that they all pass `DoBuild test`

And test that the debug and release of CASAL2 compile and run. `DoBuild debug`

Then run the second phase of unit tests (requires the debug version is built). This runs the tests that comprise of complete model runs `DoBuild modelrunner`

Build the archive `DoBuild archive`

6. An example of adding a new process

The following shows a sequence of figures and text of an example, where we demonstrate how to modify the code base by adding a new process called survivorship.

Example goes here.....

7. Merging changes form a forked repository to a master repository

This section describes how to merge changes from a forked repository to the master repository. This is under the assumptions that the contributor has followed the rules laid out in Section 5. The aim of the forked repository is to maintain code integrity. So we only want contributors who have tested documented there code to be included into the master. This section is the opposite to Section 3 where we are merging changes from a forked repository into the master, which will be incorporated into the next publicly available compiled version of CASAL2. From this example we can see from Figure 7.1 that our forked repository has three commits (underlined in blue) that we want to incorporate into the master repository.

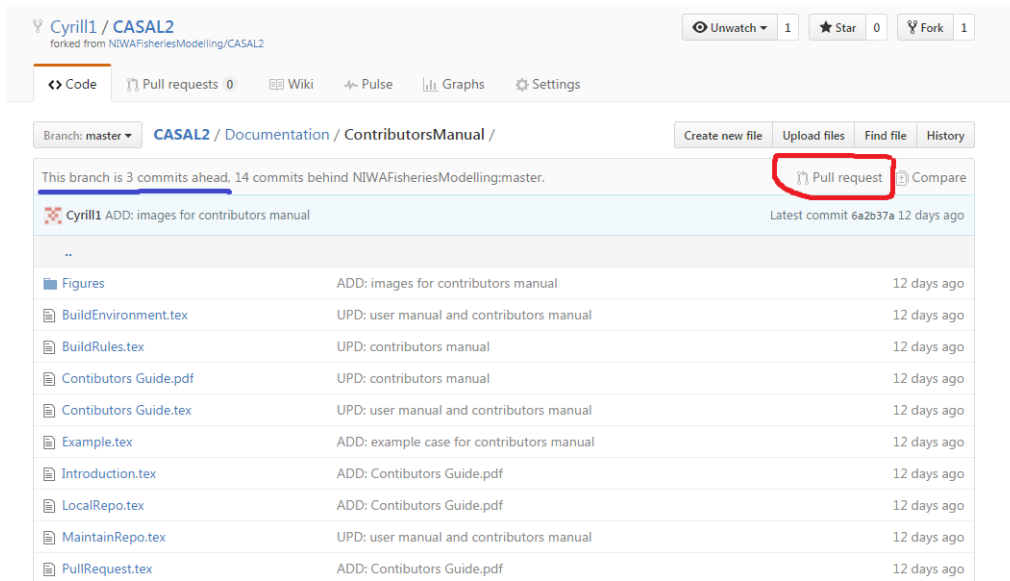


Figure 7.1

To incorporate these changes into the master you need to click on the "pull request" button circled in red. This will prompt a comparison of the changes that you are submitting for inclusion into the master, see Figure 7.2

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base fork: NIWAFisheriesModelling/CASA...
base: master
...
head fork: CyrillJ/CASAL2
compare: master

Able to merge. These branches can be automatically merged.

Create pull request

Discuss and review the changes in this comparison with others.

3 commits

10 files changed

0 commit comments

1 contributor

Commits on Jul 17, 2016

CyrillJ

Merge pull request #2 from NIWAFisheriesModelling/master

5f84c7

CyrillJ

ADD: SurvivalConstantRate process

5985a

CyrillJ

ADD: images for contributors manual

6a2b37

Showing 10 changed files with 234 additions and 0 deletions.

Unified
Split

175
CASAL2/source/Processes/Children/SurvivalConstantRate.cpp

View

```

... @@ -0,0 +1,175 @@
1  /**
2   * @file SurvivalConstantRate.cpp
3   * @author You're Name e.g. Craig Marsh
4   * @institute NIWA
5   * @version 1.0
6   * @date date of creation e.g. 17/07/16
7   * @licence
8   *
9   */
10
11 // Headers
12 #include "SurvivalConstantRate.h"
13
14 #include <numeric>

```

Figure 7.2

We can see that there are 10 files changed over three commits. At the top left corner there is a green button with "create pull request" click that button. This will open a pull request on the master repository as shown in Figure 7.3

Merge pictures and SurvivalConstantRate #149

[Open](#) **Cyrill1** wants to merge 3 commits into `NIMAFisheriesModelling:master` from `Cyrill1:master`

Conversation 0 Commits 3 Files changed 10 +234 -0


Cyrill1 commented just now

No description provided.

Cyrill1 added some commits 12 days ago

- Merge pull request #2 from NIMAFisheriesModelling/master 5fa4c74
- ADD: SurvivalConstantRate process 59a85a2
- ADD: images for contributors manual 6a2b37a

Add more commits by pushing to the `master` branch on **Cyrill1/CASAL2**.

 **This branch has no conflicts with the base branch**
Only those with [write access](#) to this repository can merge pull requests.

Write Preview

AA B i “ < > ☰ ☷ ↶ @

Let us know what you think

Labels
None yet

Milestone
No milestone

Assignees
No one assigned

1 participant

Notifications
Unsubscribe

You're receiving notifications because you authored the thread.

Figure 7.3

This will notify the maintainers of the master repository that a contributor is requesting a merge of the forked code into the master. Maintainers are able to look through the proposed code changed

and have conversations with the contributors if they need clarification. But at this point we are just waiting for the maintainers to accept the changes which then get incorporated into CASAL2 which then makes you a legend.

If you make it this far, we thank you for your development and contributions of this tool and we hope that you get great use out of it =)