Software Engineering

In the Mariokart system

Wim Looman[†]

Department of Electrical Engineering
University of Canterbury
Christchurch
New Zealand

26 September, 2011

[†] wgl18@uclive.ac.nz

Outline

1. Build Toolchain
   - GNU, GNU, GNU
2. Source Control
   - Git
   - Github
3. Continuous Integration
   - What is it?
   - Why would you use it?
   - CI Joe
   - Examples
4. Questions

Build Toolchain

Build Toolchain

- `arm-eabi-gcc` – The GNU Compiler Collection

Build Toolchain

- `arm-eabi-gcc` – The GNU Compiler Collection
- `arm-eabi-gdb` – The GNU Project Debugger

Build Toolchain

- `arm-eabi-gcc` – The GNU Compiler Collection
- `arm-eabi-gdb` – The GNU Project Debugger
- `openocd` – Open On-Chip Debugger

Build Toolchain

- `arm-eabi-gcc` – The GNU Compiler Collection
- `arm-eabi-gdb` – The GNU Project Debugger
- `openocd` – Open On-Chip Debugger
- GNU `make`
  - Lots of targets to enable easy programming, debugging and testing.

Git

Git

- Distributed Version Control System (DVCS)

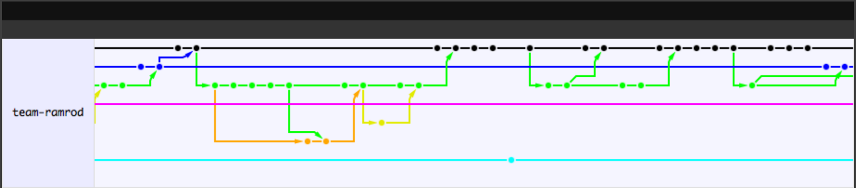
Git

- Distributed Version Control System (DVCS)
- Created by Linus Torvalds for the Linux kernel project.

Git

- Distributed Version Control System (DVCS)
- Created by Linus Torvalds for the Linux kernel project.
- High performance and strong safeguards against corruption.

- Distributed Version Control System (DVCS)
- Created by Linus Torvalds for the Linux kernel project.
- High performance and strong safeguards against corruption.
- Very *"branch-friendly."*

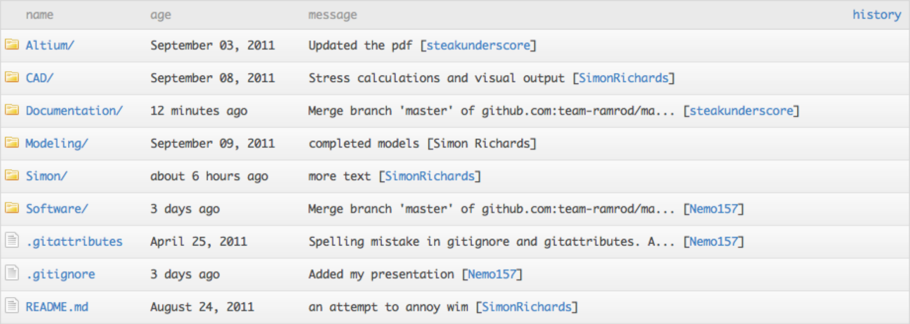


Github

About

"Originally founded by Chris Wanstrath, PJ Hyett and Tom Preston-Werner as a project to simplify sharing code, GitHub has grown into an application used by nearly a million people to store over two million code repositories, making GitHub the largest code host in the world." – `https://github.com/about`

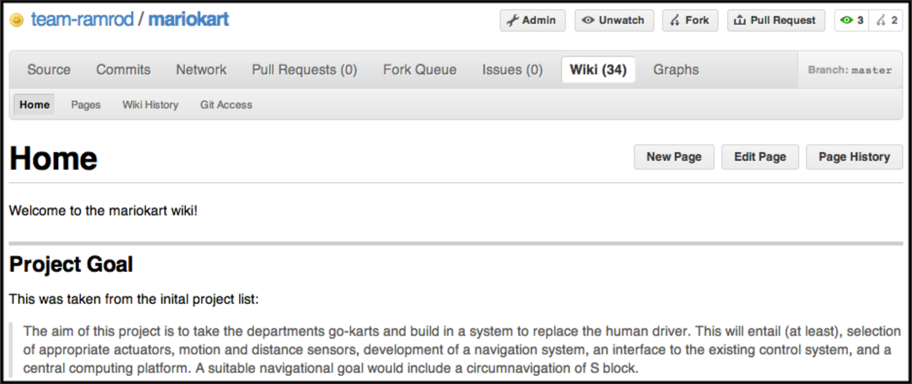
Web Interface[†]



name	age	message	history
Altium/	September 03, 2011	Updated the pdf [steakunderscore]	
CAD/	September 08, 2011	Stress calculations and visual output [SimonRichards]	
Documentation/	12 minutes ago	Merge branch 'master' of github.com:team-ramrod/ma... [steakunderscore]	
Modeling/	September 09, 2011	completed models [Simon Richards]	
Simon/	about 6 hours ago	more text [SimonRichards]	
Software/	3 days ago	Merge branch 'master' of github.com:team-ramrod/ma... [Nemo157]	
.gitattributes	April 25, 2011	Spelling mistake in gitignore and gitattributes. A... [Nemo157]	
.gitignore	3 days ago	Added my presentation [Nemo157]	
README.md	August 24, 2011	an attempt to annoy wim [SimonRichards]	

[†] https://github.com/team-ramrod/mariokart

Wiki[†]



Home

Welcome to the mariokart wiki!

Project Goal

This was taken from the inital project list:

> The aim of this project is to take the departments go-karts and build in a system to replace the human driver. This will entail (at least), selection of appropriate actuators, motion and distance sensors, development of a navigation system, an interface to the existing control system, and a central computing platform. A suitable navigational goal would include a circumnavigation of S block.

[†] https://github.com/team-ramrod/mariokart/wiki

What Is Continuous Integration?

Definition[†]

"Continuous Integration is a software development practice . . . leading to multiple integrations per day . . . verified by an automated build . . . to detect integration errors as quickly as possible." — Martin Fowler

[†] `http://martinfowler.com/articles/continuousIntegration.html`

Why Would You Use Continuous Integration?

Why Would You Use Continuous Integration?

- Detect errors early.

Why Would You Use Continuous Integration?

- Detect errors early.
- Minimizes time between error introduction and fix.

Why Would You Use Continuous Integration?

- Detect errors early.
- Minimizes time between error introduction and fix.
- Provides a stable base for future work.

Why Would You Use Continuous Integration?

- Detect errors early.
- Minimizes time between error introduction and fix.
- Provides a stable base for future work.
  - Very useful for branch-happy development in git.

Why Would You Use Continuous Integration?

- Detect errors early.
- Minimizes time between error introduction and fix.
- Provides a stable base for future work.
  - Very useful for branch-happy development in git.
- Peer pressure.

CI Joe[†]

Repository Description

“CI Joe is a fun Continuous Integration server.”

[†] https://github.com/defunkt/cijoe

CI Joe[†]

- Simple setup.

[†] `https://github.com/defunkt/cijoe`

CI Joe[†]

- Simple setup.
- Designed to work with git.

[†] https://github.com/defunkt/cijoe

CI Joe[†]

- Simple setup.
- Designed to work with git.
- Can trigger a build via a post-hook on github.

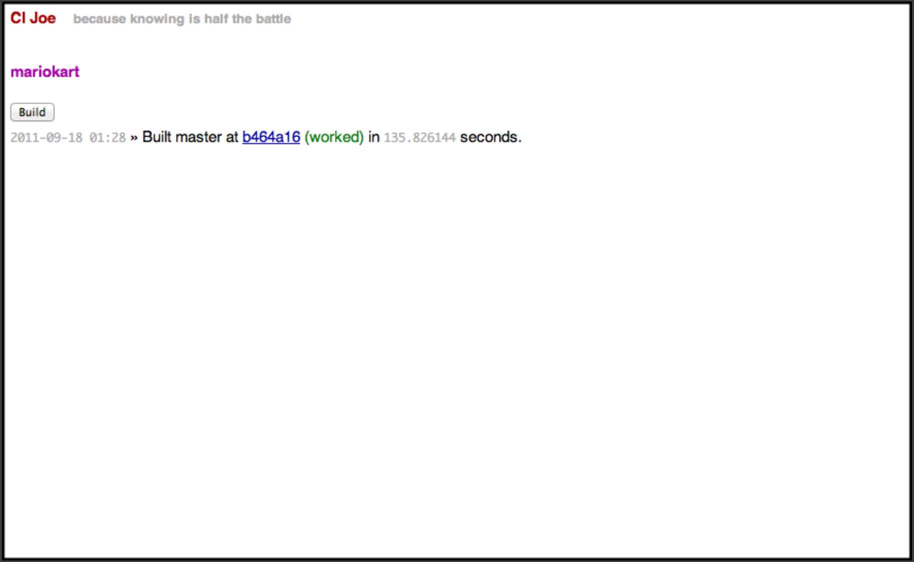
[†] `https://github.com/defunkt/cijoe`

CI Joe[†]

- Simple setup.
- Designed to work with git.
- Can trigger a build via a post-hook on github.
- Reports status via build-hook (used to send email).

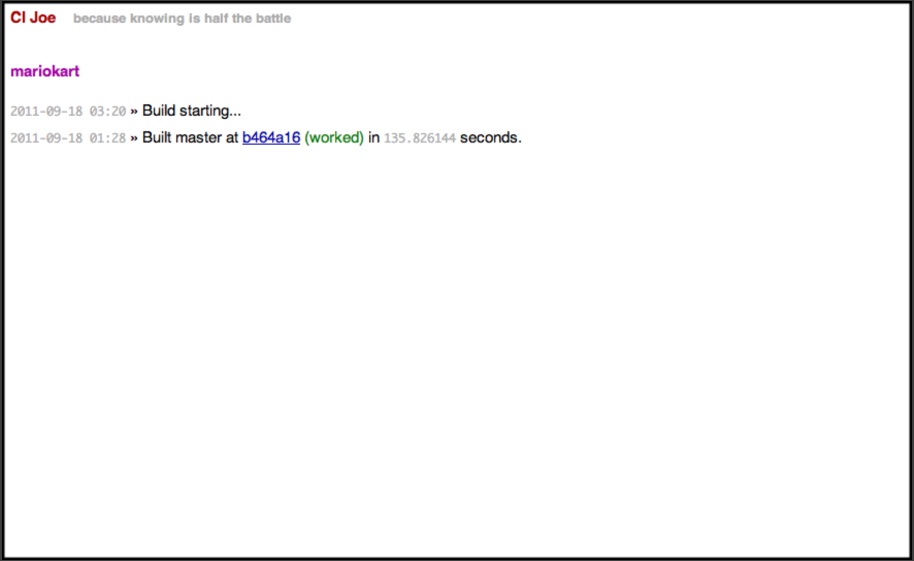
[†] `https://github.com/defunkt/cijoe`

Example 1



2011-09-18 01:28 » Built master at b464a16 (worked) in 135.826144 seconds.

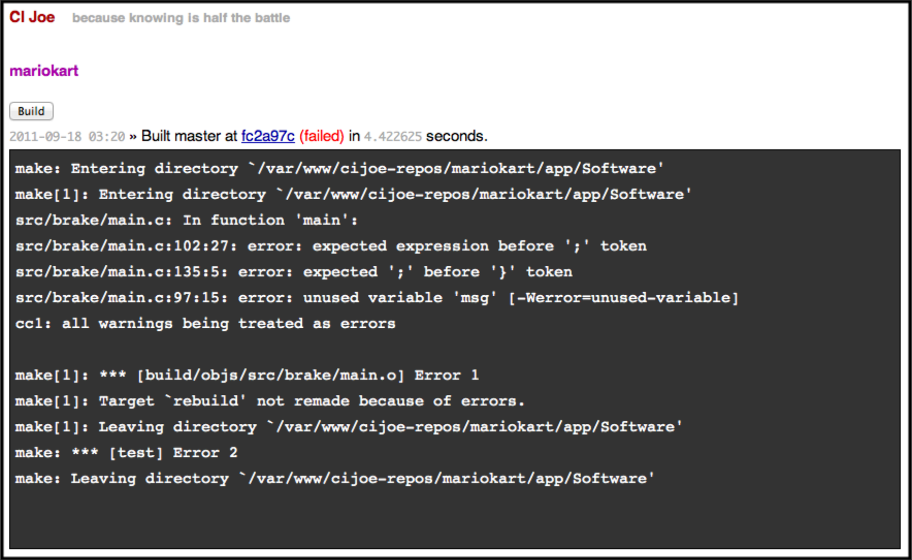
Example 1


CI Joe because knowing is half the battle

mariokart

```
2011-09-18 03:20 » Build starting...
2011-09-18 01:28 » Built master at b464a16 (worked) in 135.826144 seconds.
```

Example 1



CI Joe because knowing is half the battle

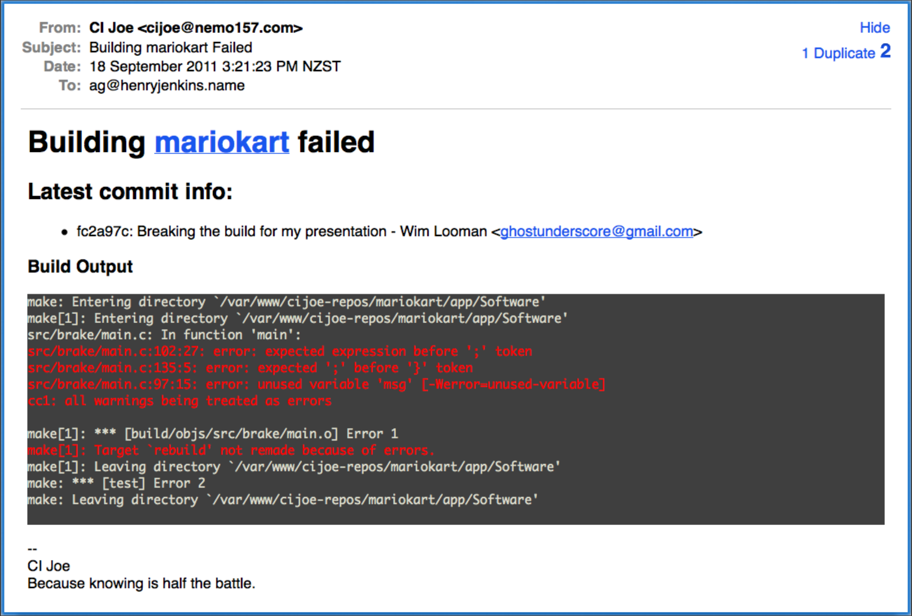
mariokart

2011-09-18 03:20 » Built master at fc2a97c (failed) in 4.422625 seconds.

```
make: Entering directory `/var/www/cijoe-repos/mariokart/app/Software'
make[1]: Entering directory `/var/www/cijoe-repos/mariokart/app/Software'
src/brake/main.c: In function 'main':
src/brake/main.c:102:27: error: expected expression before ';' token
src/brake/main.c:135:5: error: expected ';' before '}' token
src/brake/main.c:97:15: error: unused variable 'msg' [-Werror=unused-variable]
cc1: all warnings being treated as errors

make[1]: *** [build/objs/src/brake/main.o] Error 1
make[1]: Target `rebuild' not remade because of errors.
make[1]: Leaving directory `/var/www/cijoe-repos/mariokart/app/Software'
make: *** [test] Error 2
make: Leaving directory `/var/www/cijoe-repos/mariokart/app/Software'
```

Example 2



From: CI Joe <cijoe@nemo157.com>
Subject: Building mariokart Failed
Date: 18 September 2011 3:21:23 PM NZST
To: ag@henryjenkins.name

Building mariokart failed

Latest commit info:

- fc2a97c: Breaking the build for my presentation - Wim Looman <ghostunderscore@gmail.com>

Build Output

```
make: Entering directory `/var/www/cijoe-repos/mariokart/app/Software'
make[1]: Entering directory `/var/www/cijoe-repos/mariokart/app/Software'
src/brake/main.c: In function 'main':
src/brake/main.c:102:27: error: expected expression before ';' token
src/brake/main.c:135:5: error: expected ';' before '}' token
src/brake/main.c:97:15: error: unused variable 'msg' [-Werror=unused-variable]
cc1: all warnings being treated as errors

make[1]: *** [build/objs/src/brake/main.o] Error 1
make[1]: Target `rebuild' not remade because of errors.
make[1]: Leaving directory `/var/www/cijoe-repos/mariokart/app/Software'
make: *** [test] Error 2
make: Leaving directory `/var/www/cijoe-repos/mariokart/app/Software'
```

--
CI Joe
Because knowing is half the battle.

Git vs SVN

Git

- Distributed (central repository can be used).
- Every clone is a full repository.
- Continual branching and merging is expected.

SVN

- Needs a single central repository.
- Can't commit without access to central repository.
- Branching and merging is painful.

Unit Testing

- Initially was planned.
- Would require mocking and simulating the entire SAM7XC architecture.
  - Hundreds to thousands of registers.
  - Need constraints on which registers get written in which orders.
- Would be awesome, but probably too large for even its own final year project.