GIT

FOR

BEGINNERS

Vince McIntyre | 2019-03-21

WHAT

Revision control

Unlike previous systems you may have used

Used for purposes beyond pure revision control

ORIGIN

Linus Torvalds (2005)

Drew on ideas from other systems, particularly monotone and Darcs

ORIGIN

Junio Hamano

Maintainer since 2005

WHY

- Linux kernel - gigantic merging task
  - Linux 4.15:
    - 1,707 developers
    - 14,226 changesets
    - +725,000 - 407,000 lines
- Tried BitKeeper, but:
  - proprietary tool
  - key features in paid version only

WHY 'GIT'?

"I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git."

WHY 'GIT'?

slightly more plausible:

random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.

HOWTO

FIVE BASICS

- setting up your own repository
- adding files
- changing files
- reviewing changes
- undoing a change

SETTING UP

```
$ ls
hello.c
hello.h
README

$ git init
Initialized empty Git repository in /Users/mci156/demo/.git/
```

SETTING UP

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README
    hello.c
    hello.h

nothing added to commit but untracked files present (use "git add
```

ADDING

```
$ git add hello.c
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   hello.c

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README
    hello.h
```

ADDING

```
$ git add -A
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   README
    new file:   hello.c
    new file:   hello.h
```

ADDING - A STAGED APPROACH

```
$ git commit
```

ADDING - A STAGED APPROACH

```
# Please enter the commit message for your changes. Lines startin
# with '#' will be ignored, and an empty message aborts the commi
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   README
#   new file:   hello.c
#   new file:   hello.h
#
```

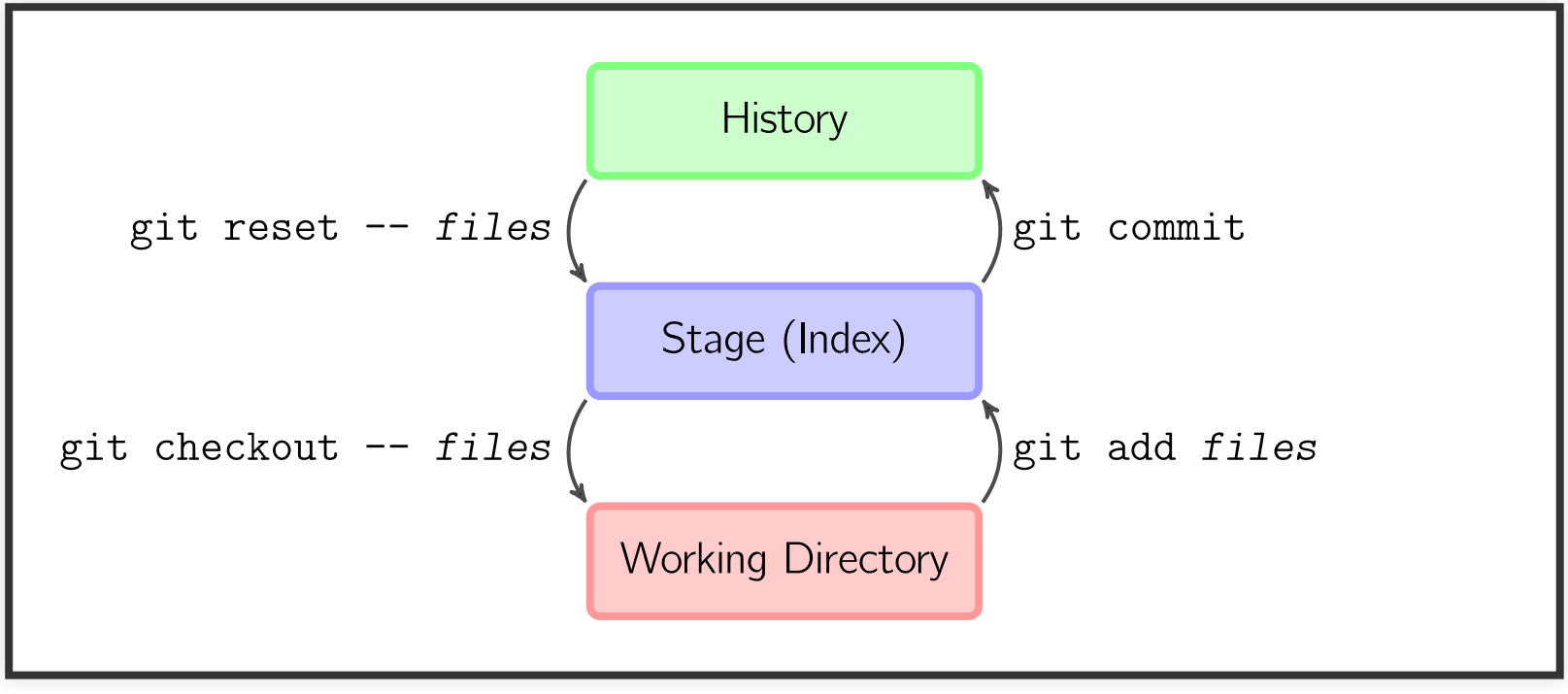
ADDING - A STAGED APPROACH

```
Initial commit of project files
# Please enter the commit message for your changes. Lines startin
# with '#' will be ignored, and an empty message aborts the commi
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   README
#   new file:   hello.c
#   new file:   hello.h
#
```

ADDING - A STAGED APPROACH

```
$ git commit
[master (root-commit) 064d1ae] Initial commit of project files
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README
 create mode 100644 hello.c
 create mode 100644 hello.h
```

ADDING - A STAGED APPROACH



https://marklodato.github.io/visual-git-guide/

HELP WITH ADDING (OR ANYTHING)

```
$ git help add
```

HELP WITH ADDING (OR ANYTHING)

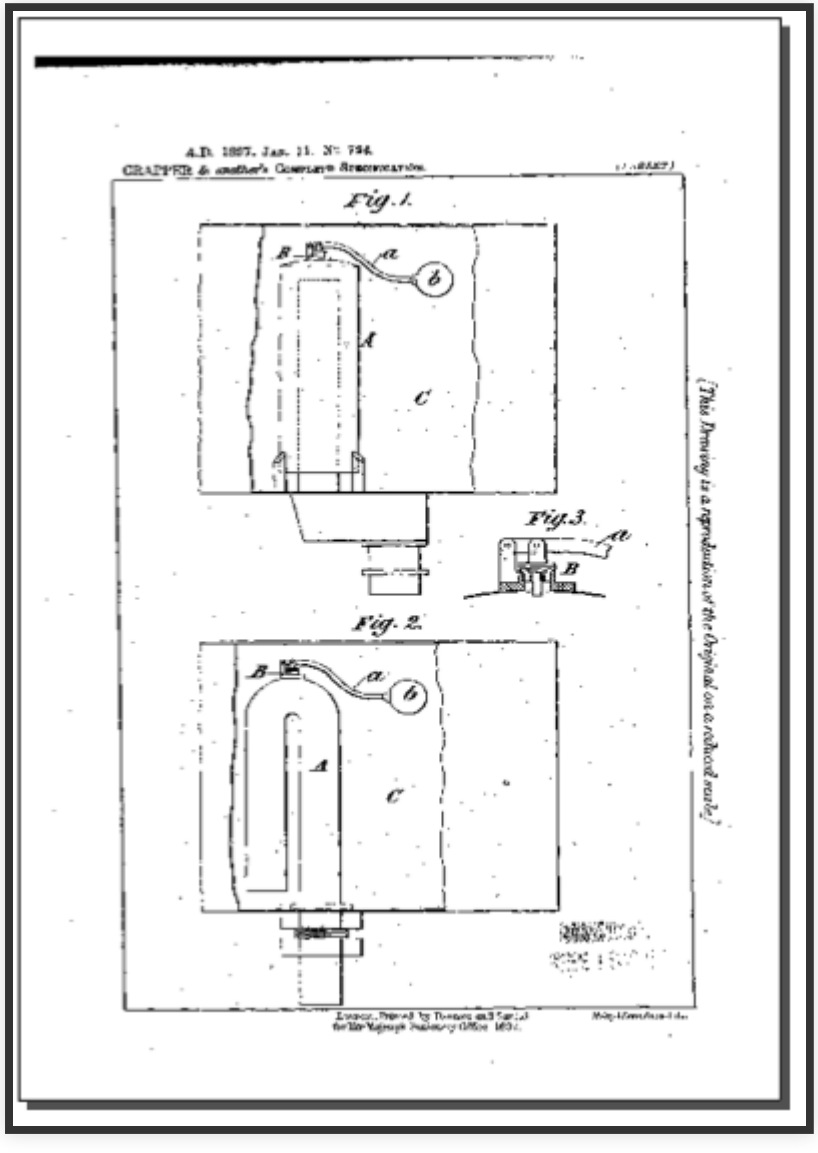
```
GIT-ADD(1)                                Git Manual

NAME
       git-add - Add file contents to the index

SYNOPSIS
       git add [--verbose | -v] [--dry-run | -n] [--force | -f] [
                 [--edit | -e] [--[no-]all | --[no-]ignore-remova
                 [--intent-to-add | -N] [--refresh] [--ignore-err
                 [--chmod=(+|-)x] [--] [<pathspec>...]

DESCRIPTION
       This command updates the index using the current content f
       working tree, to prepare the content staged for the next c
       typically adds the current content of existing paths as a
       with some options it can also be used to add content with
```

ASIDE

ASIDE - 'PORCELAIN'

- There are multiple ways to interact with git
- There's a lot going under the hood (plumbing)
  - plumbing is for toolkit writers
- generic term - 'porcelain'
  - porcelain is for interactive use

WE'RE STILL IN 1897


A.D. 1897, Jan. 11. N° 724.

CRAPPER & another's Complete Specification.

Fig. 1.

Fig. 3.

Fig. 2.

[This Drawing is a reproduction of the Original on a reduced scale.]

Inventors:
CRAPPER, GEORGE;
WHARAM, ROBERT MARR

Patent Application:
GBD189700724 18970111

CHANGING

CHANGING

```
$ ls
README          hello.c         hello.h
$ vi README
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working

    modified:   README

no changes added to commit (use "git add" and/or "git commit -a")
```

CHANGING

```
$ git diff
diff --git a/README b/README
index e69de29..74e96cc 100644
--- a/README
+++ b/README
@@ -0,0 +1 @@
+This program prints 'hello world' in various ways.
```

CHANGING

```
$ git add README
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README
$ git commit -m"Start filling out README"
[master 65ab25a] Start filling out README
 1 file changed, 1 insertion(+)
```

REVIEWING

```
$ git log

commit 65ab25a2cf8092f1c9fc64eb8940d8c3b18f504b (HEAD -> master)
Author: Vincent McIntyre <vincent.mcintyre@gmail.com>
Date:   Tue Dec 4 10:58:08 2018 +1100

    Start filling out README

commit 064d1ae3b5b466e910c735696994ba95daa5e008
Author: Vincent McIntyre <vincent.mcintyre@gmail.com>
Date:   Tue Dec 4 10:15:31 2018 +1100

    Initial commit of project files
```

REVIEWING

```
$ git log hello.c
commit 064d1ae3b5b466e910c735696994ba95daa5e008
Author: Vincent McIntyre <vincent.mcintyre@gmail.com>
Date:   Tue Dec 4 10:15:31 2018 +1100

    Initial commit of project files
```

REVIEWING

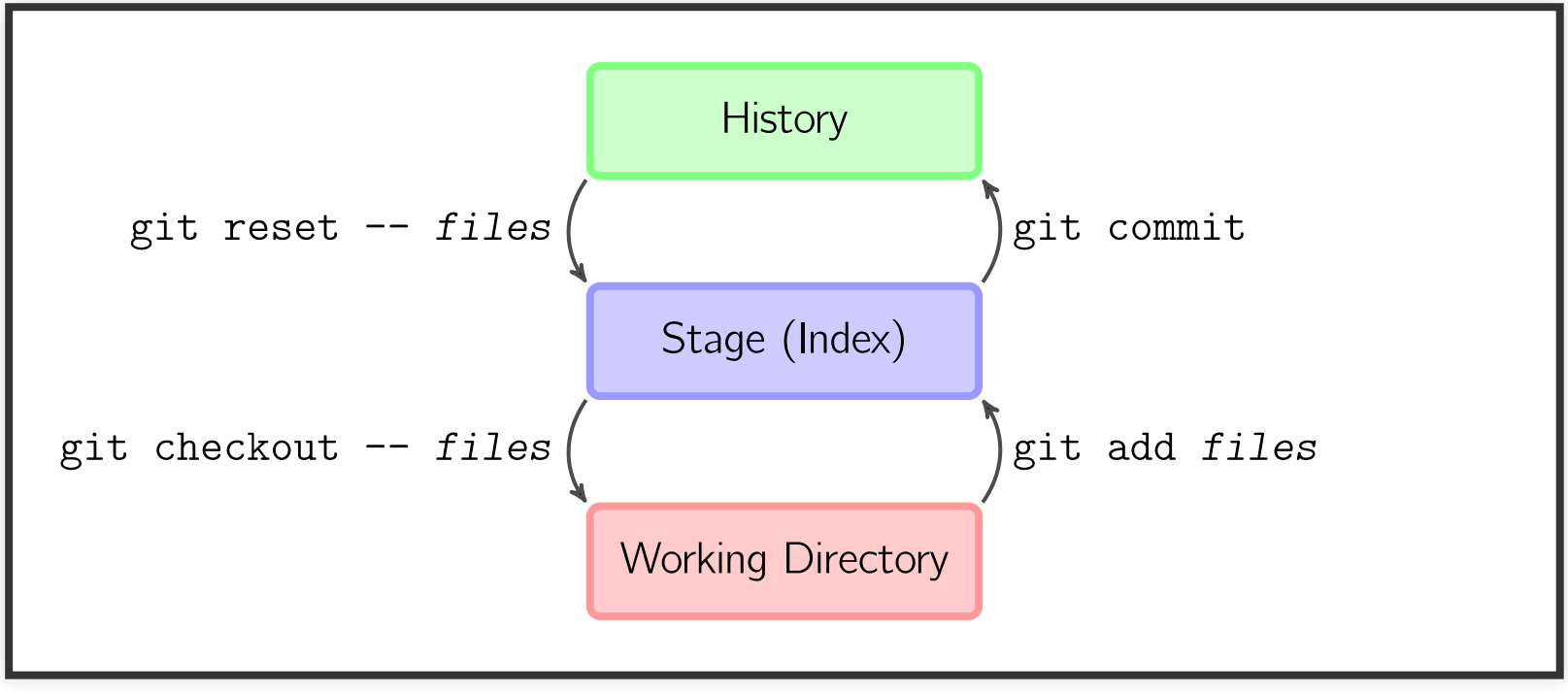
```
$ git log --oneline
65ab25a (HEAD -> master) Start filling out README
064d1ae Initial commit of project files
```

UNDOING

Three opportunities

- before staging
- before committing
- after committing

UNDOING



https://marklodato.github.io/visual-git-guide/

UNDOING - BEFORE STAGING

```
$ vi hello.c
$ git diff
diff --git a/hello.c b/hello.c
index e69de29..3d0ce61 100644
--- a/hello.h
+++ b/hello.h
@@ -0,0 +1 @@
+#include 'hello.h';

$ git checkout hello.c
$ git diff
$
```

UNDOING - BEFORE COMMITTING

```
$ vi hello.c
$ git diff
diff --git a/hello.c b/hello.c
index e69de29..3d0ce61 100644
--- a/hello.c
+++ b/hello.c
@@ -0,0 +1 @@
+#include 'hello.h';

$ git add hello.c
$ git diff
$
```

UNDOING - BEFORE COMMITTING

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.c

$ git diff HEAD
diff --git a/hello.c b/hello.c
index e69de29..3d0ce61 100644
--- a/hello.c
+++ b/hello.c
@@ -0,0 +1 @@
+#include 'hello.h';

$ git diff --staged
```

UNDOING - BEFORE COMMITTING

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.c

$ git reset HEAD hello.c
Unstaged changes after reset:
M    hello.c

$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
```

UNDOING - BEFORE COMMITTING

```
$ git diff
diff --git a/hello.c b/hello.c
index e69de29..f96df4b 100644
--- a/hello.c
+++ b/hello.c
@@ -0,0 +1 @@
+#include 'hello.h'

$ git diff HEAD
diff --git a/hello.c b/hello.c
index e69de29..3d0ce61 100644
--- a/hello.c
+++ b/hello.c
@@ -0,0 +1 @@
+#include 'hello.h'
```

UNDOING - AFTER COMMITTING

```
$ vi hello.c
$ git add hello.c
$ git commit -m"Add include stanza"
[master 5b4fb9d] Add include stanza
 1 file changed, 1 insertion(+)
$
```

UNDOING - AFTER COMMITTING

```
$ vi hello.c
$ git add hello.c
$ git commit -m"Add include stanza"
[master 5b4fb9d] Add include stanza
 1 file changed, 1 insertion(+)
$ git revert 5b4fb9d
```

UNDOING - AFTER COMMITTING

```
Revert "Add include stanza"

This reverts commit 5b4fb9d99ce9172c80b88cea4a6cbe9efa724d84.

# Please enter the commit message for your changes. Lines startin
# with '#' will be ignored, and an empty message aborts the commi
#
# On branch master
# Changes to be committed:
#   modified:   hello.h
#
```

UNDOING - AFTER COMMITTING

```
$ git revert 5b4fb9d
[master 91d44d9] Revert "Add include stanza"
 1 file changed, 1 deletion(-)
$ cat hello.c
$
$ git log --oneline
91d44d9 (HEAD -> master) Revert "Add include stanza"
5b4fb9d Add include stanza
65ab25a Start filling out README
064d1ae Initial commit of project files
```

FIVE NEXT STEPS

- partial changes
- commit log messages
- copying a repository
- working with others
- branching & merging

PARTIAL ADD

```
$ cat README
This program prints 'hello world' in various ways.

License
-------
This program is copyright CSIRO 2018.
It is licensed under the GNU Public License verison 3 or later.

Future plans
------------
 * unicode
 * translations
   * chinese
   * urdu
   * klingon
 * internationalisation
```

PARTIAL ADD

```
$ vi README
diff --git a/README b/README
index b17929c..72d435d 100644
--- a/README
+++ b/README
@@ -4,6 +4,8 @@ License
 -------
-This program is copyright CSIRO 2018 and licensed under the GNU
+This program is copyright CSIRO 2018.
+It is licensed under the GNU Public License verison 3 or later.

 Future plans
 ------------
+ * unicode
   * translations
@@ -13,2 +15,@@ Future plans
```

PARTIAL ADD

```
$ git add -p
```

PARTIAL ADD

```
$ git add -p
diff --git a/README b/README
index b17929c..72d435d 100644
--- a/README
+++ b/README
@@ -2,13 +2,14 @@ This program prints 'hello world' in various wa

 License
 -------
-This program is copyright CSIRO 2018 and licensed under the GNU
+This program is copyright CSIRO 2018.
+It is licensed under the GNU Public License verison 3 or later.

 Future plans
 ------------
+ * unicode
```

PARTIAL ADD

```
$ git add -p
diff --git a/README b/README
index b17929c..72d435d 100644
--- a/README
+++ b/README
@@ -2,13 +2,14 @@ This program prints 'hello world' in various wa

 License
 -------
-This program is copyright CSIRO 2018 and licensed under the GNU
+This program is copyright CSIRO 2018.
+It is licensed under the GNU Public License verison 3 or later.

 Future plans
 ------------
+ * unicode
```

PARTIAL ADD

```
Split into 3 hunks.
@@ -2,7 +2,8 @@

 License
 -------
-This program is copyright CSIRO 2018 and licensed under the GNU
+This program is copyright CSIRO 2018.
+It is licensed under the GNU Public License verison 3 or later.

 Future plans
 ------------
Stage this hunk [y,n,q,a,d,j,J,g,/,e,?]?
```

PARTIAL ADD

```
Split into 3 hunks.
@@ -2,7 +2,8 @@

 License
 -------
-This program is copyright CSIRO 2018 and licensed under the GNU
+This program is copyright CSIRO 2018.
+It is licensed under the GNU Public License verison 3 or later.

 Future plans
 ------------
Stage this hunk [y,n,q,a,d,j,J,g,/,e,?]? y
```

PARTIAL ADD

```
@@ -6,8 +7,9 @@

 Future plans
 ------------
+ * unicode
   * translations
     * chinese
     * urdu
     * klingon
   * internationalisation
Stage this hunk [y,n,q,a,d,K,j,J,g,/,e,?]?
```

PARTIAL ADD

```
@@ -6,8 +7,9 @@

 Future plans
 ------------
+ * unicode
   * translations
     * chinese
     * urdu
     * klingon
   * internationalisation
Stage this hunk [y,n,q,a,d,K,j,J,g,/,e,?]? n
```

PARTIAL ADD

```
@@ -9,6 +11,5 @@
  * translations
    * chinese
    * urdu
    * klingon
  * internationalisation
- * unicode
Stage this hunk [y,n,q,a,d,K,g,/,e,?]?
```

PARTIAL ADD

```
@@ -9,6 +11,5 @@
  * translations
    * chinese
    * urdu
    * klingon
  * internationalisation
- * unicode
Stage this hunk [y,n,q,a,d,K,g,/,e,?]? n
```

PARTIAL ADD

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working

    modified:   README
```

PARTIAL ADD

some things left unstaged

```
$ git diff
diff --git a/README b/README
index 4d1a6e0..72d435d 100644
--- a/README
+++ b/README
@@ -9,2 +9,3 @@ Future plans
 ------------
+ * unicode
   * translations
@@ -14,2 +15 @@ Future plans
   * internationalisation
- * unicode
```

PARTIAL ADD

some things ready to commit

```
$ git diff --staged
diff --git a/README b/README
index b17929c..4d1a6e0 100644
--- a/README
+++ b/README
@@ -4,3 +4,4 @@ License
 -------
-This program is copyright CSIRO 2018 and licensed under the GNU
+This program is copyright CSIRO 2018.
+It is licensed under the GNU Public License verison 3 or later.
```

PARTIAL ADD

fix up typo

```
$ vi README
$ git diff
diff --git a/README b/README
index 4d1a6e0..3e7e7ae 100644
--- a/README
+++ b/README
@@ -5,6 +5,7 @@ License
 This program is copyright CSIRO 2018.
-It is licensed under the GNU Public License verison 3 or later.
+It is licensed under the GNU Public License version 3 or later.

 Future plans
 ------------
+ * unicode
  * translations
@@ -14,2 +15,@@ Future plans
```

PARTIAL ADD

fix up typo

```
$ git add -p
diff --git a/README b/README
index 4d1a6e0..3e7e7ae 100644
--- a/README
+++ b/README
@@ -3,13 +3,13 @@ This program prints 'hello world' in various wa
 License
 -------
 This program is copyright CSIRO 2018.
-It is licensed under the GNU Public License verison 3 or later.
+It is licensed under the GNU Public License version 3 or later.

 Future plans
 ------------
+ * unicode
  * translations
```

PARTIAL ADD

fix up typo

```
Stage this hunk [y,n,q,a,d,s,e,?]? s
Split into 3 hunks.
@@ -3,7 +3,7 @@
 License
 -------
 This program is copyright CSIRO 2018.
-It is licensed under the GNU Public License verison 3 or later.
+It is licensed under the GNU Public License version 3 or later.

 Future plans
 ------------
Stage this hunk [y,n,q,a,d,j,J,g,/,e,?]? y
@@ -7,8 +7,9 @@

 Future plans
 ------------
```

PARTIAL ADD

typo fixed

```
$ git diff --staged
diff --git a/README b/README
index b17929c..92e2887 100644
--- a/README
+++ b/README
@@ -4,3 +4,4 @@ License
 -------
-This program is copyright CSIRO 2018 and licensed under the GNU
+This program is copyright CSIRO 2018.
+It is licensed under the GNU Public License version 3 or later.
```

PARTIAL ADD

```
$ git commit -m"Break long line"
[master b2b692f] Break long line
 1 file changed, 2 insertions(+), 1 deletion(-)
$ git diff --staged
$
```

PARTIAL ADD

```
$ git diff
diff --git a/README b/README
index 92e2887..3e7e7ae 100644
--- a/README
+++ b/README
@@ -9,2 +9,3 @@ Future plans
 ------------
+ * unicode
   * translations
@@ -14,2 +15 @@ Future plans
   * internationalisation
- * unicode
```

PARTIAL ADD

```
$ git add README

$ git diff --staged
diff --git a/README b/README
index 92e2887..3e7e7ae 100644
--- a/README
+++ b/README
@@ -9,2 +9,3 @@ Future plans
 ------------
+ * unicode
   * translations
@@ -14,2 +15 @@ Future plans
   * internationalisation
- * unicode

$ git commit -m"Reorder dot points"
```

REVIEW

```
$ git log --oneline
f4fd0cf (HEAD -> master) Reorder dot points
b2b692f Break long line
1c48cdd remove whitespace
02145d9 Start filling out readme
91d44d9 Revert "Add include stanza"
5b4fb9d Add include stanza
b8e62da Revert "Start filling out README"
65ab25a Start filling out README
064d1ae Initial commit of project files
```

COMMIT LOG MESSAGES

```
$ git log -1
commit f4fd0cfc61f2f2208de02ede80478a2656539af0 (HEAD -> master)
Author: Vincent McIntyre <vincent.mcintyre@gmail.com>
Date:   Wed Dec 5 12:28:22 2018 +1100

    Reorder dot points
```

Where's the 'why'?

COMMIT LOG MESSAGES

https://chris.beams.io/posts/git-commit/

```
$ git log --oneline -5 --author cbeams --before "Fri Mar 26 2009"

e5f4b49 Re-adding ConfigurationPostProcessorTests after its brief
        r814. @Ignore-ing the testCglibClassesAreLoadedJustInTime
        Enhancement() method as it turns out this was one of the
        the recent build breakage. The classloader hacking causes
        downstream effects, breaking unrelated tests. The test me
        useful, but should only be run on a manual basis to ensur
        not prematurely classloaded, and should not be run as par
        automated build.
2db0f12 fixed two build-breaking issues: + reverted ClassMetadata
        to revision 794 + eliminated ConfigurationPostProcessorTe
        further investigation determines why it causes downstream
        (such as the seemingly unrelated ClassPathXmlApplicationC
147709f Tweaks to package-info.java files
22b25e0 Consolidated Util and MutableAnnotationUtils classes into
```

COMMIT LOG MESSAGES

```
$ git log --oneline -5 --author pwebb --before "Sat Aug 30 2014"

5ba3db6 Fix failing CompositePropertySourceTests
84564a0 Rework @PropertySource early parsing logic
e142fd1 Add tests for ImportSelector meta-data
887815f Update docbook dependency and generate epub
ac8326d Polish mockito usage
```

- similar length & form, consise

COMMIT LOG MESSAGES

Re-establishing the context of a piece of code is wasteful. We can't avoid it completely, so our efforts should go to reducing it [as much] as possible. Commit messages can do exactly that and as a result, a commit message shows whether a developer is a good collaborator.

Peter Hutterer

COMMIT LOG MESSAGES

Canonical commit message style

- Separate subject from body with a blank line
- Limit the subject line to 50 characters
- Capitalize the subject line
- Do not end the subject line with a period
- Use the imperative mood in the subject line
- Wrap the body at 72 characters
- Use the body to explain what and why vs. how

COMMIT LOG MESSAGES

```
Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary.
Wrap it to about 72 characters or so. In some contexts,
the first line is treated as the subject of the commit
and the rest of the text as the body. The blank line
separating the summary from the body is critical (unless
you omit the body entirely); various git tools like
'log', 'shortlog` and 'rebase' can get confused if
you run the two together.

Explain the problem that this commit is solving. Focus on
why you are making this change as opposed to how (the code
explains that).
Are there side effects or other unintuitive consequences of
this change? Here's the place to explain them
```

COPYING OUR REPOSITORY

```
$ cd
$ git clone /Users/mci156/demo democlone
Cloning into 'democlone'...
done.
$ cd democlone
$ git remote -v
origin    /Users/mci156/demo (fetch)
origin    /Users/mci156/demo (push)
```

COPYING ANOTHER REPOSITORY

```
$ cd
$ git clone https://github.com/githubtraining/hellogitworld
Cloning into 'hellogitworld'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 306 (delta 0), reused 0 (delta 0), pack-reused 305
Receiving objects: 100% (306/306), 95.29 KiB | 447.00 KiB/s, done
Resolving deltas: 100% (71/71), done.
$
$ cd hellogitworld
$ git remote -v
origin    https://github.com/githubtraining/hellogitworld (fetch)
origin    https://github.com/githubtraining/hellogitworld (push)
% git log -5 --oneline
ef7bebf (HEAD -> master, origin/master, origin/HEAD) Fix groupId
ebbbf77 Update package name, directory
```

WORKING WITH OTHERS

There's a crucial step - going 'bare'

```
$ cd
$ git clone --bare demo demo.git
```

WORKING WITH OTHERS

```
$ cd
$ git clone demo.git democlone
Cloning into 'democlone'...
done.
$ cd democlone
$ git remote -v
origin    /Users/mci156/demo.git (fetch)
origin    /Users/mci156/demo.git (push)
$ git log -3 --oneline
bf4fd0cf (HEAD -> master, origin/master, origin/HEAD) Reorder dot
b2b692f Break long line
1c48cdd remove whitespace
```

WORKING WITH OTHERS

```
$ vi README
$
$ git diff
diff --git a/README b/README
index 3e7e7ae..76c525c 100644
--- a/README
+++ b/README
@@ -10,6 +10,7 @@ Future plans
  * unicode
  * translations
    * chinese
+   * dutch
    * urdu
    * klingon
  * internationalisation
$ git commit README -m"We'll need a dutch translation someday"
```

WORKING WITH OTHERS

(The exciting bit)

```
$  git push -v
Pushing to /Users/mci156/demo.git
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 341 bytes | 341.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To /Users/mci156/demo.git
   f4fd0cf..b272768  master -> master
updating local tracking ref 'refs/remotes/origin/master'
```

BRANCHING AND MERGING

- git makes branching and merging easy
- allows exploration of ideas
- allows rapid mergning into the 'main line'

BRANCHING AND MERGING

```
$ git branch -v
* master b272768 We'll need a dutch translation someday
$ git checkout -b translate
Switched to a new branch 'translate'
$ git branch -v
  master    b272768 We'll need a dutch translation someday
* translate 22da8b6 We'll need a dutch translation someday
$ mkdir i18n
$ touch i18n/strings_en.json
$ git add i18n/strings_en.json
$ git commit -m"Add internationalisation subdirectory"
[translate 22da8b6] Add internationalisation subdirectory
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 i18n/strings_en.json
$
$ git log -5 --oneline
```

BRANCHING AND MERGING

```
$ git diff master
diff --git a/i18n/strings_en.json b/i18n/strings_en.json
new file mode 100644
index 0000000..e69de29
$
$ ls
README        hello.c        hello.h        i18n/
$
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
$ ls
README        hello.c        hello.h
$
$ git log -5 --oneline
b272768 (HEAD -> master, origin/master, origin/HEAD) We'll need a
```

BRANCHING AND MERGING

```
$ git branch -v
* master    b272768 We'll need a dutch translation someday
  translate 22da8b6 Add internationalisation subdirectory
$ git merge -v translate
Updating b272768..22da8b6
Fast-forward
 i18n/strings_en.json | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 i18n/strings_en.json
$
$ ls
README        hello.c        hello.h        i18n/
$ git branch -v
* master    22da8b6 [ahead 1] Add internationalisation subdirecto
  translate 22da8b6 Add internationalisation subdirectory1
$
```

BRANCHING AND MERGING

```
$ git push -v
Pushing to /Users/mci156/demo.git
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 332 bytes | 332.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To /Users/mci156/demo.git
   b272768..22da8b6  master -> master
updating local tracking ref 'refs/remotes/origin/master'
$
$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

SUMMARY

- Git is worth your time
- There's a lot more functionality
  - git grep; git blame; git rebase
- There are different 'procelains' available
  - magit - emacs integration
  - gitk - I use this daily
  - GitX, Versions, MS Visual Studio

ACKNOWLEDGEMENTS

- Image of Linus Torvalds from Wikipedia https://commons.wikimedia.org/wiki/File:LinuxCon_Eu
- Image of Junio Hamano from usesthis.com https://usesthis.com/interviews/junio.c.hamano/

COPYRIGHT

Copyright 2019 CSIRO

License: cc-by-sa

https://creativecommons.org/licenses/by-sa/4.0/