Best Practices

Joaquim Rocha | joaquim.rocha@cern.ch

IT-DSS-TD

January 24, 2014

About yours truly

- Contributor to/Author of many Open Source projects
- Member of the GNOME Foundation
- Former member of Igalia and Red Hat
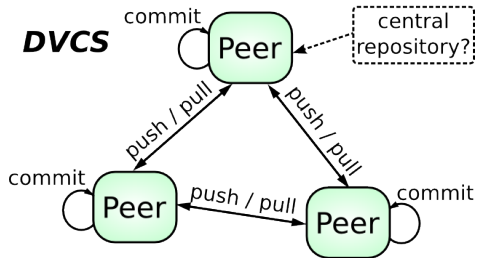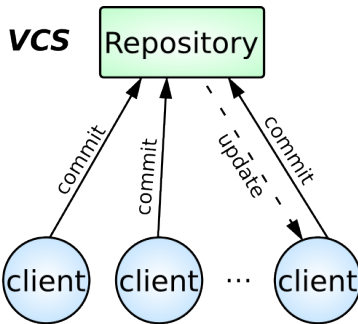- Currently at the Data Storage Services group at CERN
- http://www.joaquimrocha.com

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

Linus Torvalds

What's git

- **Distributed** Version Control System (*DVCS*)

- Initially written in C by Linus Torvalds

- Replacement for *BitKeeper* in Linux kernel development

- Widely used nowadays, both for new and old projects

VCS

Repository

commit

commit

update

commit

client   client   ⋯   client

DVCS

commit

Peer

central repository?

push / pull   push / pull

commit

Peer   push / pull   Peer
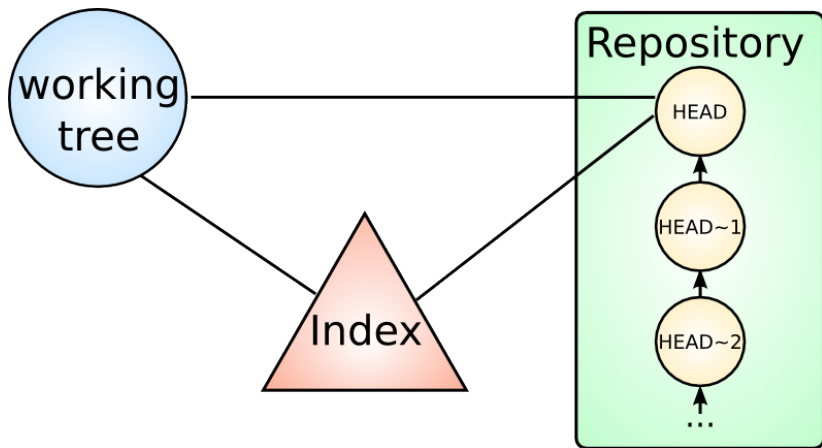
commit

It's the sandbox
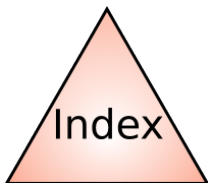
Index

Holds the new changes to be committed
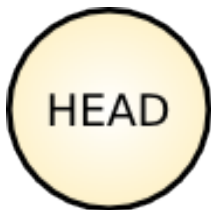
Points to the current commit

working tree

Repository

HEAD

git add

*Add changes
to the Index
(prepare commit)*

Index

HEAD~1

HEAD~2

...
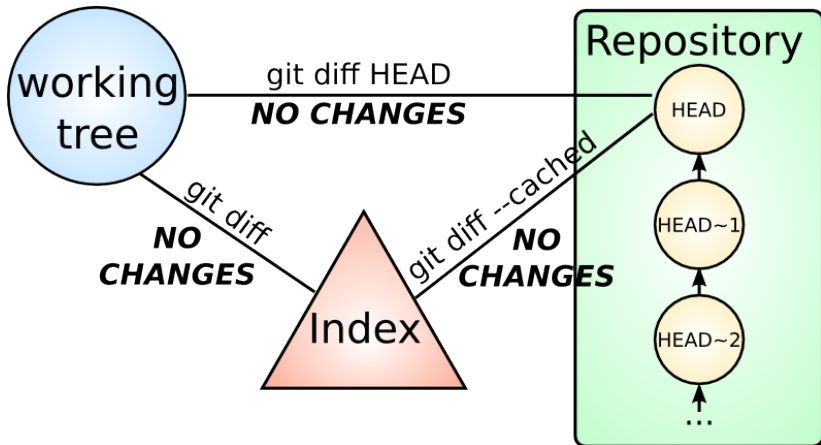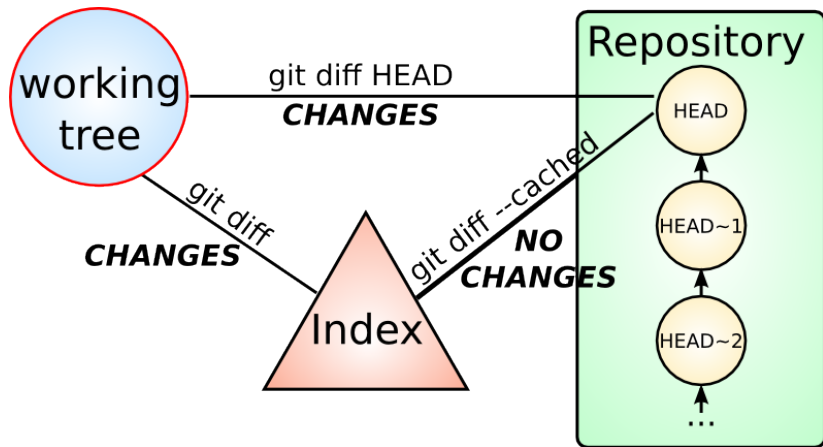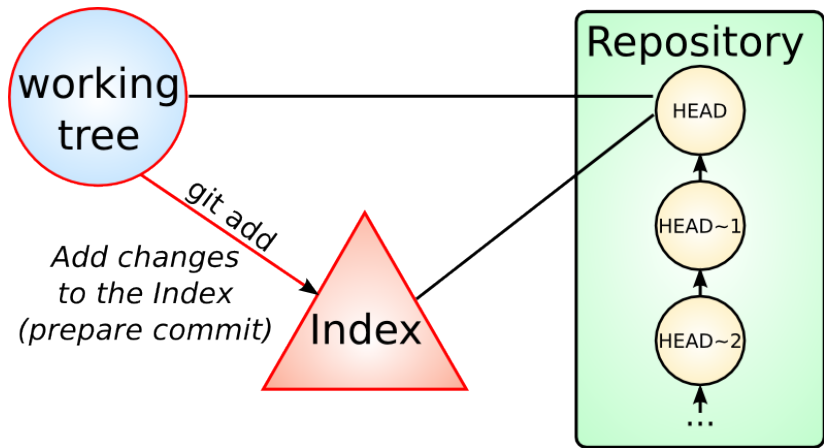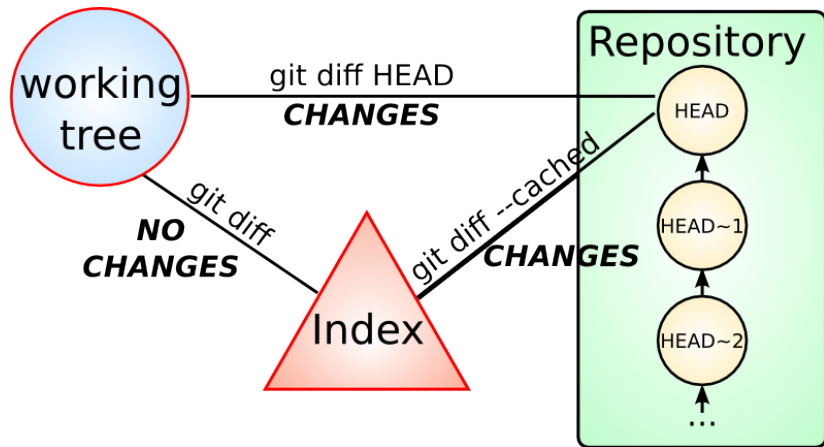
Typical beginners' workflow with git:

1. `git clone URL` or `git init .`
2. *Do stuff...*
3. `git add --all` or `git add FILE`
4. `git commit -m "Update"` *[or another bad description]*
5. `git push`

Best Practices

Adding Files

Adding Files

Do NOT:

- `git add *`
- `git add .`
- `git add --all`
- `git add -u .`
  ... and unless you're adding a new (unknown to index) file:
- `git add FILE_PATH`

Please **Do**:

git add --patch

Adding Files

```
git add --patch or git add -p
```

- asks you which chunks of code should be added and how!
- means you review your code before adding it!
- helps you create atomic commits!

Adding Files: Example

```
[...]

   if (delegateAuthLibPath)
     loadDelegateAuthLib(delegateAuthLibPath);
+
+  fprintf(stderr, "WHY THE HELL DOESN'T THIS WORK!!");
 }

 AuthChangeFsUid::~AuthChangeFsUid()
Stage this hunk \[y,n,q,a,d,/,j,J,g,e,?\]?
```

Committing

Committing

- Do **atomic commits**!
- Write useful, descriptive commit messages!
- Don't be afraid of committing (you'll see why later)!

Sherlock was investigating why the new version of his team's software didn't work...

Atomic Commits

He quickly looked at his favorite git UI (which uses `git log --oneline` log) for clues...

```
1d60cd2 Update for 1.5.0
817ee03 Innocent stuff...
8341828 Innocent stuff #1...
afee627 Innocent stuff #2...
d4c29b7 Innocent stuff #3...
c3bdb6d Innocent stuff #4...
a3e8c35 Innocent stuff #5...
fd9f14b Innocent stuff #6...
da8a2cf Innocent stuff #7...
c2debc0 Innocent stuff #8...
283293c Innocent stuff #9...
...
```

Atomic Commits

Not finding anything suspicious in the log, Sherlock starts a bisect in order to find the problem!

With all the compiling and testing, he spends hours in the bisect before finding the culprit:

1d60cd2 Update for 1.5.0

Atomic Commits

Sherlock finds out the commit has more to it than meets the eye...

```
commit 1d60cd23c6597f1d11288644691b582bd531e704
Author: Moriarty <moriarty.indeed@evil-r-us.co.uk>
Date: Thu Jun 6 18:06:06 2013 +0100

    Update for 1.5.0

    Evil stuff because I can!
    More evil stuff because I can!
    ...
```

Always write atomic commits!

Write useful commit messages

Write useful commit messages

A good commit message:

```
Imperative tense summary, <= 50 chars

When necessary, more details can come here, until 72 chars
each line.
A reference to a bug tracker issue can be added as well:

http://bugtracker.earth/issue/1985
```

Write useful commit messages

Bad

```
fixes a crash
```

Awful

```
fix
```

Good

```
Fix crash when performing update

The issue was being caused when the Updater was called
and a network connection that had been used before
is no longer available.

https://sherlocksbugtracker.co.uk/issue/1985
```

Know How to Use References

Know How to Use References

Specific References:

- The current commit: *HEAD*
- A hash from some commit, e.g.:
  d3a7c852d2c789f791b11091894cc71387e562e9
  Also works with just a prefix: *d3a7c85*
- A branch, e.g.: *master*, *newfeature*
- A tag, e.g.: *release0.9*

Know How to Use References

Relative References:

- Referring to previous commits:

```
Commit 1 position before: ref~1 or ref^
Commit 5 position before: ref~5 or ref^^^^^
E.g. a branch's parent commit: newfeature^
```

- Referring to different parents:

```
1st parent of a commit: ref^1
2nd parent of a commit: ref^2
Nth parent of a commit: ref^N
```

Use git bisect to track issues

Use git bisect to track issues

`git bisect` helps finding a commit that introduced a bug by making a binary search

```
$ git bisect start
$ git bisect bad
$ git bisect good HEAD~10
... [check until finding the culprit]
$ git bisect reset
... [fix it]
```

git reset

git reset

Want to undo adding a file (after `git add FILE_PATH`)?

```
git reset FILE_PATH
```

- The file is now in the Working Tree (only) again
- Index moved back

git reset

Want to undo the previous commit?

```
git reset HEAD^
```

- The changes are now only in the Working Tree (you have to add them again before committing)
- HEAD and Index moved back one step

git reset

Want to undo the previous commit but keep it ready to commit?

```
git reset -soft HEAD^
```

- The changes are now in the Index (ready to be committed)
- HEAD moved back one step

git reset

Want to delete the previous commit?

```
git reset -hard HEAD^
```

- It's as if the previous commit didn't happen
- HEAD, Index and Working Tree move back one step

git reflog

git reflog

Sometimes things don't go as expected! E.g.:

```
...
$ git commit -m "Very important commit"
...
$ git reset --hard HEAD^
$ git log --oneline
e897f7f Other things...
61be62b Other things #1...
...
$ NOOOOOOOOOOOOO
bash: NOOOOOOOOOOOOO: command not found...
```

git reflog

Luckily, `git reflog` comes to rescue!

```
$ git reflog
e897f7f HEAD@{0}: reset: moving to HEAD^
ca33ef2 HEAD@{1}: commit: Very important commit
e897f7f HEAD@{2}: checkout: moving from master
to importantfeature
...

$ git rebase ca33ef2
$ git log --oneline
ca33ef2 Very important commit
e897f7f Other things...
61be62b Other things #1...
...
```

Branches

A branch is simply a pointer to a commit!

(unlike other VCS which copied directories...)

Branches

- Create a new branch:
  ```
  git branch <new-branch> [<start-point>]
  ```
- Delete a branch:
  ```
  git branch -d|-D <branch>
  ```
- Rename a branch:
  ```
  git branch -m <old-name> <new-name>
  ```
- List branches:
  ```
  git branch [-r|-a]
  ```
- Move into (checkout) a branch:
  ```
  git checkout <branch>
  ```
- Create a branch and check it out:
  ```
  git checkout -b <branch>
  ```

Rebase a branch

master   A → B → E → F

mybranch   B → C → D

⬇

master   A → B → E → F

mybranch   F → C* → D*
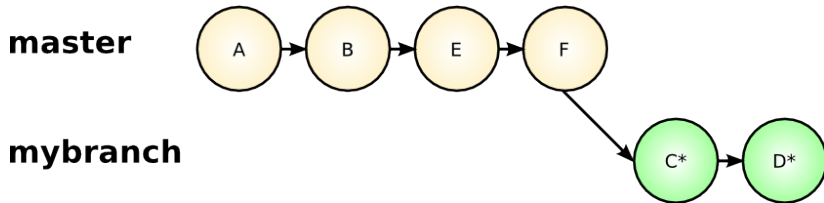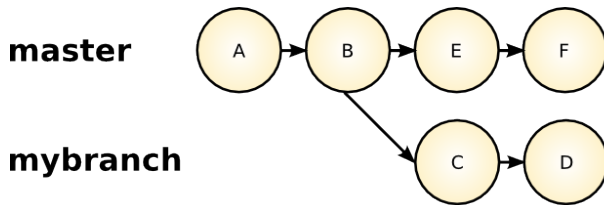
Merge two branches
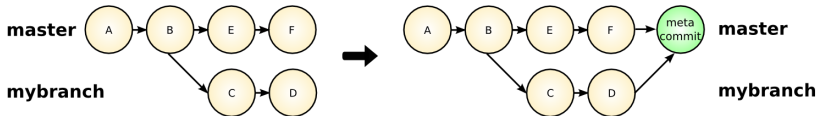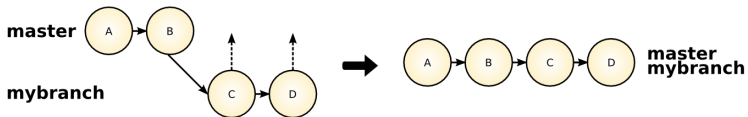
Branches

Pro tip:

- Use one branch per feature/bug (contained development)
- Only merge with master after you're done
- Remember to rebase your feature branch before merging it to master
- Specify the origin and branch when pushing (might avoid mistakes)
- A good use of branches should prevent the need of `git cherry-pick`

Branches

Proposed workflow:

```
... [we're on master]
$ git checkout -b newfeature
... [make changes]
$ git commit -m "New feature"
$ git rebase master
$ git checkout master
$ git merge newfeature
$ git push origin master
```

If push fails:

```
$ git pull --rebase
```

Working with Remote Branches

List remote branches:

```
$ git branch -r
remotes/origin/HEAD -> origin/master
remotes/origin/master
remotes/origin/newfeature
remotes/origin/newfeature2
remotes/origin/newfeature3
```

Branches

Push a branch to origin:

```
$ git checkout newfeature
$ git push origin newfeature
```

Delete a remote branch in origin:

```
$ git push origin :newfeature
```

Replace a remote branch in origin:

```
$ git push origin +otherfeature:newfeature
```

It's not advisable to replace branches you share with other people (it "breaks" other people's branch)

git rebase --interactive

git rebase --interactive

git rebase --interactive is a great tool that allows to:

- Meld commits together
- Remove commits
- Edit commits (it stops the rebase and allows to amend a commit)
- Reorder commits

git rebase --interactive

```
$ git rebase -i HEAD~5

pick d3a7c85 New changes
pick b538761 Other changes
pick 61be62b Some nice new changes
pick e897f7f Update for release 0.9.2
pick ca33ef2 Commit that should have been in the release

# Rebase 8d9b5a1..ca33ef2 onto 8d9b5a1
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
```

git rebase --interactive

```
pick d3a7c85 New changes
squash b538761 Other changes
pick 61be62b Some nice new changes
pick ca33ef2 Commit that should have been in the release
pick e897f7f Update for release 0.9.2

# Rebase 8d9b5a1..ca33ef2 onto 8d9b5a1
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
```

Name your stash

Name your stash

`git stash` is great for storing changes away temporarily

- Use `git stash save "Description of changes"` so it's easy to keep track of what's what!

Use repos for different modules

Use repos for different modules

Sometimes modules shouldn't be in the same repo

- A base + plugins project should live in different repos
  e.g.: GStreamer

Do NOT add every single file

Do NOT add every single file

Binary files, distro packages (rpms, debs) and other **auto-generated** files should not be included in the repo! Upload them somewhere else, a repo should be for source code.

Don't break others' repos

Don't break others' repos

Make sure that you do not change history in main branches (shared with other people)!

- Be careful when resetting and rebasing
- Do not force a push to a shared branch

Pull and rebase

Pull and rebase

Use `git pull --rebase` in order to avoid merges from upstream commits.

Other (Pro) Tips

Use git hooks to enforce standards

Git hooks can be used to e.g.:

- make sure the source honors a certain coding style
- force a commit message to include a bug tracker reference
- etc.

Be careful with git clean!

`git clean -xdf` will really delete things, no reflog, no nothing!
Always dry-run first!
`$ git clean -nxdf`

Use command aliases and auto-completion

Use git aliases and auto-completion for extra productivity:

- `git config --global alias.ci 'commit'`
- `git config --global alias.co 'checkout'`
- `git config --global alias.st 'status'`

Auto-completion:
`http://code-worrier.com/blog/autocomplete-git/`

Show your HEAD in shell's prompt

Change your shell's prompt to show your git branch:

http://code-worrier.com/blog/git-branch-in-bash-prompt/

[17:42][~/presentations/git-best-practices **(master)**]$

Do NOT copy/paste diffs

Create patches by using `git format-patch`

- Patches of the last 2 commits:

  ```
  $ git format-patch HEAD~2
  0001-Some-nice-changes.patch
  0002-Some-other-nice-changes.patch
  ```

You can even send them by email from git using `git send-email`!

See what your tree looks like

Having a global picture of your tree is important!

Use `git lola`:
`git config --global alias.lola "log --graph --decorate --pretty=oneline --abbrev-commit --all"`

... or use `tig` for an interactive CLI viewer

... or use or a graphical tool like *gitg*

gitg – eos (master)   ✕

File   Edit   View   Help

History   Commit

Branch: All branches   ⌄   🔍

Subject	Author	Date
⬤ origin/beryl 0.3.11 BUILD: move to 0.3.11	Andreas Peters	Fri 10 Jan 2014 10:24:05 AM
⬤ MGM: redirect reads of zero size file to the FSTs to allow par	Andreas Peters	Fri 10 Jan 2014 10:08:42 AM
⬤ 0.3.10 BUILD: move to 0.3.10	Andreas Peters	Wed 08 Jan 2014 10:58:12 A
⬤ Merge branch 'beryl' of eos.cern.ch:/var/repos/eos into beryl	Andreas Peters	Wed 08 Jan 2014 10:57:15 A
⬤ FST: fix double finalization of checksum in read case which	Andreas Peters	Wed 08 Jan 2014 10:55:00 A
⬤ MQ: revert patch using static error object (bug is in XrdXro	Andreas Peters	Tue 07 Jan 2014 01:20:56 PI
⬤ MGM: remove debug fprintf from ConfigEngine	Andreas Peters	Tue 07 Jan 2014 01:20:27 PI

· · · · ·

Details   Changes   Files

        SHA: ab423f956d29a6d9a8743441fd8b6c4a0b05b0e3

    Author: Andreas Peters <Andreas.Joachim.Peters@cern.ch> (Wed 08 Jan 2014 10:58:12 AM CET)

Committer: Andreas Peters <Andreas.Joachim.Peters@cern.ch> (Wed 08 Jan 2014 10:58:12 AM CET)

   Subject: **BUILD: move to 0.3.10**

    Parent: 67fb6cbbcbc0bb37ded9ddf2b10aa205a16eb52d : Merge branch 'beryl' of eos.cern.ch:/var/repos/eos into beryl

BUILD: move to 0.3.10

Loaded 3617 revisions in 0.60s

Thank you!

- Git's logo, CC by Jason Long
- Linus Torvalds picture, CC by The Linux Foundation
- Kid picture, CC by Juhan Sonin
- Monkey picture, CC by Scott Monty
- Slides and diagrams based in
  Git: The Stupid Content Tracker by Mario Sánchez Prada