GIT
(Version Control System)

D. Manubens, O. Mula

Outline

- Introduction
- Why GIT?
- Starting with GIT
- Making changes
- Branching
- Merging
- Usual commands
- References

Introduction

- Git is a distributed version control system that allow to handle very large projects with efficiency and with minimal knowledge.

<u>Project Information:</u>

- Original author(s) Linus Torvalds
- Developer(s) Junio Hamano, Linus Torvalds
- Stable release 1.7.10 (April 6, 2012)
- Written in C, Bourne Shell, Perl
- License GNU General Public License v2
- Website http://git-scm.com

Why GIT?

Not possible in SVN but possible in GIT:

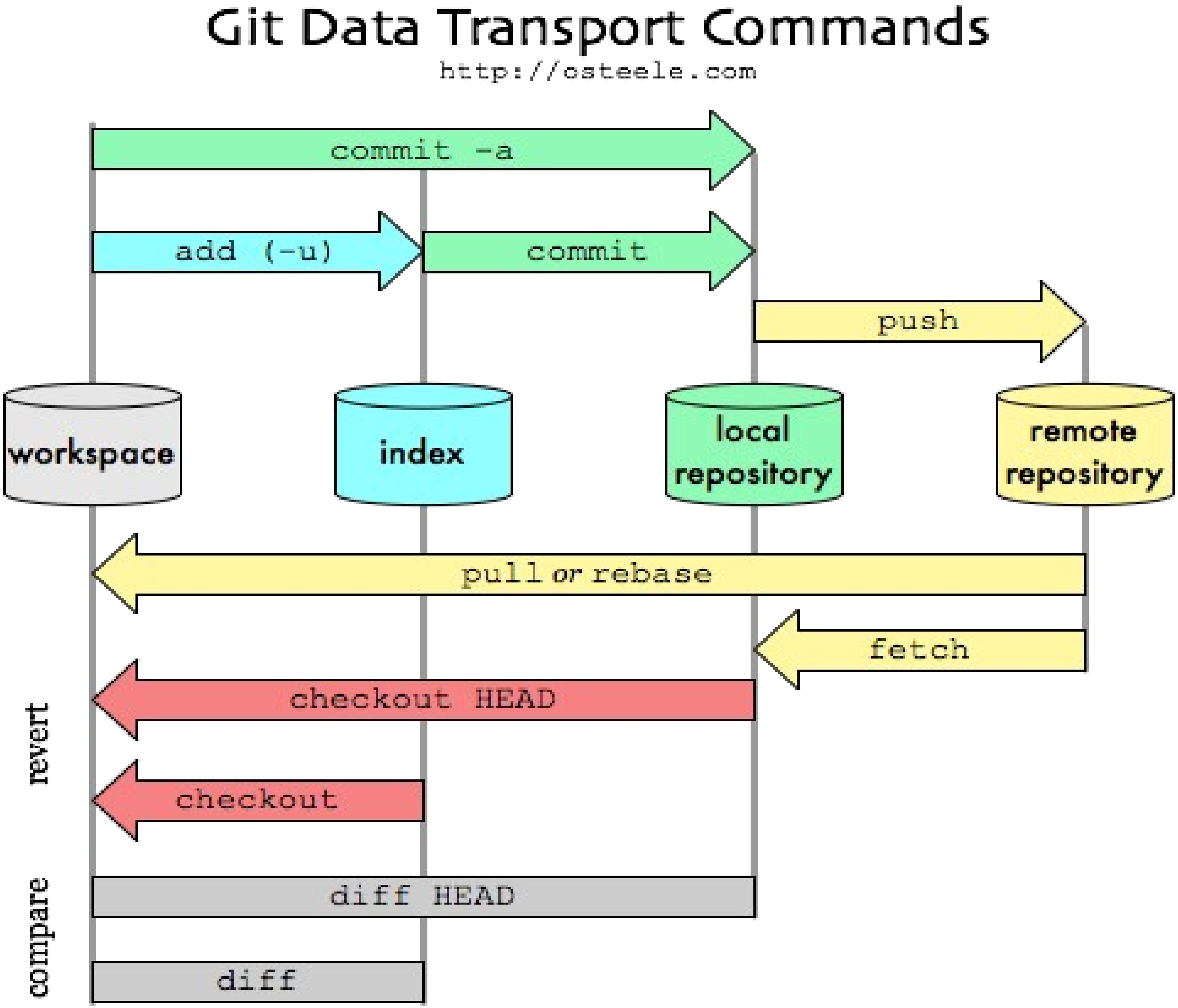
- Full clones server-independent
- Centralized but decentralized, sub-teams fetches
- Easy branching and merging

In addition:

- LDAP integration

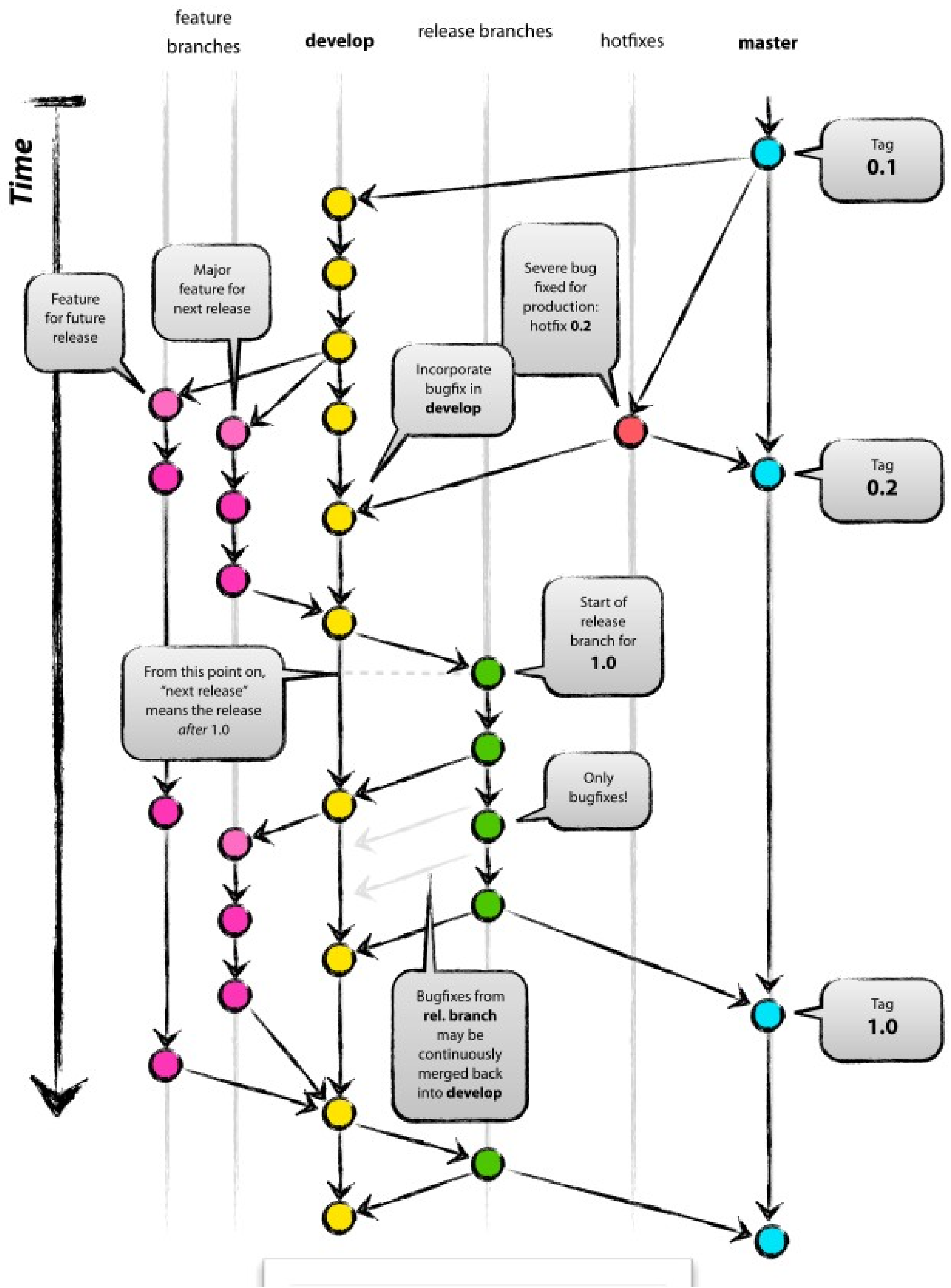
All this make GIT:

- More suitable for future developments


Git Data Transport Commands
http://osteele.com
commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
checkout
diff HEAD
diff
revert
compare

Climate Forecast

- “origin/**master**”: production-ready state
- “origin/**develop**”: state with the latest delivered development changes for the next release


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Severe bug fixed for production: hotfix 0.2
Major feature for next release
Feature for future release
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Starting with GIT

- git-core, git-doc installed on your machine
- Clone repository folders from the "origin" GIT remote:

```
git clone https://dev.cfu.local/autosubmit.git <localdir>
git clone https://dev.cfu.local/autosubmit3.git <localdir>
git clone https://dev.cfu.local/cfutools.git <localdir2>
git clone https://dev.cfu.local/cmor-specs.git <localdir2>
git clone https://dev.cfu.local/specs2cmor.git <localdir2>
```

- Clone makes a copy of all the repository branches (e.g. "origin/master",“origin/develop”,etc) in <localdir>

Starting with GIT (contd.)

- Before start working with “origin/master”: Fetch and merge changes that origin has but that you don’t yet have in your repository

```
git pull origin master
```

- Start working with the source code, as always, following the documentation instructions.

Making changes

- Jump to the development branch "develop":

```
git checkout develop
```

- Pull the latest version of "origin/develop":

```
git pull origin develop
```

- After working, check the status of “develop":

```
git status
```

Making changes (Contd.)

- Add new files (if new files created) and files we could have changed:

```
git add <file>...
```

- Commit the changes or contributions that we could do:

```
git commit -m “Explain what I changed”
```

- If you want to distribute your changes: upload the changes to the remote GIT "origin":

```
git push origin develop
```

- All changes uploaded to the “origin” generate a report which is sent automatically by e-mail to all users

Branching

- Jump to the branch from what I want to start (develop recommended):

```
git checkout develop
```

- Pull the latest version of "origin":

```
git pull origin develop
```

- Create a branch e.g. "develop-feature":

```
git branch develop-feature
```

- Switch to new branch:

```
git checkout develop-feature
```

- Work in the branch (edit files) Caution!: use **git mv, git rm**

Branching (contd.)

- Add new/changed files and commit:

```
git add <file>...
git commit -m “Explain what I changed”
```

- (If distribution needed) upload the changes to the remote GIT "origin":

```
git push origin develop-feature
```

Merging

- Switch to branch “develop” (or the branch to put the changes to):

```
git checkout develop
```

- Pull the latest version of "origin":

```
git pull origin develop
```

- Merge:

```
git merge --no-ff develop-feature
```

- (if needed) Resolve conflicts and commit
- (Optional) Delete “develop-feature” branch:

```
git branch -d develop-feature
```

- Push:

```
git push origin develop
```

Merging master

- Switch to branch “master”:

```
git checkout master
```

- Pull the latest version of "origin":

```
git pull origin master
```

- Merge:

```
git merge --no-ff develop
```

- Resolve conflicts if needed and commit
- Push:

```
git push origin master
```

Usual commands

- View the status of the local copy of the repository:

git status

- See revision history, with the changes made by users in the repository:

git log

- See more information from origin

git remote show origin

- Info:

man git

Usual commands

- Global config

```
git config --global user.name "Your Name"
git config --global user.email yourmail@yourdomain.org
```

- See file changes w.r.t. previous commit

```
git log --stat
```

- See revision history (pretty mode):

```
git log --graph --decorate --pretty=oneline --abbrev-commit –all
```

- Amend latest commit:

```
git commit --amend
```

Usual commands

- Discard changes from local copy of a file (not commited)

```
git checkout -- <file>...
```

- Restore previous version of a file (commited):

```
git reset HEAD <file>...
```

- Get a file from a specific revision

```
git reset HEAD~4 <file>...
```

- See a file form a specific revision

```
git show HEAD~4:<file>
```

- Usual linux commands

```
git grep, git diff, git mv, git rm
```

References

- GIT web site: http://git-scm.com/
- GIT book: http://git-scm.com/book
- GIT ready: http://gitready.com/
- http://nvie.com/posts/a-successful-git-branching-model/
- http://www.gnulinux.cat/documentacio/git-guia-daprenentatge/
- http://hpckp.org/images/conference/HPCKP-11/GIT_introductio

GIT
(Version Control System)

O. Mula, D. Manubens