git – the SCM system

Jan-Simon Möller

training.linuxfoundation.org
Topics

- What is git (what is a SCM)
- How to install git
- How to personalize git
- How to use git for development
What is git?
What is a SCM System?
Source Code Management

- short: SCM
- also known as: Version Control
- component of: Software Configuration Management

Used to track:
- Changes to documents or source-code
- revisions ↔ timestamp ↔ author ↔ other metadata
Trunk, Branch, Tag?

- The central "golden thread" is called "trunk"
- **Branches** can fork from the trunk, **merged** or be **discontinued**
- **Tags** serve as labels/pointers
SCM system challenges

- Multiple Users modify code at the same time
- Even quasi-concurrent edits of the same file
- Central/Decentralized operation
- File locking (old approach)
- Atomic operations (not all SCMs)
- Merging (from painful to easy)
History and evolution of SCM tools

- RCS / SCCS
- CVS, SVN,
- ClearCase, Perforce, ...
- Distributed: git, Bazaar, Darcs, arch, Mercurial
- Local File
- Client-Server
- Distributed
- Distributed version control system
- Open Source
- created by Linus Torvalds in April 2005
  as replacement for the proprietary tool „bitkeeper“
- Maintainer since July 2005 Junio Hamano
Design goals

• Patching in $\leq 3$ seconds
• !CVS (not like cvs)
• distributed workflow ($\rightarrow$ Kernel)
• safeguards against corruption
• Debian/Ubuntu:
  – apt-get install git
• CentOS/Fedora/RHEL
  – yum install git
  – dnf install git
• OpenSUSE
  – zypper install git
Personalize (username/email)

- `git config --global user.name "testuser"`
- `git config --global user.email "testuser@example.com"`
Typical workflow

- Clone the repo
- Do your work
- Add & commit
- Amend
- Push

*) we will go through examples next, but wait for the lab part before you try
Clone the repo

- **URL=** https://git.opnfv.org/sandbox.git

```bash
# preparations
$~/> export URL="https://git.opnfv.org/sandbox"
$~/> mkdir test
$~/> cd test

# clone the repository
$~/test/> git clone $URL

or
$~/test/> git clone https://git.opnfv.org/sandbox
```
Do some work

- Create a new file <yourname>.txt containing „Hello World!“

```
~/test/> cd sandbox
~/test/sandbox/> echo 'Hello World!' > jansimon.txt
~/test/sandbox/> git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

  jansimon.txt

nothing added to commit but untracked files present
( use "git add" to track)
```
State of a file in git

Untracked

Add the file

Unmodified

Edit the file

Modified

Stage the file

Staged

Remove the file

Commit

OPNFV
Stage a file for a commit

```
$~/test/sandbox/> git add jansimon.txt
```
Stage a file for a commit

$~/test/sandbox/> git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:  jansimon.txt

$~/test/sandbox/>
Commit the file to your local repository

$~/test/sandbox/> git commit --signoff -m“Initial commit.”
[master d7b98bc] Initial commit.
  1 file changed, 1 insertion(+)
  create mode 100644 jansimon.txt
$~/test/sandbox/>
Initial empty repository

Author: Jan-Simon Möller <jsmoeller@linuxfoundation.org> 2016-01-13
Committer: Jan-Simon Möller <jsmoeller@linuxfoundation.org> 2016-01-13
Parent: 4e745ada6420f9cb97b25cd0fc862a65320a2d495a (Initial empty repository)
Branch: master
Follows:
Precedes:

Signed-off-by: Jan-Simon Möller <jsmoeller@linuxfoundation.org>

-----------

new file mode 100644
index 6900000..5983a8d5
@@ -0,0 +1 @@
+Hello World!
Commit Messages

Title: 50 chars max
Add new base for common puppet modules

Blank line

Detailed description
Intent of this commit is to be a common place where we can contain a common set of puppet modules that installers should leverage when installing/configuring a target system.

Optional: Track modifications (e.g. on ML)
v3: fixed typos
v2: added 80 col linebreaks

Optional: References
(url, ml-post, change-id)

Sign-off with name and email
Signed-off-by: Jan-Simon Möller <dl9pf@gmx.de>
Commit Messages

- Provide a brief description of the change in the first line.
- The first line should be limited to 50 characters and should not end with a period.
- Insert a single blank line after the first line.
- Provide a detailed description of the change in the following lines. Use breaking paragraphs where needed.
- Subsequent lines should be wrapped at 72 characters.
- Use present tense.
• Let's adapt the commit with a proper message:

```bash
~/test/sandbox/> git commit --amend
## THIS WILL OPEN AN EDITOR WINDOW
## ( change with $~/> export EDITOR=/usr/bin/nano )

~/test/sandbox/>
```
This is my very first jansimon.txt

This patch adds the very first version of a jansimon.txt file as hello world example.

Signed-off-by: Jan-Simon Möller <jsmoeller@linuxfoundation.org>

# Please enter the commit message for your changes. Lines starting with '#' will be ignored, and an empty message aborts the commit.
#
# Date:       Wed Jan 13 11:49:23 2016 +0100
#
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#  (use "git push" to publish your local commits)
Push

- Synchronize a remote repository with your local repository

```
~/test/sandbox/> # git push <remote> <localbranch>:<remotebranch>
```

```
~/test/sandbox/> git push # (using defaults, e.g. remote=origin)
or
~/test/sandbox/> git push origin
or
~/test/sandbox/> git push origin master:master
```

**ATTENTION:** Different when using gerrit for code review!
Branches

• It is easy to create a branch and call checkout.

```
$~/test/sandbox/> git branch
  * master

$~/test/sandbox/> git branch mytopic

$~/test/sandbox/> git checkout mytopic

$~/test/sandbox/> git branch
  master
  * mytopic

$~/test/sandbox/>```
Branches

• Git makes creating local branches easy
• Branches are useful to keep topics separate
• Branches allow you to track, amend and resubmit changes under review
Changes to an existing (tracked) file

- Changes need to be staged and committed
Change existing file

- Lets adapt `<yourname>.txt` with the current date:

```
$~/test/sandbox/> date >> jansimon.txt
$~/test/sandbox/> git diff
diff --git a/jansimon.txt b/jansimon.txt
index 980a0d5..0c57ebe 100644
--- a/jansimon.txt
+++ b/jansimon.txt
@@ -1 +1,2 @@
 Hello World!
+Mon Jan 11 12:10:47 CET 2016
$~/test/sandbox/>
Check the status

- Let's check with 'git status':

```
~/test/sandbox/> git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes
      in working directory)

    modified:   jansimon.txt

no changes added to commit (use "git add" and/or "git commit -a")
```
Let's stage the file for the commit:

```
~/test/sandbox/> git add jansimon.txt
~/test/sandbox/> git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:  jansimon.txt

~/test/sandbox/>
```
Commit change

- Let's commit the change

```bash
$~test/sandbox/> git commit -s -m 'Add date to <name>.txt'
$~test/sandbox/> git log -1
commit 5af6f74203d0b50f0cb465184985217baf0ee521
Author: Jan-Simon Möller <jsmoeller@linuxfoundation.org>
Date:   Mon Jan 11 12:17:22 2016 +0100

    Add date to <name>.txt

    Signed-off-by: Jan-Simon Möller <jsmoeller@linuxfoundation.org>

$~/test/sandbox/>
Switching branches and merging

- Back to master, add another file, ...

```bash
~/test/sandbox/> git checkout master
~/test/sandbox/> cat jansimon.txt
Hello World!
~/test/sandbox/> date >> killroy-was-here.txt
~/test/sandbox/> git add killroy-was-here.txt
~/test/sandbox/> git commit -s -m 'Add date to killroy-was-here.txt'
```

![Git branch diagram](image)
Switching branches and merging

- ... merge branch mytopic

```bash
$~/test/sandbox/> git merge mytopic

Merge made by the 'recursive' strategy.
jansimon.txt | 1 +
1 file changed, 1 insertion(+)

$~/test/sandbox/> cat jansimon.txt
Hello World!

$~/test/sandbox/> ls
jansimon.txt  killroy-was-here.txt
```
State of a file in git

- Untracked
- Unmodified
- Modified
- Staged

- Add the file
- Edit the file
- Stage the file
- Commit
- Remove the file
Some advanced commands
git init

- Used to initialize a folder as git repository

```
$~/test/> mkdir newrepo ; cd newrepo
$~/test/newrepo/> git init
Initialized empty Git repository in /home/user/test/newrepo/.git/
$~/test/newrepo/> tree -a -L 1
.git
|-- HEAD
|-- branches
|-- config
|-- description
|-- hooks
|-- info
|-- objects
 `-- refs
5 directories, 3 files
```
git remote

- Used to add/change connections to remote repositories

```
$~/test/newrepo/> export URL='https://github.com/dl9pf/example.git'
$~/test/newrepo/> export PUSHURL='git@github.com:dl9pf/example.git'

$~/test/newrepo/> git remote add origin $URL
$~/test/newrepo/> git remote -v
origin https://github.com/dl9pf/example.git (fetch)
origin https://github.com/dl9pf/example.git (push)

$~/test/newrepo/> git remote set-url --push origin $PUSHURL
$~/test/newrepo/> git remote -v
origin https://github.com/dl9pf/example.git (fetch)
origin git@github.com:dl9pf/example.git (push)
```
git fetch

- git fetch synchronizes the local database with a remote
- does update the working directory for tracking branches

$~/test/newrepo/> git fetch
remote: Counting objects: 8, done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 8
Unpacking objects: 100% (8/8), done.
From https://github.com/dl9pf/example
* [new branch] master -> origin/master
* [new branch] otherbranch -> origin/otherbranch
* [new branch] yetanotherbranch -> origin/yetanotherbranch
git pull

- Fetch from and integrate with another repository or a local branch

```bash
~/test/newrepo/> git pull origin otherbranch
From https://github.com/dl9pf/example
 * branch            otherbranch -> FETCH_HEAD
Updating 18947fa..d320f58
Fast-forward
  other  | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 other
```
git push

- Update a remote repository with the objects specified

```
$~/test/newrepo/> git push origin master:yetanotherbranch
Total 0 (delta 0), reused 0 (delta 0)
To git@github.com:dl9pf/example.git
  * [new branch] master -> yetanotherbranch
```

```
## git push <remotename> <sourcebranch>:<destinationbranch>
```
git rebase

- Forward-port local commits to the new state of the branch
- Useful for clean history - BUT no-go if you're the maintainer

$~/test/newrepo/> $~/test/newrepo/> git checkout otherbranch
$~/test/newrepo/> git rebase master
First, rewinding head to replay your work on top of it...
Applying: other

![Diagram showing repo structure before and after rebase]
Resources

- http://wiki.opnfv.org/developer/getting_started
- http://training.linuxfoundation.org
  - free videos → introduction-to-git
  - LFD262 - Developing with GIT
- More tutorials on the net:
  - rogerdudler.github.io/git-guide
  - marklodato.github.io/visual-git-guide
The labs are integrated with next chapter on 'gerrit'
End