



# USING GIT FOR ENSTORE

---

DMD meeting

Alex Kulyavtsev

2/15/2013

# What is git?

- git is distributed version control system
- written by Linus Torvalds for Linux kernel development
- allows easy management of large complex code base
- manipulates multiple code change streams
- large user base
- common code repositories like `github`
- integrated in many IDE, has GUI tools

# Git Features

- git is offline and fast.
  - Most of the time you access files on the local disk
- snapshot based, not changelogs like cvs
- git stores snapshot of the project when you commit the change

# Git Features

- local code management (think RCS)
  - history
  - branches
- remote repository access (think CVS, SVN)
  - distributed collaboration
  - all distributed repositories “are equal”
  - remote branches can be accessed like local branches
- has familiar concepts:
  - code repository
  - working tree
  - checkout, commit
  - branches, tags
  - status, diff, log (history)

# Git Features

- git is fast and it is easy to switch to the different branch and switch back or stash the work
- very easy manipulation of code in different branches to accept changes to the files
  - create new branch
  - merge
  - rebase
- git can store binary files
- now you can delete that old empty directory, unlike the CVS
- access rights: who can do what in each subtree

# Enstore git on redmine

- Enstore git test subproject on Fermilabs' Redmine  
<https://cdcvs.fnal.gov/redmine/projects/enstore-git-test>
- Test drive for enstore code converted to git repository
  - right now code is outdated on cdcvs, will be updated soon
- Repository contains imported enstore code with preserved
  - branches
  - modifications history
  - browse, diff source tree
- Wiki has tips
  - how to pull enstore code from enstore git repository
  - links to git documentation

# Repositories and Access

- code is stored in git *repositories*
- all repositories for the same project “are equal”
  - there is no “central repository” in git
  - but we all agree to have one repository as central repository to keep production branch and other branches
  - test enstore repository is on on redmine server cdcvs.fnal.gov
  - We *fetch* or *pull* files from remote repository to local repository and *push* files to the remote repository
- access to *remote* directory through protocols like git, http(s), ssh. URL looks like:  
<https://github.com/git/git.git>  
ssh://p-enstore-git-test@cdcvs.fnal.gov/cvs/projects/enstore-git-test  
user@mysrv.fnal.gov:/opt/enstore-git-test/enstore.git

# Hashes

- As we commit changes to repository few files are changing
- All unchanged files in repository refer to the same file in .git subtree (through SHA hash)
- Similar, only some files are changing when we change branches
- When we checkout previous revision, branch, pull code from remote git checks hash and only few files are updated therefore updates are fast.

# Hashes

- repository files are stored in Unix file system as files in .git subtree
- for each repository file the file SHA hash calculated and it used as file name in repository (but there blobs)
- there is index which maps files in working directory to the hashed files in repository
- instead of CVS file versions like 1.2 or 4.3.1.2 you may see long hex numbers like for commit id  
`commit 7bd3a3bfe9d38da2c1283d331c160be21c08f1bb`
- you may type only 7-8 first hex digits when needed
- normally you access file by tag or date, or ‘two revisions before”

# Creating git repository: create new

- Two ways to *create local git repository*
- create empty directory, initialize and put files
  - `mkdir myproj`
  - `cd myproj`
  - `git init`
- OR initialize git repository in existing tree:
  - `cd /opt/enstore`
  - `git init`
- In either case the new subdirectory `.git` is created by `init` where git keeps its files:
  - repository, index, and other files
- There are NO files in repository yet
- Unlike CVS or SVN, there are no extra files in each working subdirectory
- maybe except `.gitignore` if you create it there

# Staging area or Index

- *git add* is not the same as *cvs add*
- There is *staging area (index)* between *working tree* and *repository*
- Saving files in local repository is two step process:
- *git add* “marks” files which need to be stored into repository when you do next commit and stores them in staging area
- *git commit* stores ‘marked’ files into *repository*
  - <<change file1 file2 file3>>
  - *git add* file1 file2
  - *git commit -m* “some changes”
  - *git add* file3
  - <<change file4>>
  - *git add* file4
  - *git commit -m* “more changes”

# git add

- You need to do 'git add' each time you changed the file and tell git you want to stage it
- There are shortcuts:
- If git tracks the file (you did "git add")
  - `git commit -m "the test 2" -- test.txt`
- to save all changed files, git does 'add' and then 'commit'
  - `git commit -a -m "all my changes"`
- beware you may commit some old change.

# Some other Commands

- Now we have files in repository, try
  - git help
  - git help checkout
  - man git-checkout
  
- git status
- git status -s
- git status -uno
  
- git log myfile
- git log --oneline
- git log --oneline --graph
  
- git diff myfile
- There are modifications of some of these commands to operate files in cache, to access previous revisions of files

# Branching

- `$ git branch`
  - \* master

shows the current branch. “**master**” is the name of default branch created when you initialized git

- Create new branch

```
$ git branch cdf_test_config
```

You're still at master. Change files and commit

```
$ git commit -am "committing all changed files"
```

Switch to new branch

```
$ git checkout cdf_test
```

Do some work, commit to test branch

```
$ emacs some_file
```

```
$ git commit -am "committing all changes to test branch"
```

Get back to your default branch

```
$ git checkout master
```

```
$ git status
```

# Branches, Tags

- When creating new branch you may want to switch to it immediately:  
`git checkout -b newbranch`
- is the same as  
`git branch newbranch`  
`git checkout newbranch`
- Create annotated tag all files in current branch  
`git tag -a v1.0`

Now you can refer to files in this snapshot by this tag

# Remote Repositories

- The other way to start work and create local repository is to copy files from some other git remote (upstream) repository.
  - `$ git clone git://github.com/someuser/somerepo.git`
- You can check, set, remove alias for remote directory:
- To simplify typing use aliases. Check remote alias defined

```
$ git remote -v
my-repo      ssh://z.fnal.gov//opt/en.git (fetch)
my-repo      ssh://z.fnal.gov//opt/en.git (push)
```
- Add new repository with alias “github”

```
git remote add en-srv mysrv.fnal.gov:/opt/enstore-git-test/enstore.git
git remote add en-cdsrv \  
ssh://p-enstore-git-test@cdcvs.fnal.gov/cvs/projects/enstore-git-test e-g-t--git-clone
```

alias ‘origin’ is set as default name for the URL after you fetched files from remote server, you can check URL with “git remote”

# Remote Repositories

- Get all branches from remote server to local repository, but do not update working tree

```
git fetch en-cdsrv
```

```
git fetch <full-URL>
```

- ‘fetch’ copies files to local repository but does not change work tree. To get these files you need to do merge
- `pull` is equivalent of `fetch + merge` for the remote branch with current branch, like “`cvs checkout`”

- Publish code from local repository to remote

```
git push master branch_1_0
```

```
git push en-cdsrv my_test_branch
```

# Remote Repositories

- remote branches used like local branches
- merge branch 'production' from remote repository 'enstore' to the current branch:  
`git merge enstore/production`
- List changes on remote server since last update:  
`git log origin/master ^master`
- Update all branches from all remote repositories  
`git fetch --all`

# Links

- Intro

<http://gitref.org>

- Git visual cheat sheet:

[http://ndpsoftware.com/git-cheatsheet.html#loc=workspace;](http://ndpsoftware.com/git-cheatsheet.html#loc=workspace)

Internals:

<http://ftp.newartisans.com/pub/git.from.bottom.up.pdf>

- Enstore git test subproject on Fermilabs' Redmine

<https://cdcvs.fnal.gov/redmine/projects/enstore-git-test>

# Questions ?

-