**Check version**

$git –version

Git version 1.7.4

**Configuration**

Config file is under home directory (.gitconfig) global

System /etc/gitconfig

Change global setting(current user)

$git config --global user.name “First Last”

$git config --global user.email [firstlast@123.com](mailto:firstlast@123.com)

$git config --global color.ui true

Change system setting (all users) user must have administrator privilege

$sudo git config --system alias.st status

$sudo git config –system alias.ci commit

Edit config file

$cd /path/to/my/workspace/demo

$git config –e

$git config –e –global

$git config –e –system

Read value

$git config <section>.<key>

[core]’

Bare = false

$git config core.bare

False

Write value

$git config <section>.<key> <value>

$git config a.b something

[a]

B = something

$git config x.y.z others

[x “y”]

Z = others

**Use git config to change other INI file(config)**

$GIT\_CONFIG=test.ini git config a.b.c.d “hello, world”

Read

$GIT\_CONFIG=test.ini git config a.b.c.d

Hello, world

Delete setting

$git config –unset –global user.name

**Git initialization(init)**

$cd /path/to/my/workspace

$mkdir demo

$cd demo

$git init

Initializaed empty Git repository in /path/to/my/workspace/demo/.git/

If git version is higher than 1.6.5

$cd /path/to/my/workspace

$git init demo

Initializaed empty Git repository in /path/to/my/workspace/demo/.git/

$cd demo

**Add file to be commited(add)**

$git add <file name>

**Commit file to version control system(commit)**

$git commit –m “initialized”

*-m*

Add updates comments

Allow submit empty file

--allow-empty

$git commit –allow-empty

--*amend*

Fix the submission info

*--reset-author*

Fix author id, date

**Search files(grep)**

$git grep “search content”

**Check version status(status)**

Trace search path

$strace –e ‘trace=file’ git status

$git status

-s

Use simple format

$git status –s

M filename

M means modified. If M is displayed in col 1, this file is modified between Repository and Stage(2 and 3).

If M is displayed in col 2, this file is modified between work copy and stage(1 and 2).

Repository (3)

Stage (2)

Work Copy (1)

**Find .git location**

Find .git folder location

$cd /path/to/my/workspace/demo/

$mkdir –p a/b/c

$cd /path/to/my/workspace/demo/a/b/c

$git rev-parse –show-toplevel

/path/to/my/workspace/demo

$git rev-parse –show-prefix

a/b/c

$git rev-parse –show-cdup

../../../

**Log info(log)**

–pretty=fuller

Use full format

$git log –pretty=fuller

--pretty=oneline

Use simple format

$git log –pretty=oneline

–stat

Display file change history

$cd /path/to/my/workspace/demo

$git log –stat

Commit 40 bit commit id

Author: xxxx xxx<email>

Date:

**Compare file(diff)**

$git diff

Diff –git a/xxxxx.filename b/xxxxx.filename

Index

--- a/filename

+++ b/filename

HEAD

$git diff HEAD

Stage (2)

Work Copy (1)

Repository (3)

No paramter

$git diff

Compare the differences between work copy and stage (1 and 2)

HEAD

$git diff HEAD

Compare the differences between work copy and repository (1 and 3)

--cached, --staged

$git diff –cached or –staged

Compare the differences between stage and repository(2 and 3)

**Check reference (show)**

$git show-ref

Commit id refs/heads/master

Commit id refs/remotes/origin/HEAD

Commit id refs/remotes/origin/master

Commit id refs/tags/A

Refs/heads are branches

Refs/remotes are remote branches local reference.

Refs/tags are milestone

**Withdraw the changes (checkout)**

$git checkout – welcome.txt

$git checkout . (all files)

All change in work copy will be gone.

**Clean up file(clean)**

$cd /path/to/my/workspace/demo

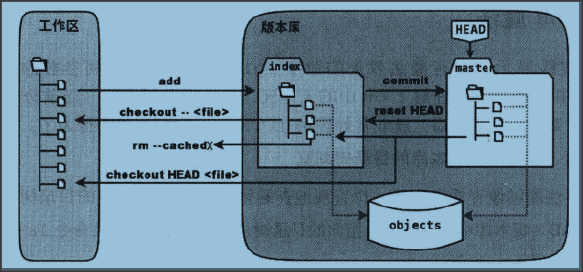
$git clean –fd

$git checkout .

Clean up the files in work copy which haven’t been added to stage

Replace the rest of files with stage.

**Unstage file(reset or checkout)**



**Git add(add file from work copy to stage)**

**Git checkout --<file> (work copy will be replaced by stage)**

**Git checkout HEAD. Or Git checkout HEAD <file> (work copy and stage file will be replaced by repository – HEAD)**

**Git commit(submit file from stage to repository - HEAD)**

**Git reset HEAD(stage will be replaced by repository –HEAD)**

**Git rm –cached <file> (the file will be deleted from stage)**

**Clone repository (Clone)**

$git clone

Copy repository A to another repository B. Use “push” and “pull” to sync up the changes between A and B

A clone B

A push to B

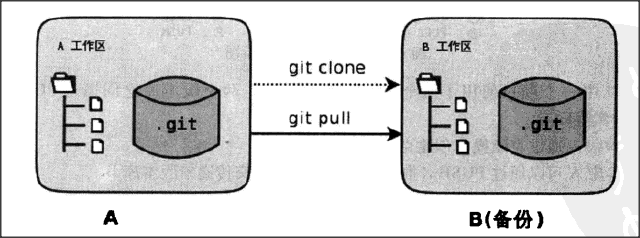
A pull from B

B pull from A

B push to A

$git clone <repository> <directory>

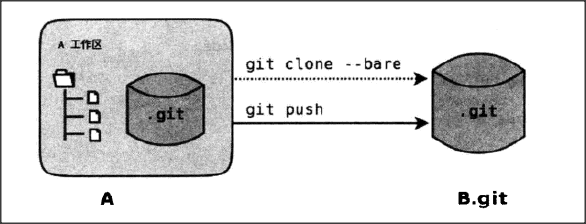
Clone <repository> HEAD version to <directory> folder. <directory> is a work copy of <repository>. The file objects is under .git folder.



*--bare*

$ git clone –bare <repository> <directory.git>

Clone <repository> HEAD version to directory.git. This will only generate file version info (only file under .git folder in above command). No work copy.



*--mirror*

$git clone –mirror <repository> <directory.git>

Same as –bare command, but this will remember the remote file version. So you can use git fetch to sync up with remote file version (only file version, no work copy).

**Push/Pull file (Push, Pull)**

$git push [<remote-repos>] [<refspec>]

$git pull [<remote-repos>] [<refspec>]

<remote-repos> the remote repository address or name

<refspec> reference to file version

**GitHub Tutorial**

http://www.worldhello.net/gotgithub/03-project-hosting/020-repo-operation.html

**Account setup**

Sign up git hub account (use rim credential for github.rim.net)

Add ssh key to your profile.

Command:

Ssh-keygen –c “emailaddress” –f ~/.ssh

This will generate two key(public and private) under .ssh folder. Id\_rsa.pub is public key, id\_rsa is private key. Open public key in a text editor tool and copy and paste into your github account profile.

**Create repository**

**New project:**

click new repository. Type project name, description and homepage URL.

If this is a new repository, we can do a clone of this empty project then commit changes, and push to github

$git clone [git@github.com:<username>/<project>.git](mailto:git@github.com:%3cusername%3e/%3cproject%3e.git)

Create readme.md file

$git add readme.md

$git commit –m “readme for this project”.

$git push origin master

**Start with an existing project**

In local computer

$mkdir <project>

$cd <project>

$git init

Add readme.md file

$git add README.md

$git commit –m “README for this project”

Add remote “origin”

$git remote add origin [git@github.com:<username>/<prjoect>.git](mailto:git@github.com:%3cusername%3e/%3cprjoect%3e.git)

$git push –u origin master

**Create Branch**

Git branch was a reference which saved under ./git/refs/heads/. Content is the branch commit id

For example, the default master -> .git/refs/heads/master

How to create a branch in github

1. Create new branch in local
2. Push the new branch from local to github

Create new branch mybranch1

$git checkout –b mybranch1

$touch hello1

$git add hello1

$git commit –m “add hello1 from mark”

$git push –u origin mybranch1

Go to github, you will be able to see a new branch “mybranch1” on the webpage.

**Setup default branch**

Click “Admin”, change default branch in “Options”

After change the default branch, if you clone repo from github, the default branch will be pointing to default branch.

$ git branch -r

origin/HEAD -> origin/mybranch1

origin/master

origin/mybranch1

HEAD and mybranch1 pointing to the same location

$ git ls-remote

From git@github.com:gotgithub/helloworld.git

f46a28484adb6c1b4830eb4df582325c740e9d6c HEAD

e1e52d99fa71fd6f606903efa9da04fd0055fca9 refs/heads/master

f46a28484adb6c1b4830eb4df582325c740e9d6c refs/heads/mybranch1

**Delete a branch**

$ git branch -d mybranch1

error: Cannot delete the branch 'mybranch1' which you are currently on.

$ git checkout master

$ git branch

\* master

mybranch1

Switch to master branch.

$ git branch -D mybranch1

Deleted branch mybranch1 (was f46a284).

By default, git doesn’t allow to delete a branch before it merges. Use –D to ignore this warning.

**Delete branch from github**

Have to use git push instead of git branch to delete a branch from github

$ git push origin :mybranch1

remote: error: refusing to delete the current branch: refs/heads/mybranch1

To git@github.com:gotgithub/helloworld.git

! [remote rejected] mybranch1 (deletion of the current branch prohibited)

error: failed to push some refs to 'git@github.com:gotgithub/helloworld.git'

change default branch to master

$ git push origin :mybranch1

To git@github.com:gotgithub/helloworld.git

- [deleted] mybranch1

list remote branches.

$git ls-remote [git@github.com:<username>/<project>.git](mailto:git@github.com:%3cusername%3e/%3cproject%3e.git) list all remote branch

$ git ls-remote git@github.com:gotgithub/helloworld.git

From git@github.com:gotgithub/helloworld.git

e1e52d99fa71fd6f606903efa9da04fd0055fca9 HEAD

e1e52d99fa71fd6f606903efa9da04fd0055fca9 refs/heads/master

**Manage Tags**

Tag is very similar to branch but it is saved under .git/refs/tags/

Light way, create a tag refering to a <commit>

$git tag <tagname> [<commit>]

With descriptions, create a tag object with descriptions.

$git tag –a <tagname> [<commit>]

With signatures.

$git tag –s <tagname> [<commit>]

Example:

Add a file and commit it

$ touch hello1

$ git add hello1

$ git commit -m "add hello1 for mark."

Create mytag1, mytag2 and mytag3

$ git tag -m "Tag on initial commit" mytag1 HEAD^

$ git tag -m "Tag on new commit" mytag2

$ git tag mytag3

Check the new tags

$ git tag -l -n1

mytag1 Tag on initial commit

mytag2 Tag on new commit

mytag3 add hello1 for mark.

Push tags to github

$ git push origin refs/tags/\*

Counting objects: 6, done.

Delta compression using up to 2 threads.

Compressing objects: 100% (4/4), done.

Writing objects: 100% (5/5), 548 bytes, done.

Total 5 (delta 0), reused 0 (delta 0)

To git@github.com:gotgithub/helloworld.git

\* [new tag] mytag1 -> mytag1

\* [new tag] mytag2 -> mytag2

\* [new tag] mytag3 -> mytag3

**Delete tags from local**

$ git tag -d mytag3

Deleted tag 'mytag3' (was c71231c)

**Delete tags from github**

$ git push origin :mytag3

To git@github.com:gotgithub/helloworld.git

[deleted] mytag3