

# Introduction to Git

Yonglei Tao  
GVSU

# Manajemen Perubahan File / Version Control System

---

- ▶ Version Control System adalah sistem yang mengelola suatu perubahan pada file dokumen, source code, atau kumpulan informasi lainnya. VCS mencatat setiap perubahan pada file yang dikerjakan oleh seseorang.

## Jenis-jenis Version Control System :

- Mercurial
- Git
- Subversion
- CVS









# Summary GIT dan GitHub

- ▶ Git didirikan oleh Linus Torvalds di tahun 2005
- ▶ Git berguna untuk membantu tim dalam berkolaborasi dalam membangun dan mengembangkan sebuah project, dengan Git, pekerjaan menjadi lebih praktis. Kita dapat melacak perubahan pada berkas yang ada dalam repository atau direktori kerja. Selain itu, kita juga dapat mengelola versi rilis dari sebuah proyek dalam repository.
- ▶ Penggunaan Git memerlukan adanya repository, yaitu wadah atau tempat penyimpanan proyek di mana setiap berkas yang ada di dalamnya dapat dilacak jika terjadi perubahan. Berdasarkan jenisnya, repository terbagi menjadi 2, yaitu local repository dan remote repository.
- ▶ Local repository merupakan tempat penyimpanan lokal yang berada di komputer kita. Local repository dapat kita ubah-ubah (hapus, modifikasi, dan tambah) sesuai dengan keinginan kita, sebelum akhirnya nanti di-push.
- ▶ Remote Repository merupakan tempat penyimpanan berkas-berkas pekerjaan atau kenangan yang kita miliki di dalam server. Anda bisa menggunakan berbagai layanan penyimpanan berbasis cloud yang sangat populer seperti GitHub. Dengan menggunakan Remote Repository, orang lain dapat mengakses repository yang kita simpan dengan mudah.
- ▶ Saat membuat repository terdapat pengaturan visibilitas yang terdiri dari 2 yaitu private dan public.
- ▶ Private repository merupakan repository yang bersifat tertutup/pribadi dan hanya akun-akun yang telah diberikan akses saja yang bisa melihatnya.
- ▶ Github adalah layanan berbasis git yang berjalan secara online



# Ilustrasi Sebelum ada Git :

---

-  Tugas akhir (Baru Mulai)
-  Tugas akhir (Revisi Ke-1)
-  Tugas akhir (Revisi Ke-2)
-  Tugas akhir (Revisi Ke-3)
-  Tugas akhir (Revisi Ke-4)
-  Tugas akhir (Revisi Ke-5)
-  Tugas akhir (Revisi Ke-6)
-  Tugas akhir (Revisi Ke-7)
-  Tugas akhir (Revisi Ke-8)
-  Tugas akhir (Revisi Ke-9)
-  Tugas akhir (Revisi Ke-10)
-  Tugas akhir (Revisi Ke-11)
-  Tugas akhir (Udah jadi)
-  Tugas akhir Alhamdulillah Lolos
-  Tugas akhir Alhamdulillah Wis Udah



# Ilustrasi setelah menggunakan git :

📄 Tugas akhir (Baru Mulai)  
📄 Tugas akhir (Revisi Ke-1)  
📄 Tugas akhir (Revisi Ke-2)  
📄 Tugas akhir (Revisi Ke-3)  
📄 Tugas akhir (Revisi Ke-4)  
📄 Tugas akhir (Revisi Ke-5)  
📄 Tugas akhir (Revisi Ke-6)  
📄 Tugas akhir (Revisi Ke-7)  
📄 Tugas akhir (Revisi Ke-8)  
📄 Tugas akhir (Revisi Ke-9)  
📄 Tugas akhir (Revisi Ke-10)  
📄 Tugas akhir (Revisi Ke-11)  
📄 Tugas akhir (Udah jadi)  
📄 Tugas akhir Alhamdulillah Lolos  
📄 Tugas akhir Alhamdulillah Wis Udah



● Master  
● Penambahan Isi  
● Menambah teori menurut ahli  
● Merubah abstrak  
● Merubah cover  
● Kata Pengantar  
● Format tabel  
● Daftar Pustaka  
● Dokumenter (Gambar)  
● Standar format halaman  
● Standar format penulisan  
● Penutup  
● Menambah poin anggaran  
● Menambah dokumentasi  
▼ 📄 Tugas akhir

# Jadi apa saja manfaat menggunakan Git?

---

- ▶ - Mencatat riwayat perubahan pada berkas atau proyek
- ▶ - Mengelola berkas pada saat bekerja secara kolaborasi
- ▶ - Mengelola perubahan berkas atau proyek
- ▶ - Bisa berkontribusi pada Proyek Sumber Terbuka
- ▶ - Bisa memperdalam pengembangan aplikasi berbasis CLI ( Command Line Interface )



# Version Control Systems

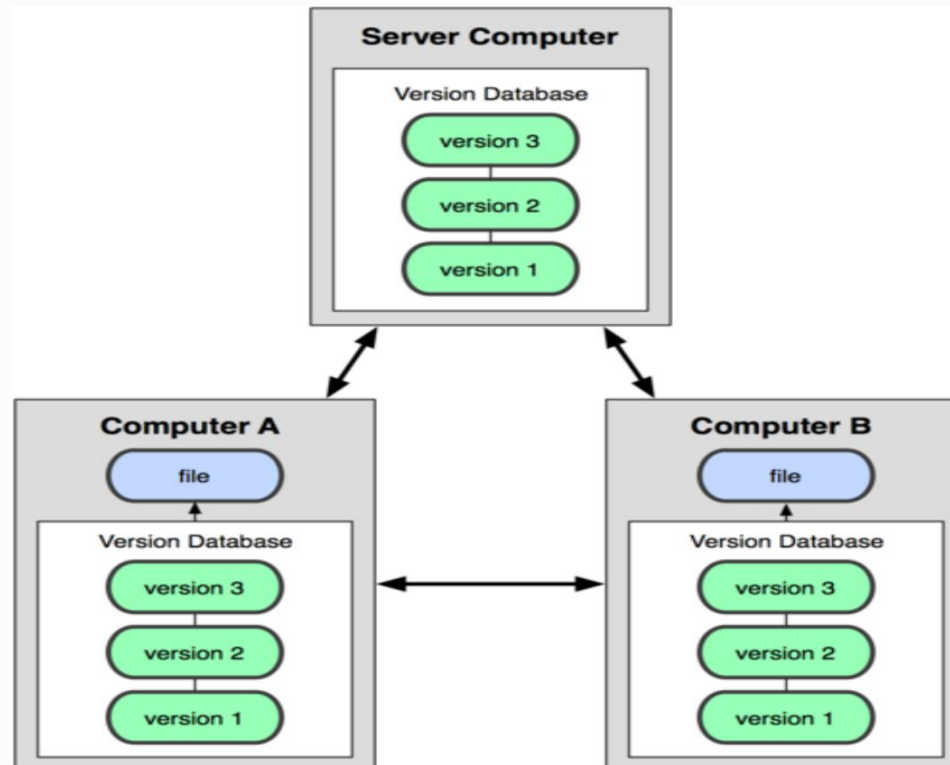
---

- ▶ Also known as Source Code Management systems
- ▶ Increase your productivity by allowing you to
  - ▶ Track every version of your work
  - ▶ Work at multiple locations and synchronize the changes
  - ▶ Coordinate work with multiple developers
  - ▶ Test changes safely in a branch
  - ▶ Undo your mistakes
  - ▶ Backup your work
- ▶ Popular tools - VCS, Git, SVN
  - ▶ Plug-ins available for the major IDEs



# Git

- ▶ Allow developers to create multiple repositories (local or remote) and to synchronize them
- ▶ Written in C and being used by Linux and more ([github.com](https://github.com))





# Key Git Files/Directories

---

- ▶ **~/.gitconfig**
  - ▶ In the user's home directory
  - ▶ Contains user name, email, and other options
- ▶ **.git**
  - ▶ In top level of repository
  - ▶ Contains all objects, commits, configuration, for project
  - ▶ .git/config has project specific configurations
- ▶ **.gitignore**
  - ▶ Stored in directory for ignoring such as files that are automatically generated (build and temporary files)



# Help

---

- ▶ **Git provides three equivalent ways to get help**
  - ▶ `man git-command`
  - ▶ `git help command`
  - ▶ `git command --help`
- ▶ **For example, to learn about `git init`, use**
  - ▶ `man git-init`
  - ▶ `git help init`
  - ▶ `git init --help`



# Basic Workflow for Individual Projects

---

- ▶ Setup your repository
- ▶ Work on your project
- ▶ Stage changes for commit
- ▶ Commit changes
- ▶ Create a branch
- ▶ Switch to a branch
- ▶ Merge changes in different branches
- ▶ Inspect/revert changes
  
- ▶ Commit often



# Getting Started

---

- ▶ `cd ~` // switch to home
- ▶ `mkdir repo`
- ▶ `cd repo`
- ▶ `mkdir datafiles`
- ▶ `touch test1` // create a few files
- ▶ `touch test2`
- ▶ `touch datafiles/data.txt`
- ▶ `ls > test1` // put text in file test1



# Creating a Repository for Your Work

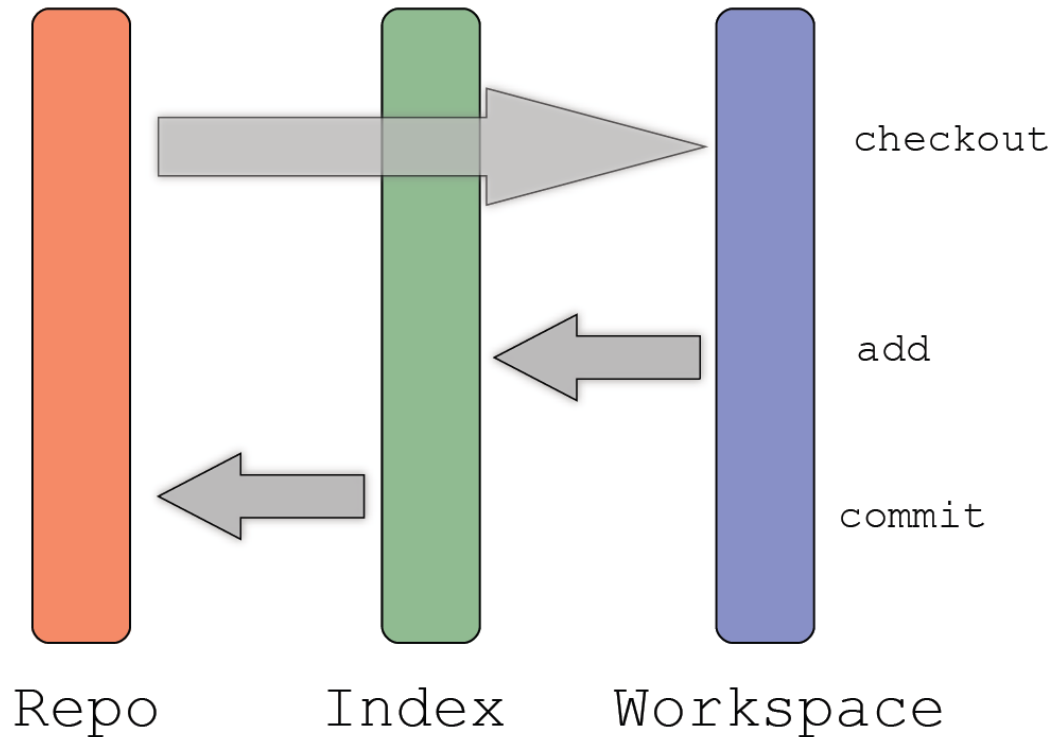
---

- ▶ `git init`
- ▶ `# configure your repository - optional`
- ▶ `touch .gitignore`
- ▶ `echo "*~" >> .gitignore`      `// ignore temporary files`
- ▶ `echo "a.out" >> .gitignore`
- ▶ `echo "*.oa" >> .gitignore`      `// ignore objects/archives`
- ▶ `git add .`
- ▶ `git commit -m "Initial commit"`



# Git Concepts

---



- ▶ Committed changes in the repository are referred to as the history

# Inspecting Your Workspace State

---

- ▶ `echo "make some change" >> test1`
  - ▶ `echo "and more change" >> test2`
  
  - ▶ `# show the state of the working copy`
  - ▶ `git status`
  - ▶ `# find the difference since the last commit`
  - ▶ `git diff`
  
  - ▶ `git add .` `// or git add test1 test2`
  - ▶ `git commit -m "Second commit"`
  
  - ▶ `git log` `// show the history`
- 



# Reverting Changes

---

- ▶ `echo "changes not needed" >> testl`
- ▶ `# checkout the previous version`
- ▶ `git checkout testl`
- ▶ `cat testl`





# Reverting Back to Earlier State

---

- ▶ `echo "add one line here" >> test1`
- ▶ `cat test1` `// show the content of the file`
- ▶ `git add test1`
- ▶ `git commit -m "add one line in test1"`
  
- ▶ `# delete the commit beforehand`
- ▶ `# or ~2 would delete the last two`
- ▶ `git reset --hard HEAD~1`
- ▶ `cat test1` `// check its content`



# Deleting Files

---

- ▶ `echo "useless data" > test3`
- ▶ `git clean -f` `// delete it`
- ▶ `touch test4` `// create a file`
- ▶ `git add test4` `// add it to the index`
- ▶ `# remove it`
- ▶ `git rm test4`



# Branches

---

- ▶ Independent copies of the source code which can be changed independently from each other
  - ▶ Local or remote
  - ▶ The original branch is referred to as master
- ▶ List all local branches (\* for the current one)
  - ▶ `git branch`
- ▶ See all branches (including remote ones)
  - ▶ `git branch -a`

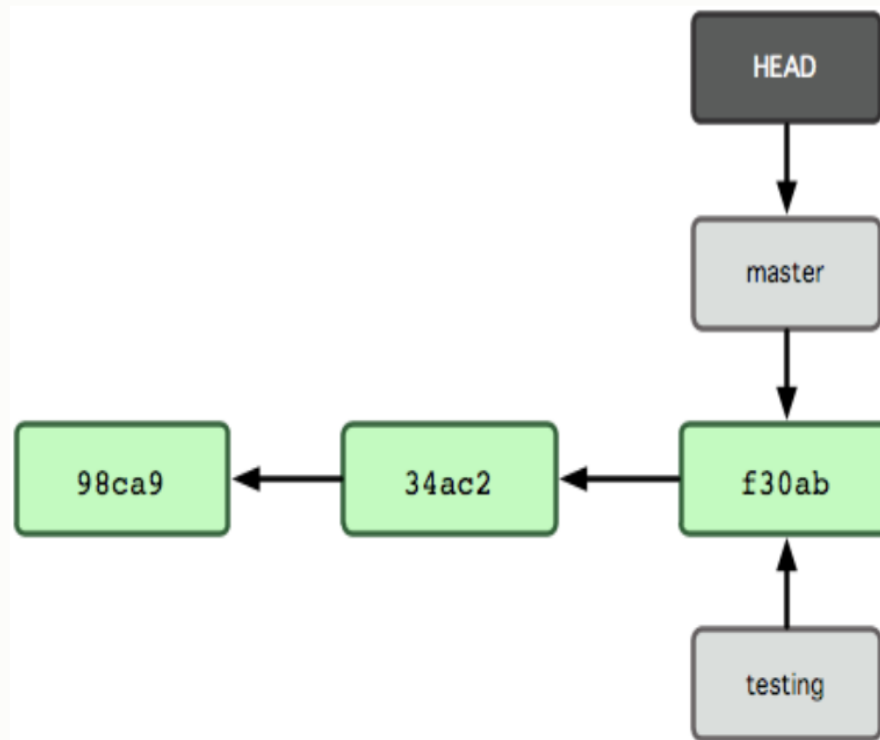


# Creating a New Branch

---

- ▶ `git branch testing`
- ▶ `git branch`

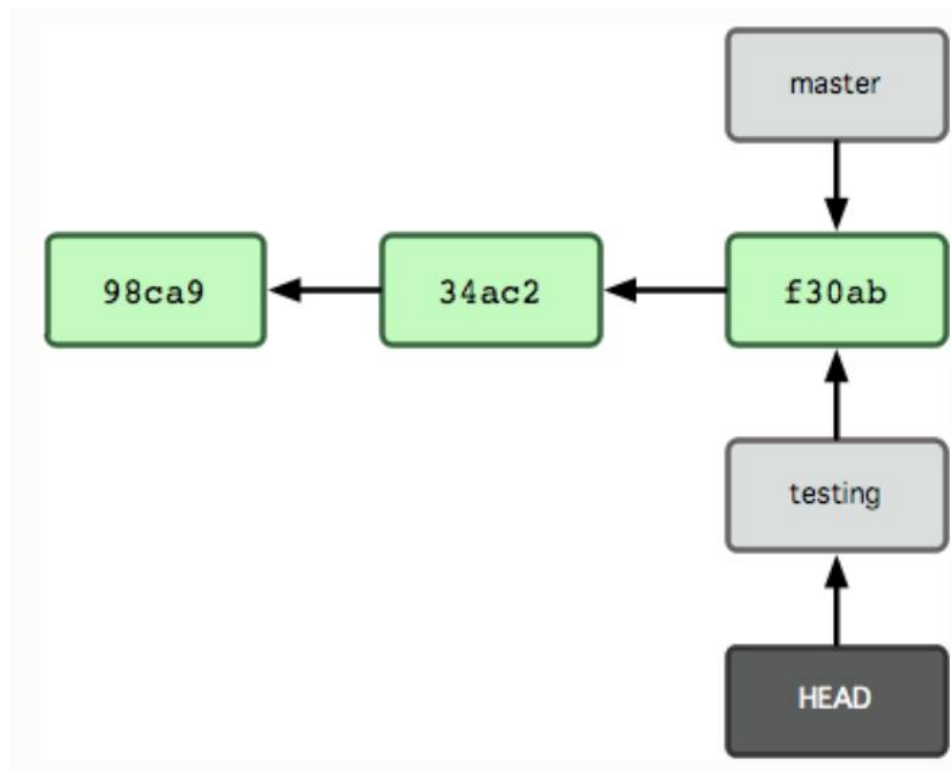
// list your current branches



# Switching to a Branch

---

- ▶ git checkout testing
- ▶ git branch



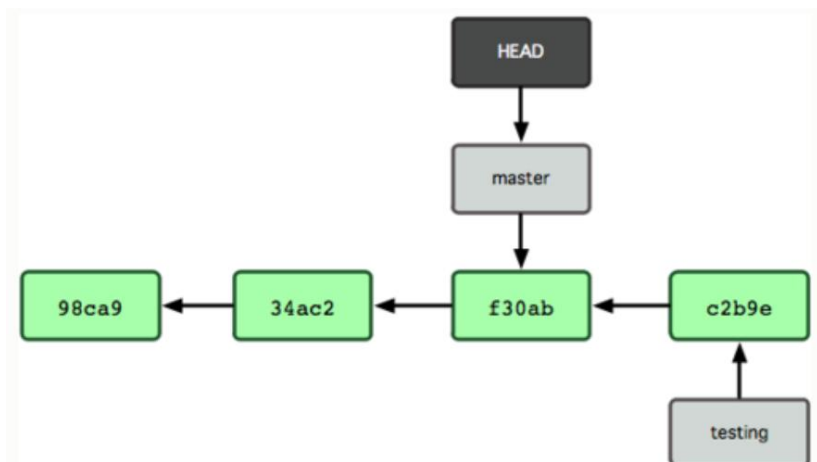
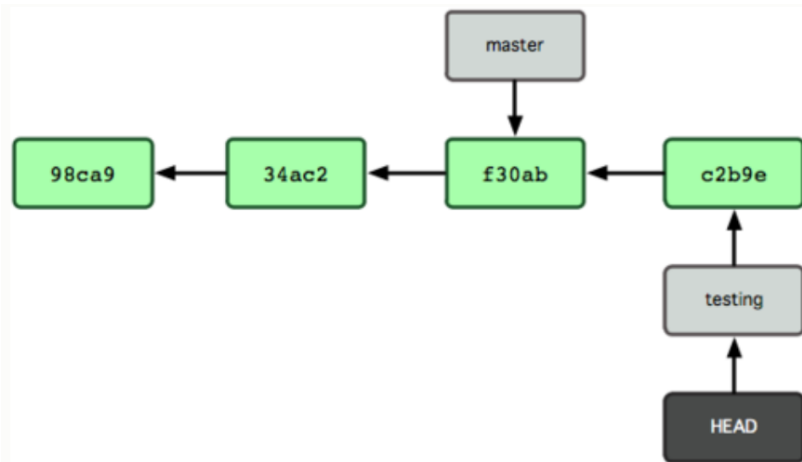
# Working in Branch testing

echo “add new features in testing” >> testI

git commit -a -m “make changes in testing”

git checkout master // switch to the master branch

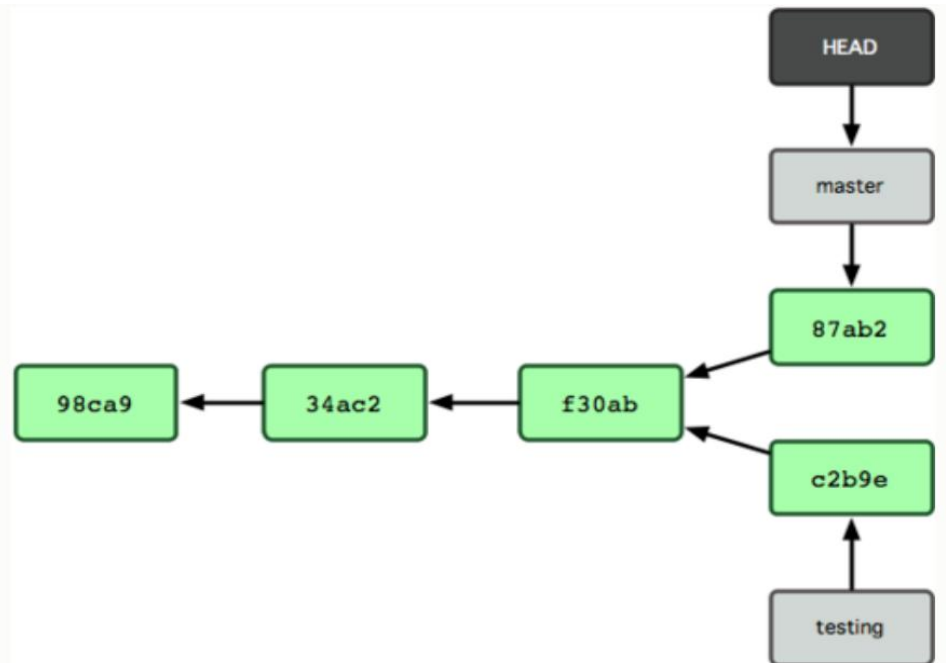
cat testI // see the original content



# Working in Branch master

---

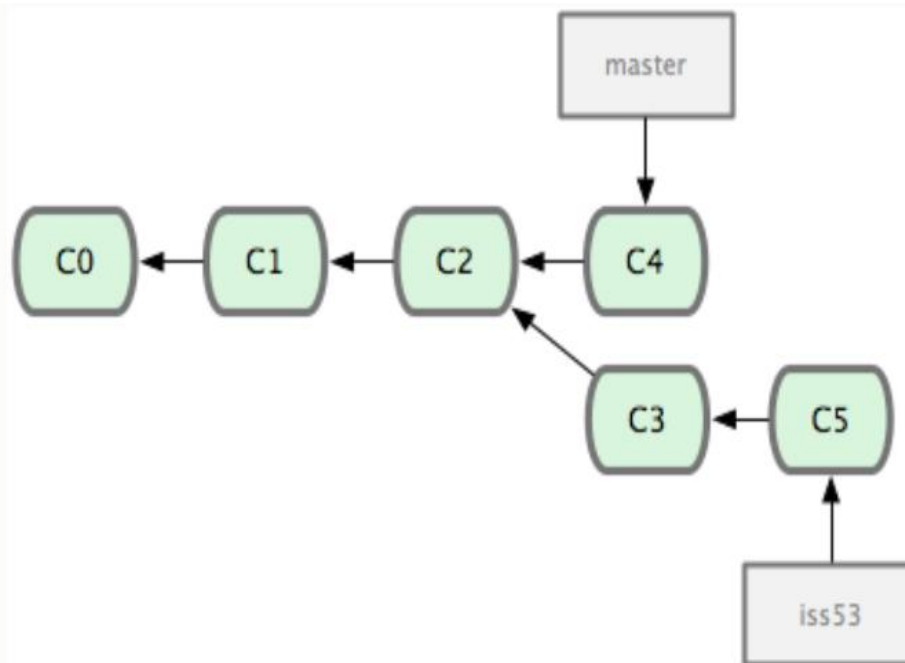
- ▶ # Make changes to testI in master
- ▶ nano testI
- ▶ git commit -a -m "made changes in master"
- ▶ cat testI



# Working in Branch testing

---

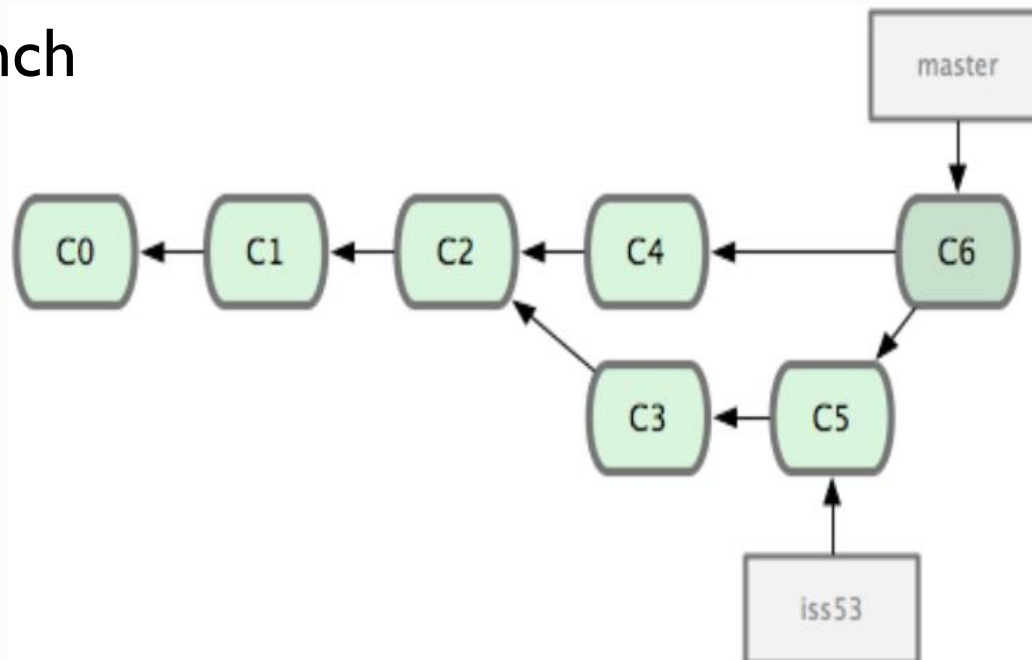
- ▶ git checkout testing
- ▶ nano testI
- ▶ git add .
- ▶ git commit -m “add more changes in testing”





# Merging Changes in Both Branches

- ▶ `git checkout master`
- ▶ `git merge testing`
- ▶ `# delete a branch if no need any more`
- ▶ `git branch -d testing`
- ▶ `git branch`



# Resolving Conflicts

---

- ▶ If you change the same place in a file in two different branches, there will be a conflict when you merge
- ▶ Must resolve manually by editing the file
- ▶ Use `git diff -cc` to verify changes after resolving conflict
- ▶ Finally stage and commit the changes



# Distributed Development

---

- ▶ **Multiple repositories**
  - ▶ Allow to collaborate with others or synchronize your work across multiple computers
- ▶ **Support for working with remote repositories**
  - ▶ Clone a remote repository
  - ▶ Fetch and pulling from a remote
  - ▶ Push to a remote
  - ▶ Inspecting a remote
  - ▶ Manage remote repositories and repository hosting



# Tagging

---

- ▶ Git allows you to tag specific points in history as being important
- ▶ People usually use this feature to mark release points such as v1.0, v1.5, and so on



# Conclusion

---

## ▶ Basic commands

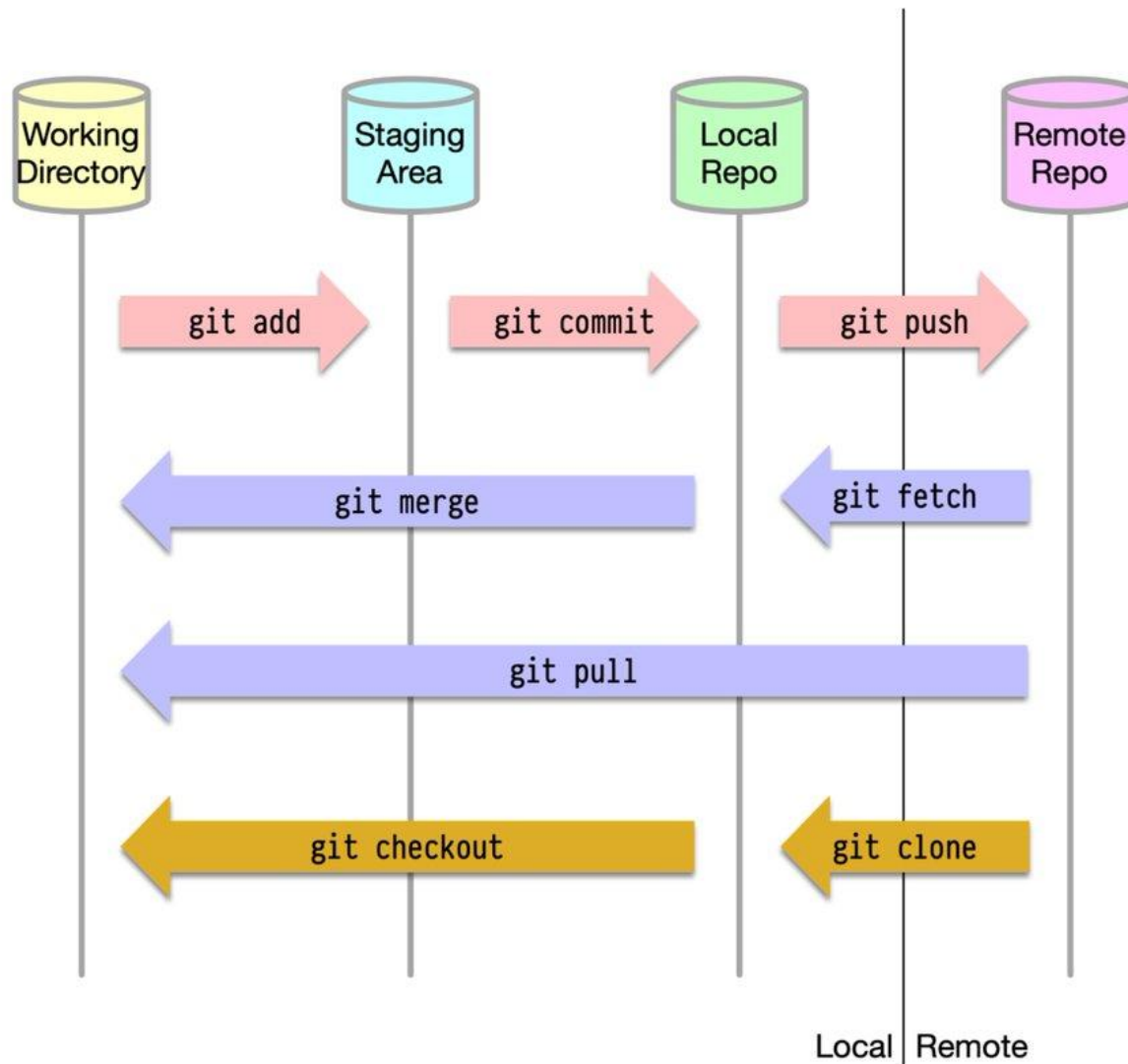
- ▶ git init                      create a repository
- ▶ git add                      stage changes
- ▶ git commit                  store changes to repository
- ▶ git branch                  create a branch
- ▶ git checkout                change workspace to a different branch
- ▶ git merge                    merge two branches

## ▶ Other commands

- ▶ clone, remote, fetch, pull, push, rebase



## How Git Commands work



# References

---

- ▶ <http://www.vogella.com/articles/Git/article.html>
- ▶ <http://git-scm.com>

