How to Git CHARA software development under control

Brian Kloppenborg
Georgia State University
bkloppenborg@gsu.edu

What is a version control system?

Specialized software which keeps track of incremental changes to files

- Examples:
  - Copies of folders on local machine, CVS, SVN, Hg, git

Why use version control systems?

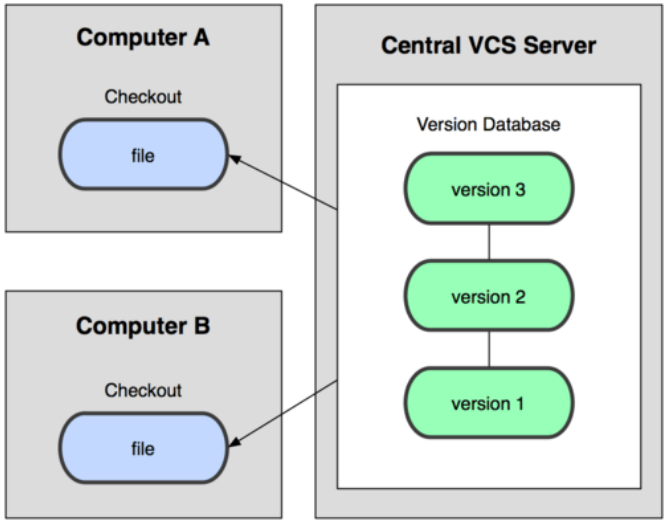
- Have you ever:
  - Made a change to code, realized it was a mistake and wanted to revert back?
  - Lost code or had a backup that was too old?
  - Had to maintain multiple versions of a product?
  - Wanted to see the difference between two (or more) versions of your code?
  - Wanted to prove that a particular change broke or fixed a piece of code?
  - Wanted to review the history of some code?
  - Wanted to submit a change to someone else's code?
  - Wanted to share your code, or let other people work on your code?
  - Wanted to see how much work is being done, and where, when and by whom?
  - Wanted to experiment with a new feature without interfering with working code?

Text taken from http://stackoverflow.com/questions/1408450

Status quo at CHARA

CVS and email

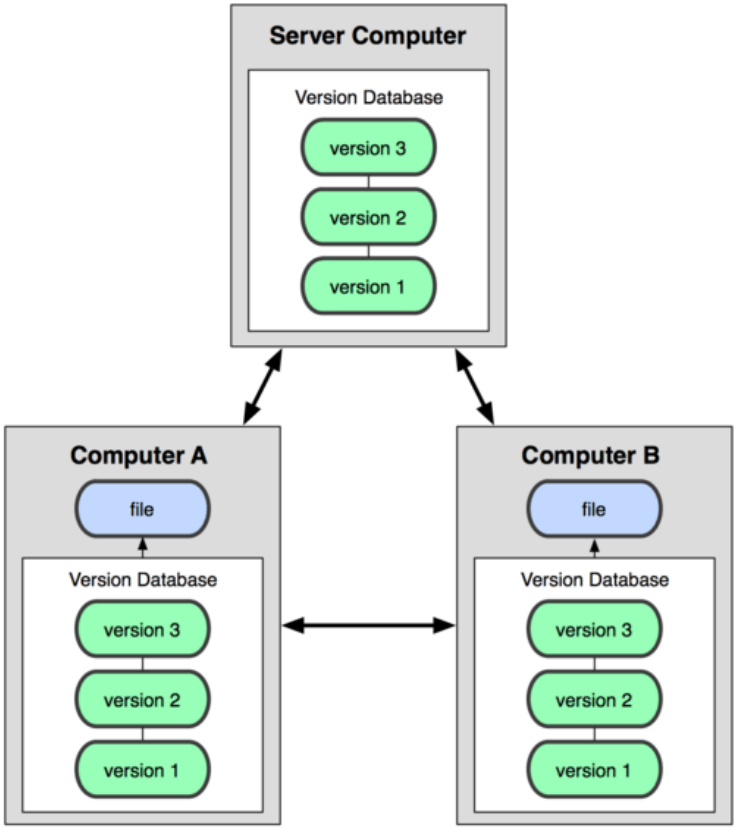
- A single CVS repository
- Bugs/features via. email
- Documentation in man pages and a wiki



Problems:

- Network access required
- Repository not redundant
- Single branch development
- High level of interdependency
- No bug/issue/feature tracker
  - Design decisions hidden in email
  - No clear development plan or time line
- Some documentation difficult to update

git: Distributed version control

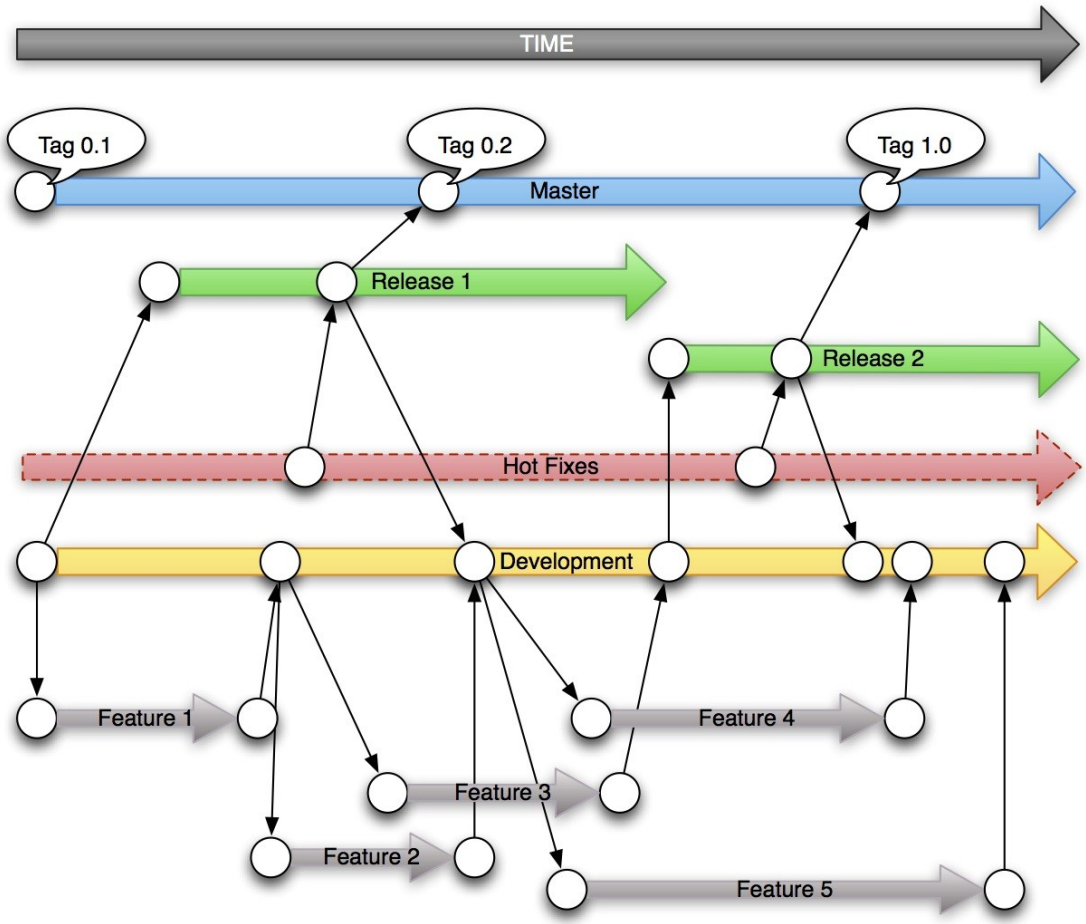


About git: http://git-scm.com/
Intro to git: http://git-scm.com/book

Benefits

- Works offline
- Intelligent merging
- Branches: fast and local
- Find regressions via. bisection
- Easy to share, send patches
- More than one “central” repository
- Name commits via. tags
- Submodules

Branching



Graphic from http://erickryski.com/2012/06/01/my-git-branching-strategy/

git – typical daily usage

- Create initial copy of the repository:

```
git clone user@yourserver.com:thing.git
```

- Switch to a different branch (this is really fast)

```
git checkout [-b] branch_name
```

- Add/stage file(s) and make commits

```
git add test1.cpp test1.h test2.* …
git commit -m “Short commit message”
```

- Once done, push your changes somewhere else

```
git push destination branch_name
```

- Update your local copy

```
git pull
```

Gitlab: Collaborative coding

Migration time line

- Verify CHARA machines run git – done
- Trial CHARA CVS to git conversion – done
- Get gitlab server operational – done

- Configure backups for gitlab server – Friday
- Wait for SSL certificate from IS&T – who knows?
- Put core CHARA libraries on git + gitlab – next week