# Intermediate Git

## Day 1: Understanding Git's Worldview

## Raman A. Shah

# Some initial configuration

```
git config --list
```

If your user name and email are not set:

```
git config --global user.name "Raman A. Shah"
git config --global user.email "raman@uchicago.edu"
```
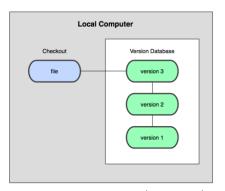
If you don't like `vim` firing up in the middle of doing Git stuff:

```
git config --global core.editor "nano"
```

...a distributed version control system.
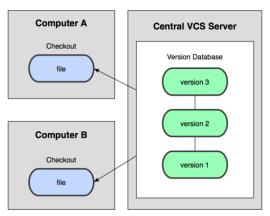
. . . a distributed
version control
system.
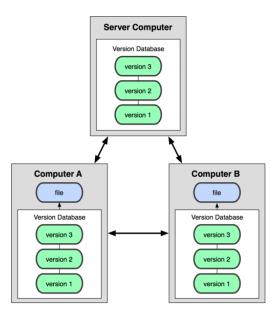
. . . a distributed version control system.

# Git is. . .



Local version control (*e.g.*, rcs).

# Git is...



Centralized version control (*e.g.*, CVS, Subversion (SVN), Perforce).

# Git is...



Distributed version control (*e.g.*, rcs).

. . . a great way to collaborate on projects consisting of many code or text files.

. . . meant for perfecting (software) *products*.

...a content-addressable
filesystem.

# Exploring a Git repository's internals

From a place where you wouldn't mind a new subdirectory:

```
git clone [URL]
```

```
cd [repo name]
```

Explore the contents of `.git` and `.gitignore`. To list a directory's contents including hidden "dotfiles":

```
ls -al
```

To write out the contents of a file to the terminal:

```
cat [filename]
```

. . . safe because it tracks every single bit in your files and commits with hash functions.

...fast because it stores a (compressed) copy of every version of every file locally.

. . . hard because efficiently managing version control and collaboration is hard.

# Playing with the Past

git log

git diff

git show

git checkout

# Reviewing history: git log

# Tracking down changes: git diff

# Seeing old versions: git show

...a great system for archiving the data created in (experimental) projects.

. . . ideal for storing bulky
   data.

. . . quite as helpful for binary
files as for text files.

. . . a golden bullet for collaborating on written works.