Revision Control

An Introduction Using Git

Overview

1. What is revision control?
2. 30,000 foot view
3. Software - git and gitk
4. Setting up your own repository on onyx

What is version control?

- From Wikipedia: Revision control, also known as version control and source control (and an aspect of software configuration management), is the management of changes to documents, computer programs, large web sites, and other collections of information.

What is version control?

- Version control (aka *Revision Control System* or *Source Control System* or *Source Code Management*) is the art and science of managing information, which for software projects implies managing files and directories over time.
- A repository manages all your files and directories. The repository is much like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows you to recover older versions of your data, or examine the history of how your files have changed.

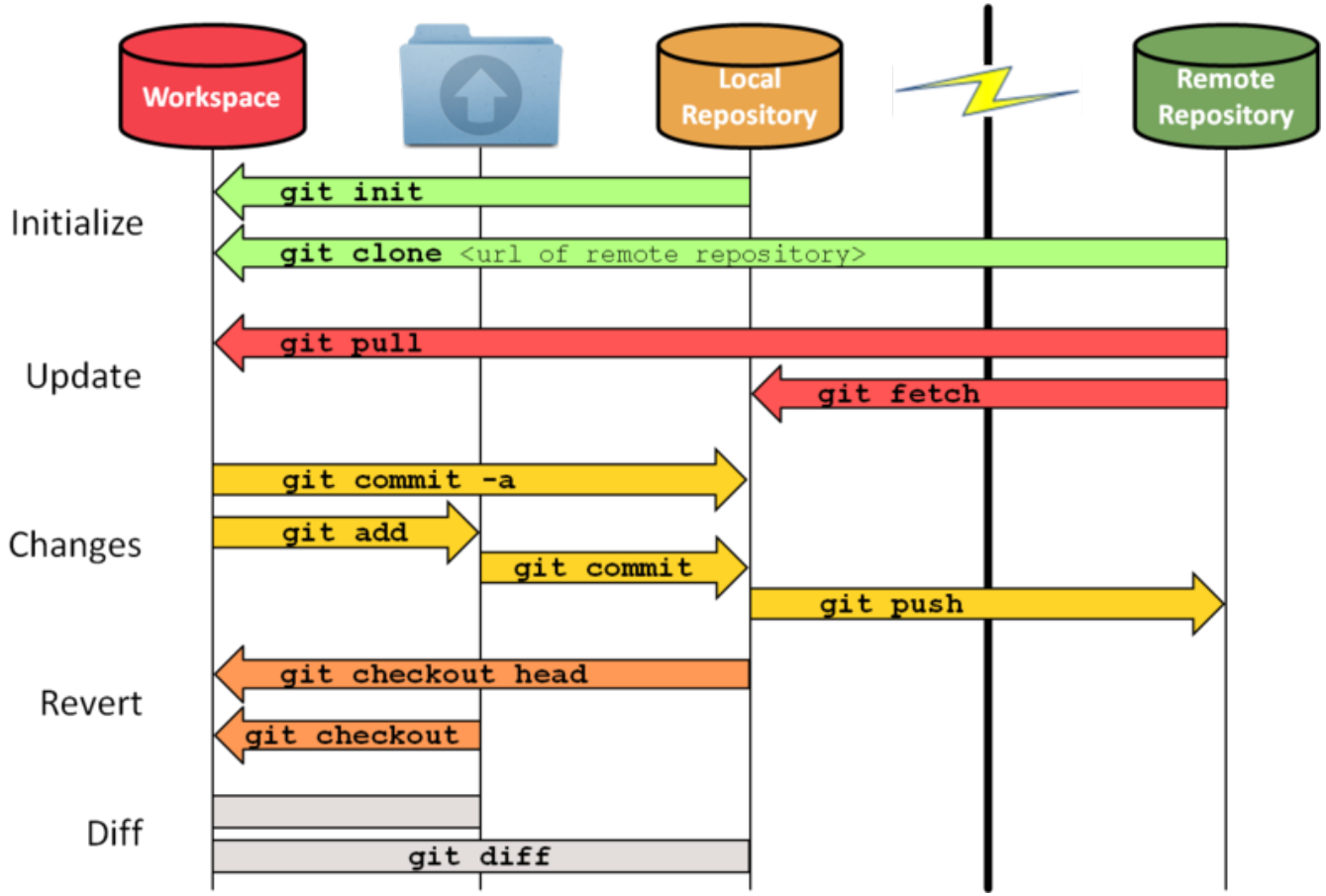
Version control

- Keeps a history of all changes to files under its control.
- Similar to DropBox but with more control and power.
- Similar to Google docs.
- Allows multiple developers to work on the same files at the same time.

Various open-source version control systems

- RCS (Revision Control System). Simple, text based system. Included in Linux and Unix systems by default. No remote access. No directory level access.
- CVS (Concurrent Versioning System). Built on top of RCS. Adds directory level access as well as remote access.
- Subversion. A modern CVS "replacement" that isn't built on top of RCS. Allows directory access, web access (via an Apache Web server module), remote access (via ssh or svn server). Uses a centralized model with mulitple access-control possibilities.
- Git. A distributed version control system. There is no central repository like in subversion. Everyone has a copy of the repository.

Git Workflow



Git Setup

1. Download git client.
2. Git hosting options.
3. Create a new repository.
4. Checkout (clone) your repository.
5. Work on your local repository.
   - Add and commit changes.
   - Push changes.
6. Update local repository.

Download Git Client

- For latest download information, see `http://git-scm.com/downloads`.
  - Linux - install using your package manager
  - Windows - install from git-scm.com
  - MacOS - included with XCode install

Add, Commit, and Push Changes (1)

Assume we want to modify the README that we created when we initialized our repository.

- We would make the changes.
- Then, stage the changes for commit. By staging, we are telling git to include the file in the index for the next commit. To stage a file, we use add.

```
git add README
```

- After staging, we want to commit the changes.

```
git commit -m "Updated README file with some text"
```

- You can also add and commit in one step.

```
git commit -a -m "Updated README file with some text
```

- But this is just a commit to our local repository. We still need to push the committed changes to the master/central repository.

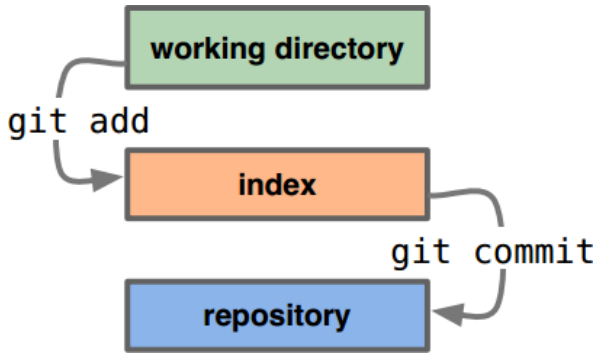
```
git push origin master
```

- The full process is demonstrated on the next slide.

Add, Commit, and Push Changes (2)

- Make the changes. Add, commit and push changes.

```
[spanter@event_horizon cs253(master)]$ gvim README
[spanter@event_horizon cs253(master)]$ git commit -a -m "Updated README file with some text"
[master 38be3cc] Updated README file with some text
 1 file changed, 3 insertions(+)
[spanter@event_horizon cs253(master)]$ git push origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 364 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To spanter@onyx.boisestate.edu:git/cs253.git
   e11edb0..38be3cc  master -> master
[spanter@event_horizon cs253(master)]$
```



Summary of Basic Commands

- Clone your repository
  - `git clone`
- Pull any changes from the master branch
  - `git pull origin master`
- Make changes
  - `edit files`
  - `git add`
- Examine your changes
  - `git status`
  - `git diff`
  - `gitk`
- Get information
  - `git gui`
  - `git log`
- Commit your changes
  - `git commit -m`
- Pushing your changes
  - `git push origin master`

Tools

- gitk - this is a great tool to visualize your repository
- git gui - gives a gui interface into git
- tortoisegit - Windows only integration of git into the explorer shell
- git bash script - adds useful git status information to your bash prompt.
  - You can clone the tool repository from `https://github.com/BoiseState/tools`

Some advanced features

We will not be using these features in this class but they are what make git shine in industry

- Branching is painless and fast in git
- Cherry pick - pulling specific commits to/from branches
- Rebase - take all the changes from one branch and replay them on another
- squash merge - combine several commits into one commit (helps correlate features/bug requests to actual code)

References

1. http://en.wikipedia.org/wiki/Revision_control
2. http://git-scm.com
3. http://rogerdudler.github.io/git-guide/