Version control

Paul Fodor
CSE316: Fundamentals of Software Development
Stony Brook University

http://www.cs.stonybrook.edu/~cse316
Never Lose Your Code Again

• Have you ever been working on a project and:
  • your computer died?
  • you made changes to something working and now it’s not working?
  • you wanted to combine your work with a teammate’s work?

• The solution?
  • Version Control (aka. revision control systems, source control, or source code management)
Software versioning and revision control systems

- Version control (also known as revision control, source control, and source code management, VCS - version control system) is the management of changes to documents, computer programs, large web sites, and other collections of information.

- A system for managing changes to files

- Used by individuals and teams to keep:
  - History of changes
  - Share and distribute common source code
Software versioning and revision control systems

- Large product development needs some capabilities to control development
  - Track changes to source including: Date changed, version #, developer, comments
  - Ability to ‘revert’ changes
  - Ability to mark all sources at a specific version to produce a release/distribution
There are many tools out there

CVS

Subversion

git

Visual Studio Team Foundation Server

Perforce

mercurial

beanstalk

(c) Paul Fodor (CS Stony Brook)
Version Control System Services

- Backup and Restore
- Synchronization
- Short-term undo
- Long-term undo
- Track Changes
- Track Ownership
- Sandboxing
- Branching and merging
Backup and Restore

- Files are saved and committed at versioning steps
- One can jump to any moment in time
- Need that file as it was on August 23, 2020?
  - no problem, just ask the VCS for it
Synchronization

• Lets developers:
  • share files
  • stay up-to-date with the latest version
• Even while developers are working simultaneously.
Short-term Undo

• Editing a file and messed it up?
• Throw away your changes and go back to the “last known good” version in the database
Long-term Undo

- For particularly bad mistakes
- Suppose you made a change a year ago, and it had a bug
- Jump back to the old version, and see what change was made that day
Track Changes

- As files are updated, you can leave messages explaining why the change happened
- stored in the VCS log, not the file
- you can query it
- This makes it easy to see how a file is evolving over time, and why
- Developers should document every change
Track Ownership

- A VCS tags every change with:
  - the name of the person who made it
  - date/time of change
- Helpful for blamestorming
Sandboxing/branching

• Making a big change?
  • You need an insurance against yourself

• You can make temporary changes in an separate isolated area
  • test and work out the kinks before “checking in/merging” your changes to the main branch
Branching and Merging

• A larger sandbox
• You can branch a copy of your code into a separate area and modify it in isolation
  • tracking changes separately
• Later, you can merge your work back into the common area.
Terms

• Repository (repo): The database storing the files.
  • May also mean the Repository Server: The computer storing the repository

• Client: The computer connecting to the repository

• Working Set/Working Copy: Your local directory of files, where you make changes.

• Trunk/Main: The “primary” location for code in the repository
Terms

- Changelog/History: A list of changes made to files since it was created
- Revision: What version a file is on (v1, v2, etc.)
- Head: The latest revision in the repository
VCS Basic Actions

- **init**: initializes a new repository.
  - If you want to place a project under revision control, this is the first command you need to learn.
- **Check out**: Download the code from the repository server for the first time.
VCS Basic Actions

• Add: Put a file into the repository for the first time, i.e. begin tracking it with Version Control

• Check in: Upload files to the repository (if they have changed).
  • the files get new revision numbers, and people can “check out” the latest one
Basic Actions

- Update/Sync: Synchronize your files with the latest from the repository
  - this lets you grab the latest revisions of all files
- Revert: Throw away your local changes and reload the latest version from the repository
Advanced Actions

- Branch: Create a separate copy of a file/folder for private use (bug fixing, testing, etc)
  - Branch is both a verb ("branch the code") and a noun ("Which branch is it in?"

- Diff/Change/Delta: Finding the differences between two files
  - useful for seeing what changed between revisions.
Advanced Actions

• Merge (or patch): Apply the changes from one file to another, to bring it up-to-date
  • For example, you can merge features from one branch into another

• Conflict: When pending changes to a file contradict each other
  • both changes cannot be applied

• Resolve: Fixing the changes that contradict each other and checking in the correct version
Advanced Actions

- Locking: “Taking control” of a file so nobody else can edit it until you unlock it.
  - some VCSs use this to avoid conflicts.
- Breaking the lock: Forcibly unlocking a file so you can edit it.
  - may be needed if someone locks a file and leaves
- Check out for edit: Checking out an “editable” version of a file
  - some VCSes have editable files by default, others require an explicit command.
Types of VCSs

- Major models used in various products
  - Centralized server based version control
    - File Locking
    - Version merging
  - Distributed Version Control
    - When you check out the code from a repository, you create a local repository
      - Allows many developers to work on a given project without requiring that they maintain a connection to a common network.
VCSs

- Revision Control System (RCS)
  - dead as a stand-alone system
- Concurrent Versioning System (CVS)
  - dying
- Subversion (SVN)
  - killing CVS
  - open source under the Apache license
  - http://subversion.apache.org
- Distributed/decentralized revision control:
  - Git
  - Mercurial
- GNU Bazaar
- BitKeeper
Apache Subversion (SVN)

- Developed by the Apache Software Foundation
- Distributed under Apache License (an open source license)
- Used by:
  - Apache Software Foundation
  - FreeBSD
  - GCC
  - Mono
  - SourceForge
- Server-client model: Native SVN server or Apache HTTP Server.
SVN Common operations

• Import: is the act of copying a local directory tree (that is not currently a working copy) into the repository for the first time.

• Checkout: is to create a local working copy from the repository. A user may specify a specific revision or obtain the latest.

• Commit (check in or ci): is to write or merge the changes made in the working copy back to the repository.

• Update (or sync): merges changes made in the repository (by other people or by the same person on another machine) into the local working copy.

• Merge: is an operation in which two sets of changes are applied to a file or set of files: updates or syncs the user working copy with changes made and checked into the repository by other users + check in files + incorporate branches into a unified trunk.
Apache Subversion

• How to run SVN?
  • Command line: `svn` executable
    `svn commit a.txt`
    `svn update`
  • SVN Clients: TortoiseSVN, Netbeans SVN plugin, Eclipse Subclipse, etc.
Distributed Version Control

- There are multiple copies of the repository
  - Not one ‘cannonical’ reference copy of the repository
  - Multiple repos are each a ‘working copy’ on some peer system

Advantages
  - Common Operations (commit, view history, revert changes, etc) are fast since they happen on a local ‘owned’ repo.
  - Each peer repo acts as a ‘remote backup’ of other repos (protects against data loss)

Drawbacks
  - Complex merge operations (usually can be entirely automated)
  - Merges with conflicts must be resolved by hand
Git:

- Distributed (Decentralized) Version Control System
  - Has a local repository but can also ‘push’ the repo to a remote server.
  - Each developer works on their own copy of the repository
  - Can continue to work even if offline

- GNU license

- Released in 2005 (original author Linus Torvalds - also created Linux)

- Used by Linux kernel

- Free download: [http://git-scm.com](http://git-scm.com)


- Repositories servers: [https://github.com](https://github.com), [https://bitbucket.org](https://bitbucket.org) (private repos., Multi-factor authentication), GitLab
Installing Git

- Navigate to: http://git-scm.com/download/

- Click the link ‘Click here to download manually’
Installing Git on Mac OSX

- Double click on the package (.exe, .pkg) file
Installing Git on Mac OSX

- Set name and email in global config
  git config --global user.name 'John Public'
  git config --global user.email 'john.q.public@mymail.com'
- Verify settings are correct
  git config --global --list
Common operations

- Setting Up a Git Repository:
  - `git init`: initializes a new Git repository.
    - If you want to place a project under revision control, this is the first command you need to learn.
  - `git clone location`: creates a copy of an existing Git repository.
    - Cloning is the most common way for developers to obtain a working copy of a central repository.
    - Example: `git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/`
  - `git add file`: moves changes from the working directory to the staging area.
  - `git commit`: takes the staged snapshot and commits it to the project history.
  - `git pull`: downloads a branch from a remote repository, then immediately merges it into the current branch.
  - `git push`: move a local branch to another repository.
Architecture

- Three areas:
  - Working area
  - Staging Area
  - Repository
Workflow

• Typical workflow
  • Pull updates from remote repository
  • Edit files
  • Add files to staging area (git add)
  • Commit changes to local repo.
  • Push committed changes to remote repository

• If you create a branch:
  • Merge branch back to master
  • Delete branch (optional)
  • Push changes
git status

• This command will show status of a repo
  • Changes files
  • Untracked files
ToDo: try to use GitHub

- Owned by Microsoft

- Make an account

- Download Git
  - [https://git-scm.com/downloads](https://git-scm.com/downloads)

- We’ll use GitBash
The GitBash Command Line

- Try these things:
  - `pwd`
  - `ls`
  - `ls -l`
  - `mkdir hw4`
  - `cd hw4`
git init

- Creates a local git repository
- Puts empty repo in .git
- Don’t ever touch anything in .git
Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner *

Repository name *

pfodor / hw4

Great repository names are short and memorable. Need inspiration? How about probable-umbrella?

Description (optional)

- **Public**
  Anyone on the internet can see this repository. You choose who can commit.

- **Private**
  You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you’re importing an existing repository.

- **Add a README file**
  This is where you can write a long description for your project. Learn more.

- **Add .gitignore**
  Choose which files not to track from a list of templates. Learn more.
  .gitignore template: None

- **Choose a license**
  A license tells others what they can and can’t do with your code. Learn more.
  License: Apache License 2.0

This will set master as the default branch. Change the default name in your settings.

Create repository
Once your Online Repo is Created

- Make sure you are logged into GitHub (like in Chrome)
- Connect your local repo to your remote one:
  `git remote add origin https://github.com/pfodor/hw4`
- `remote add` – tells git to specify the remote repository
  - `origin` – name you are giving to the remote repository
- Now your local repo can push and pull to and from your GitHub repo
First, pull

- You will want all the files in the repository
  git pull origin master
- master? What’s that?
  - a branch
- What’s the branch?
  - i.e. a version
  - a repo can have multiple branches
Add any new files

- Easiest way, just add everything
  
  `git add .`

- Adds files to the repo

- Their changes can now be tracked

- Some files you don’t want in your repo
  
  `.gitignore`
Commit Changes

`git commit -m "this is my first commit"

• Don’t forget the comments`
Push your changes to the remote repo

git push origin master

- If you are working alone, no problem
- If not, be careful, you may have to pull others’ changes first
Merge & Revert

- merge – approve all changes in VSC
- revert – to previous version or date
Summary

- Git is one of the most popular source control systems
- It is a distributed system rather than a centralized source control system
- Git contains a working directory, staging area, and repository to help manage committing changes in a very controlled manner