

# CS 4320 / 7320

# Software Engineering

## Version Control Systems

# Topics

- Version Control Systems
  - What are they
  - Types
- Terminologies
- Collaborative Development through VCS
- Introduction GIT

# Version Control Systems

## Centralized

- Work directly against a central server
- Subversion, CVS, Perforce, etc.

## Distributed

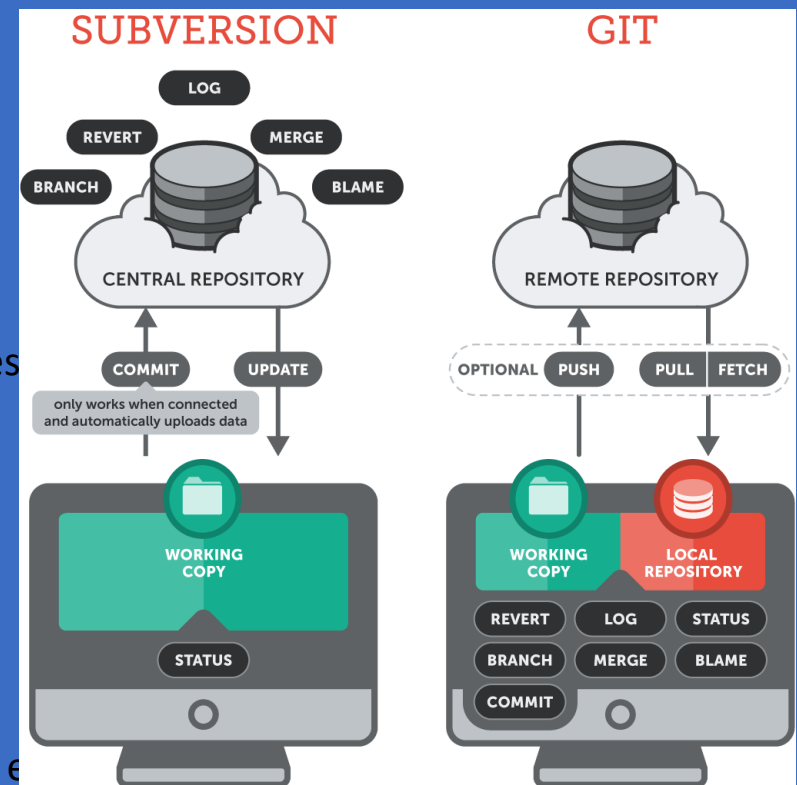
- Work locally, optionally push to remote repositories
- Git, Mercurial (Hg), etc.

## Hosted solutions

- GitHub, BitBucket, Visual Studio Online, etc.

## On-Premise solutions

- GitHub Enterprise, Stash, Team Foundation Server, etc.



# Version Control System

- Track changes and revisions to both files and file system structure of working environment
  - File Additions / Deletes
  - Folder Additions / Deletes
  - File Edits
- aka
  - Source control system
  - Source code control system
  - Revision control system

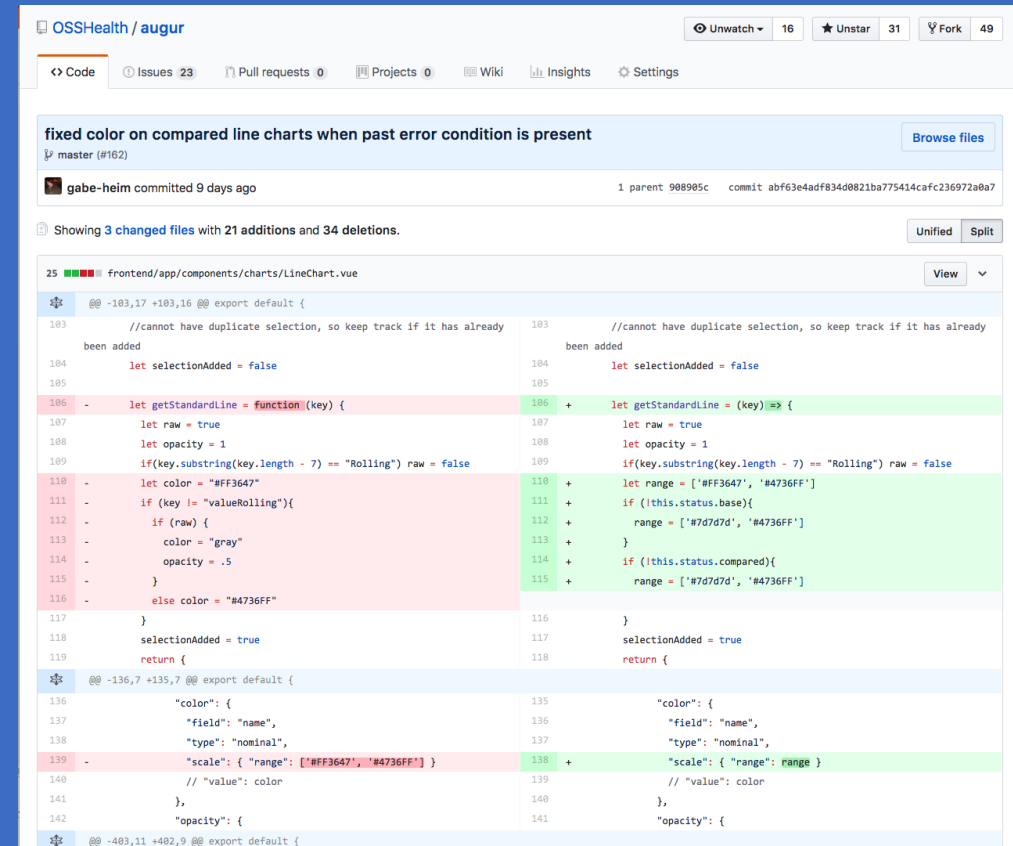
# What is Version Control?

## System to manage changes to files

- Who
  - When
  - What
  - (Sometimes) Why
- Sean Goggins  
January 3, 2012  
Files and differential  
Fix Color ...

## Why use it?

- Revert to or review prior changes
- Maintain multiple versions
- Compare differences
- Share and collaborate
- Modern solutions might assume you have it!
  - Infrastructure-as-code
  - Continuous Integration and Delivery
- Google around for many more reasons!



The screenshot shows a GitHub pull request interface for the repository OSSHealth/augur. The pull request is titled "fixed color on compared line charts when past error condition is present" and is authored by gabe-heim, committed 9 days ago. It shows 3 changed files with 21 additions and 34 deletions. The diff view is displayed for the file frontend/app/components/charts/LineChart.vue. The diff shows changes to the getStandardLine function, which sets the color and range of a line based on its status. The changes include adding a range property to the color object and updating the range value based on the line's status.

```
@@ -103,17 +103,16 @@ export default {
  //cannot have duplicate selection, so keep track if it has already
  been added
  let selectionAdded = false
  let raw = true
  let opacity = 1
  if(key.substring(key.length - 7) == "Rolling") raw = false
  let color = "#FF3647"
  if (key != "valueRolling"){
    if (raw) {
      color = "gray"
      opacity = .5
    }
    else color = "#4736FF"
  }
  selectionAdded = true
  return {
    "color": {
      "field": "name",
      "type": "nominal",
      "scale": { "range": ["#FF3647", "#4736FF"] }
    },
    // "value": color
  },
  "opacity": {
    "field": "name",
    "type": "nominal",
    "scale": { "range": range }
  },
  // "value": color
  "opacity": {
```

AKA Revision Control, Source Control

# Change Tracking

- Files (and folders) exist in a temporal condition
- File A:
  - T-1 has 100 lines
  - T-101 has 1000 lines
  - How was the file changed over time from T-1 to T-101 ?
- VCS can answer this question easily
- Can also roll-back to point in time, e.g., T-99

# VCS Types

- VCS come in various flavors that roughly align to three types of systems:
  - Local
  - Client-Server
  - Networked / Distributed

# VCS Types : Local

- Source Code Control System (SCCS)
  - Developed at Bell Labs, 70's
  - Critical early stage use in development of UNIX
  - Part of the *Single UNIX Specification*
- File are locally version controlled
- Collaboration is limited to a single system
- Some VCS still use internals



# VCS Types : Client-Server

- Client programs read and write changes to a development tree that exist on a server
  - Multiple developers can pull down to push up changes
- Concurrent Versions System (CVS)
  - Or: Concurrent Versioning System
  - 80's
- Subversion (SVN)
  - One of most popular today
  - Early 2000s

# VCS Types : Distributed

- Decentralized VCS,
  - Built from the concepts of peer-to-peer trust
  - Each user has full repository in local storage
- GIT is most common
  - Developed by Linus Torvolds specifically for Linux Kernel development
  - 2005
  - GIT and other distributed VCS becoming the most popular VCS, close to SVN in usage

# VCS Advantages

- Provides a control and tracking method to collaborative software development
  - Concepts can be applied to documents, e.g., non-software
- Changes (revisions)
  - Tracked by numerical or hash id
  - Timestamped
  - User stamped

# Hash ID's

The screenshot displays the GitHub interface for the repository `OSSHealth / augur`. At the top, navigation links include `<> Code`, `Issues 23`, `Pull requests 0`, `Projects 0`, `Wiki`, `Insights`, and `Settings`. On the right, there are buttons for `Unwatch 16`, `Unstar 31`, and `Fork 49`. Below the navigation bar, a dropdown menu shows the current branch as `master`. The commit history is organized by date, with sections for `Commits on Aug 28, 2018` and `Commits on Aug 27, 2018`.

**Commits on Aug 28, 2018**

- Merge pull request #162 from OSSHealth/dev** (Verified) `9d15adc` `<>`  
egoggins committed 7 days ago ✓

**Commits on Aug 27, 2018**

- Update version to 0.7.0** `bb5ecd8` `<>`  
howderek committed 8 days ago ✓
- Remove frontend tests from Travis until completed** `1b507e8` `<>`  
howderek committed 8 days ago ✓
- fix pandas key error** `b895295` `<>`  
ccarterlandis committed 8 days ago ✗
- Remove PublicWWW tests** `c75ed9f` `<>`  
howderek committed 8 days ago ✗
- Merge branch 'master' into dev** `cbea207` `<>`  
howderek committed 8 days ago ✗
- color changes** `fb4c8e4` `<>`  
gabe-heim committed 8 days ago ✗
- fixed frontend formatting: bubblechart width, smaller triangle for dl...** `b9be898` `<>`  
gabe-heim committed 8 days ago ✗
- Fix forkserver** `de6c2b4` `<>`  
howderek committed 8 days ago ✗
- updated chart legend names** `49f3641` `<>`  
ccarterlandis committed 8 days ago ✗
- fixed browser-specific bug that cuts off linechart title text** `f96aec5` `<>`  
gabe-heim committed 8 days ago ✗

# VCS Advantages

- Version control is important for development groups to function effectively
- Also utilized for non-software development
  - Word-processing
  - Configuration files
  - Content management systems
  - Database records

# VCS Terminology

- Trunk / Main / Master
  - the primary development branch
  - often the receiver of changes from other branches that are used for small development efforts, e.g., bug-fixes
- Branching
  - Duplication of a folder structure for the purpose of isolating development work from the Trunk

# Branch Demo

# VCS Terminology

- Merge
  - Reconciling multiple changes to a version controlled resource
  - e.g., two versions of a file, the end result is one file with both sets of changes
- Fork
  - A branch that is not intended to be later merged
    - **\*\*NOTE**, on GitHub, Forks are merged back using Pull Requests. More on this Later.
-



# Merge and Fork Demos

# VCS Terminology

- Tag
  - A read-only branch that serves as the end-point of a development effort / interval, e.g., a release
  - Captures a branch at a point in time
  - Labels the point in time
- Commit
  - Saving a change to live files into the repository's set of known edits, i.e., revisions

# Tag Demo

# VCS Terminology

- Baseline
  - The starting point of a branch
- Delta / Diff
  - A revision to one or more files or the file system (tree)
- Conflict
  - Two or more users have changed the same version controlled resource in a manner that cannot be automatically resolved by the VCS

# VCS Terminology

- Head
  - The most recent / current / up-to-date version of a branch
- Update / Pull
  - Pulling in changes from other developers
- Working copy
  - The local working copy of files from the repository

# GitHub

## Global setup:

### Set up git

```
git config --global user.name "Rich Jones"  
git config --global user.email rich@anomos.info
```

## Next steps:

```
mkdir TestRepository  
cd TestRepository  
git init  
touch README  
git add README  
git commit -m 'first commit'  
git remote add origin git@github.com:Miserlou/TestRepository.git  
git push -u origin master
```

## Existing Git Repo?

```
cd existing_git_repo  
git remote add origin git@github.com:Miserlou/TestRepository.git  
git push -u origin master
```

# GitHub Now!



**GitHub**



**git**

# Goals

Use version control for your own projects

Background to make a case to bring version control to your team at \$Work



# Further Reading

There's a lot out there. Here are some highlights. Don't be intimidated, you don't need all this, but it may come in handy if you want to dive a bit deeper into the weeds. The key is starting to use it, just like PowerShell!

## Interactive guides

- GitHub's interactive guide – Learn Git in 15 minutes: <https://try.github.io/>
- Learn Git Branching: <http://pcottle.github.io/learnGitBranching>
- Interactive Cheat sheet: <http://ndpsoftware.com/git-cheatsheet.html>

## References

- Official git reference: <http://git-scm.com/docs>
- Pro Git (free!): <http://www.git-scm.com/book/en/v2> - the first two or three chapters are a great intro
- Understanding Branches: <http://blog.thoughttram.io/git/rebase-book/2015/02/10/understanding-branches-in-git.html>
- Git Explained: For Beginners: <http://www.dotnetcodegeeks.com/2015/06/git-explained-for-beginners.html>
- GitHub Flow – GitHub from a Browser: <https://github.com/blog/1557-github-flow-in-the-browser>

## Contributing to Microsoft's DSC Resources on GitHub

- Guide to getting started with GitHub: <https://github.com/PowerShell/DscResources/blob/master/GettingStartedWithGitHub.md>
- DSC Contributions guide: <https://github.com/PowerShell/DscResources/blob/master/CONTRIBUTING.md>
- DSC Resource Style Guidelines: <https://github.com/PowerShell/DscResources/blob/master/StyleGuidelines.md>

## More references

- Pro Git (free!): <http://www.git-scm.com/book/en/v2> - the rest of the book : )
- A Visual Git Reference: <http://marklodato.github.io/visual-git-guide/index-en.html>
- Git From the Inside Out: <https://codewords.recurse.com/issues/two/git-from-the-inside-out>
- Git For Computer Scientists: <http://eagain.net/articles/git-for-computer-scientists> - Great read with pictures, don't be intimidated by the title
- Branching, forking, other concepts explained: <http://stackoverflow.com/questions/3329943/git-branch-fork-fetch-merge-rebase-and-clone-what-are-the-differences>