hackMGM

Using Git & GitHub, *Aug 17, 2017*

Jacqueline "Jacquie" McKinney

- Tuskegee Graduate – BS in Computer Science
- WPI Graduate – MS in Systems Engineering
- Software Engineer/13 years
- Worked at Xerox, CACI, Raytheon, GDIT (present)
- Worked with C, C++, CORBA, PHP, Perl, Python, HTML/CSS, JS, bash
  - Not so much with Java, TCL/Tk, MS Visual C++, MS COMM
- Linux/UNIX
- From Montgomery, AL
- Soul Searching
- Learning Django and WordPress frameworks

Agenda

- What is Git
- What is GitHub
- Working in GitHutb - Markdown
- Working with Git
- Working in GitHub – Issues
- Git Workflow
- Exercise

Handouts

- git cheat sheet
  - http://rogerdudler.github.io/git-guide/files/git_cheat_sheet.pdf
- Mastering Markdown
  - https://guides.github.com/features/mastering-markdown/
- Hello World
  - https://guides.github.com/activities/hello-world/
- Forking a Repo
  - https://guides.github.com/activities/forking/

What is Git?

Configuration Management

Git

- By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source.
- Has a distributed architecture, called DVCS (distributed version control system)
- Every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server

What is GitHub?

Configuration Management

GitHub

- A web-based Git or version control repository and Internet hosting service. It is mostly used for code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.
- The flagship functionality of GitHub is "**forking**" – copying a repository from one user's account to another. This enables you to take a project that you don't have write access to and modify it under your own account. If you make changes you'd like to share, you can send a notification called a "**pull request**" to the original owner. That user can then, with a click of a button, merge the changes found in your repo with the original repo
- These three features – **fork, pull request and merge** – are what make GitHub so powerful.
- Integrated free wiki
- Simple Markdown pages for notes and general documentation
- Integrated free issue tracker
- Easy Collaboration

GitHub Guides

- https://guides.github.com/
- 3 – 15 minutes guides
- The basics

Working in GitHub

Markdown

Markdown

- https://guides.github.com/features/mastering-markdown/
- https://help.github.com/articles/basic-writing-and-formatting-syntax/

- Let's go to GitHub and look at the editor
  - Gists
  - Comments
  - Issues
  - .md and .markdown files
  - Wiki

- GitHub not just for source code
  - Someone is using it to ask for help with travel arrangements
    - https://github.com/dylanegan/travel

Working with Git

Configuration Management

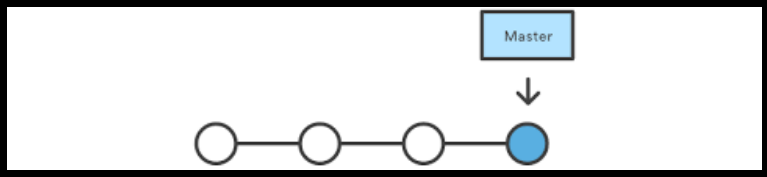
Git – Distributed Repository

- https://softwareengineering.stackexchange.com/a/35080

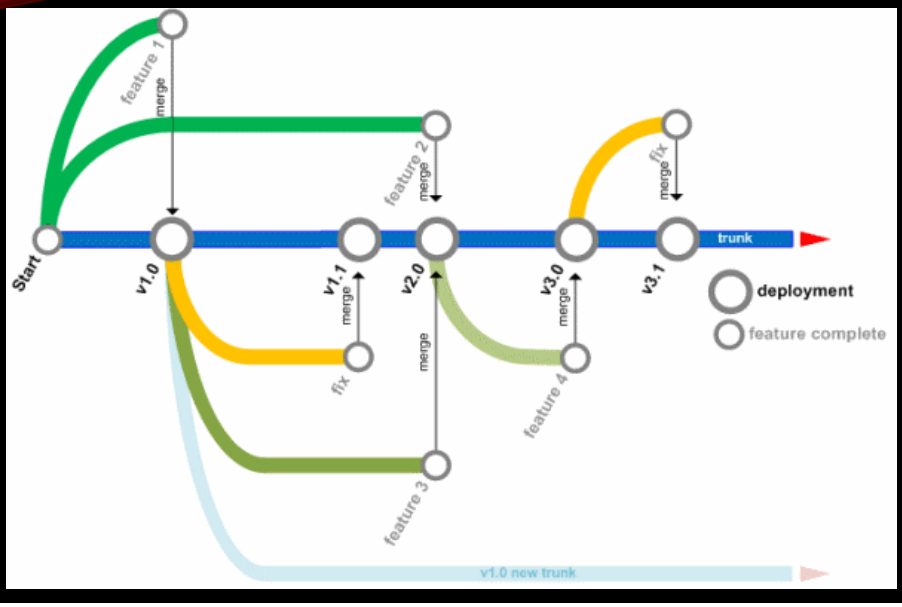
Git

- Interactive online tutorial
  - http://try.github.io/
  - http://onlywei.github.io/explain-git-with-d3/
- Visual Reference
  - https://marklodato.github.io/visual-git-guide/index-en.html
- References
  - http://rogerdudler.github.io/git-guide/
  - https://git-scm.com/docs

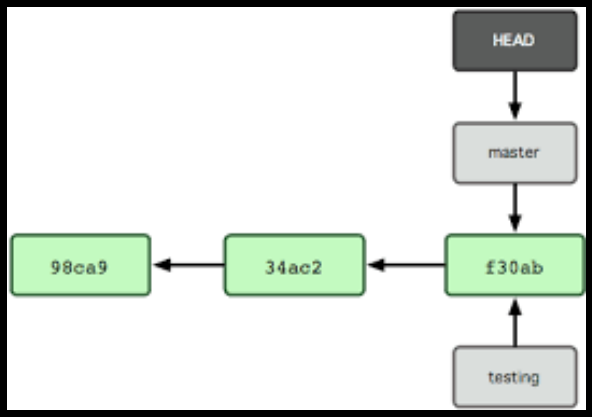
Git – Master

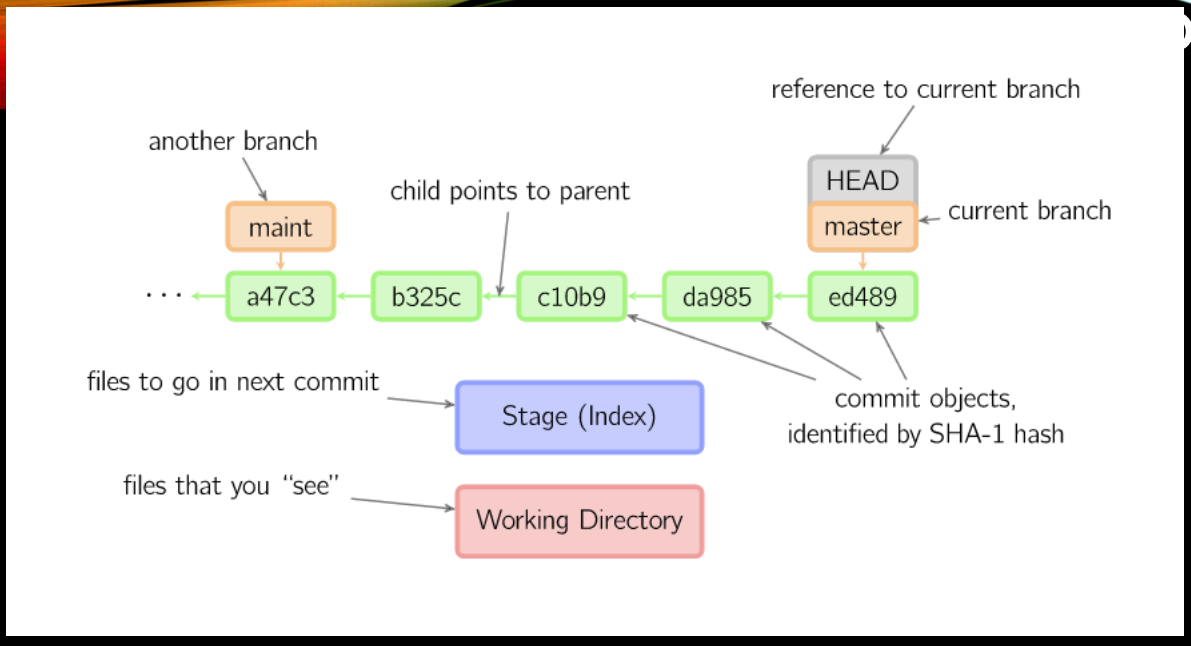


Git – Branch / Tag / Commit

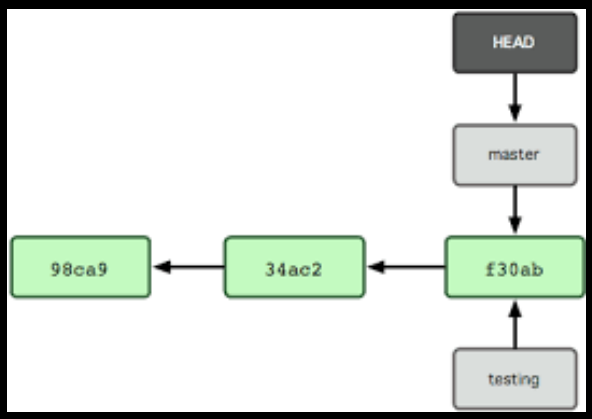


Git – HEAD

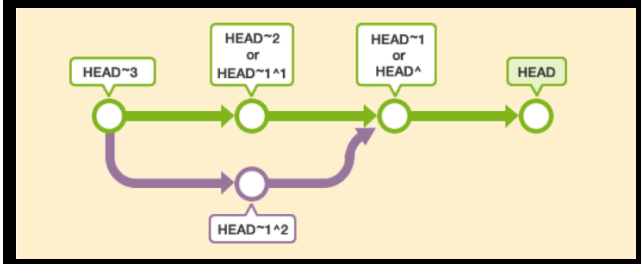

HEAD
master
98ca9
34ac2
f30ab
testing


reference to current branch
another branch
child points to parent
HEAD
maint
master
current branch
. . .
a47c3
b325c
c10b9
da985
ed489
files to go in next commit
Stage (Index)
commit objects,
identified by SHA-1 hash
files that you "see"
Working Directory

Git – HEAD



Git – HEAD (Advanced)



- The ~(tilde) and ^(caret) symbols are used to point to a position relative to a specific commit. The symbols are used together with a commit reference, typically HEAD or a commit hash.
- For instance, ~<n> refers to the <n>th grandparent. HEAD~1 refers to the commit's first parent. HEAD~2 refers to the first parent of the commit's first parent.
- ^<n> refers to the the <n>th parent. HEAD^1 refers to the commit's first parent. HEAD^2 refers to the commit's second parent. A commit can have two parents in a merge commit.

The Basics of Git and GitHub

- https://www.youtube.com/watch?v=U8GBXvdmHT4&t=566s

Working in GitHub

Issues

GitHub Issues

- https://guides.github.com/features/issues/

GitHub Closing Issues

Closing a pull request, using the following comments:
The following keywords, followed by an issue number, will close the issue:

- close
- closes
- closed
- fix
- fixes
- fixed
- resolve
- resolves
- resolved

Forking & Feature Workflow

Configuration Management

Feature & Forking Workflow

- Feature Workflow
  - https://www.atlassian.com/git/tutorials/comparing-workflows#feature-branch-workflow
- Forking Workflow
  - https://www.atlassian.com/git/tutorials/comparing-workflows#forking-workflow
- hackMGM Workflow
  - Recommend a combination of Forking + Feature Workflow
  - This is a common workflow
  - Do not specifically recommend develop, production/release type of branching
  - Anything on Master is deployed
  - https://github.com/HackMGM/hackmgm-notes/blob/master/process/hackmgm_git_workflow.md
  - https://gist.github.com/Chaser324/ce0505fbed06b947d962

GitHub - Projects

Configuration Management

Projects

GitHub Projects

- 1 minute video on GitHub Projects
- This is a Kanban style board to help manage your projects (think Trello)

https://www.youtube.com/watch?v=C6MGKHkNtxU

Git/GitHub Install

Configuration Management

Git/GitHub Install

- Git for Windows (git bash & GUI)
  - https://git-for-windows.github.io/
- Windows and Others (GUI)
  - https://www.sourcetreeapp.com/
  - https://desktop.github.com/
- Git
  - https://gist.github.com/derhuerst/1b15ff4652a867391f03
  - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Git/GitHub Install

- Sourcetree App terminal access
  - https://confluence.atlassian.com/sourcetreekb/using-terminal-in-sourcetree-781398580.html
- GitHub Desktop terminal access
  - https://saraford.net/2017/02/08/how-to-use-your-preferred-command-line-interface-from-github-desktop-039/

Exercise

Exercise

- https://guides.github.com/activities/hello-world/
- https://github.com/HackMGM/hackmgm-profiles

If you are not doing the exercise I recommend doing
- http://try.github.io/

Additional Resources

Resources

- Git Rebase
  - ??
- Git Merge Conflict
  - https://confluence.atlassian.com/bitbucket/resolve-merge-conflicts-704414003.html
- GitHub Videos
  - https://www.youtube.com/githubguides
- Git graphs
  - https://stackoverflow.com/questions/1057564/pretty-git-branch-graphs


Get to hackin'!


Extras

Agenda

- What is CM
- Concepts & Terminology
- Working in GitHutb - Markdown
- Working with Git
- Working in GitHub – Issues
- Git Workflow
- Exercise

What is CM?

Configuration Management

Configuration Management or Source Code Management

The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle. Software Configuration Management involves identifying configuration items for the software project, controlling these configuration items and changes to them, and recording and reporting status and change activity for these configuration items [SEI 2000a].

Concepts & Terminology

Configuration Management

CM

- Defined process to control and track what's going on throughout the lifecycle (different phases) of a project
- Configuration Item (CI)
  - Hardware
  - Software
  - Document (word doc, ppt, excel)
  - (Anything involved with the delivery of software or hardware)
- Control and manage change (identify, schedule, accept, and reject changes)
  - Change Control Board (CCB)
  - Tool – issue tracker
- Track those changes (what version of a CI is associated with what release)
  - Tool – GitHub or some other type of CM software

Why do you need CM?

CM

- Readily identify what functionality, bugs, changes, or fixes that were handled
- Identify what version of the software is at the different phases of your project
  - Development
  - Production
  - Currently being fixed (Maintenance)
  - What software is problematic
  - Rollback to a previous version of working software
- Connects to another part of software engineering called Requirements Management
  - Allows you to trace what requirement was implemented by identifying the software that implemented it
  - Or the reverse, revisit a requirement based on the software not functioning as expected (Was this a good requirement, does it need to be re-written)

CM

- Facilitates COLLABORATION
- Identifying who made the changes
- Merging in changes if multiple people edited the same file
- Everyone (that needs to) has access to the files
  - It's not just sitting on Paul's computer who just won the lottery and isn't returning back to work tomorrow
  - CONTINUITY

Open Source/Hobby Software

- Even in Open Source or Hobby Software, you want to be able to list the functionality and changes associated with a release.
- You could just use a word document, but if you needed to rollback your software a word document isn't going to help you.
- View a changelog

Terminology

Baseline	Lock
Release	Reserve vs Unreserve checkout
Branch	**Forking**
Stream	**Repository**
Tag	Snapshot
Label	**HEAD**
Commit	**Clean**
Checkin	**Dirty**
Push	**Stash**
Merge	Workspace
Checkout	View
Pull	

Free SCMs

- Concurrent Versions System (CVS)
- Revision Control System (RCS)
- Source Code Control System (SCCS)
- Git (GitHub, BitBucket)
- Mercurial (Heroku)
- Subversion (SVN)

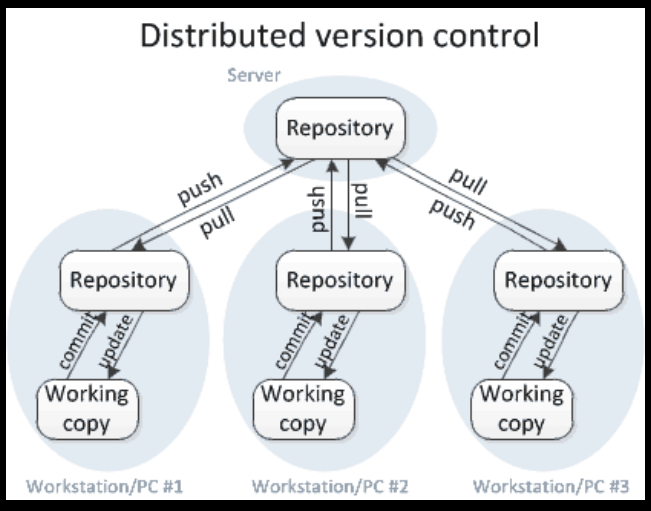
Commercial SCMs

- PVCS
- IBM Rational Clear Case
- IBM RTC

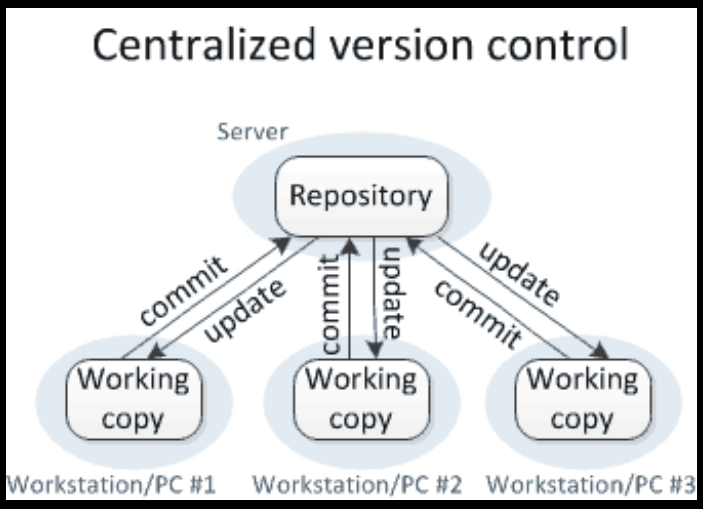
Working with Git

Configuration Management

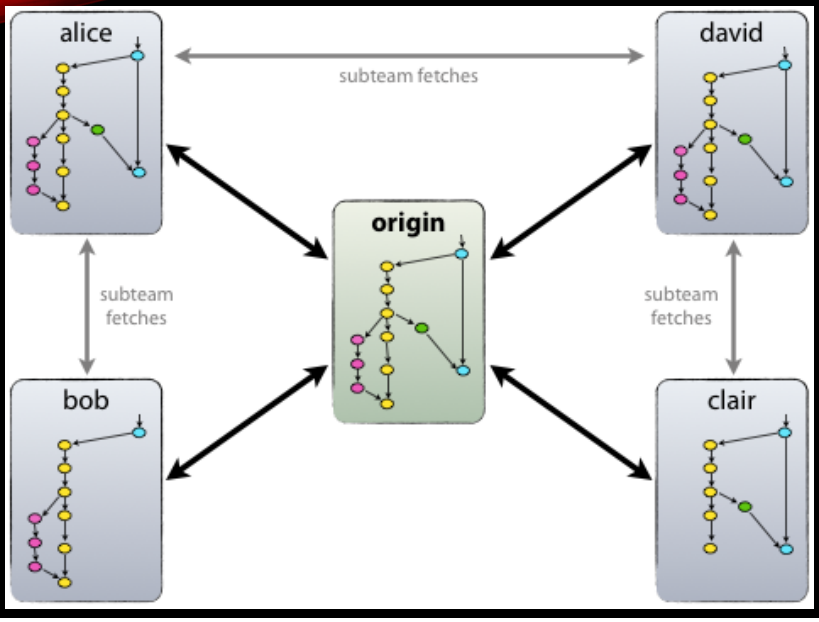
Git – Distributed Repository

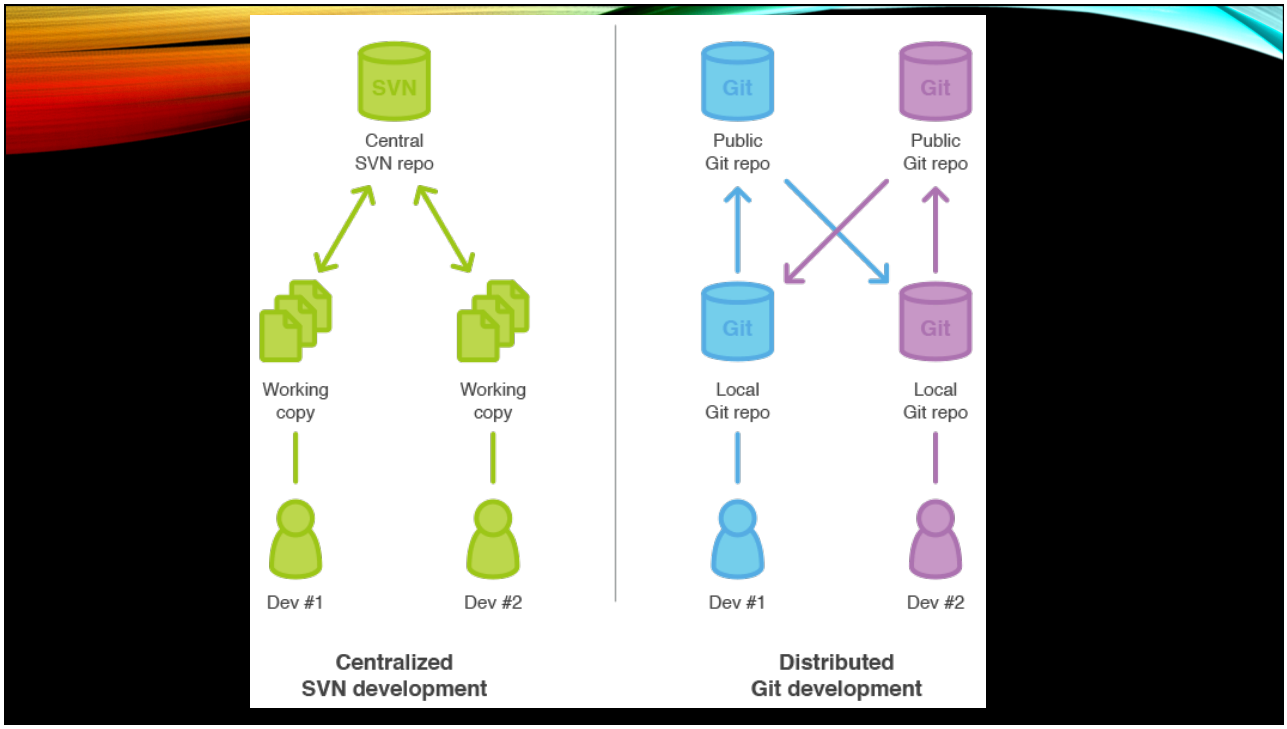


(Git is not) Centralized Repository



Git – Distributed Repository




SVN
Central
SVN repo
Working
copy
Working
copy
Dev #1
Dev #2
Centralized
SVN development
Git
Public
Git repo
Git
Public
Git repo
Git
Local
Git repo
Git
Local
Git repo
Dev #1
Dev #2
Distributed
Git development