جامعة عجمان
AJMAN UNIVERSITY

GIT & Team Tools

Practical tools for day to day development team work

*slides are heavily based on https://courses.cs.washington.edu/courses/cse403/13au/lectures/git.ppt.pdf

Mohamed Al Solh

alsolh@alsolh.com

Syllabus

Name	Date	Time	Topic	Hours
Mohamad Al-Solh	7-Jun-20	17:00 - 19:00	GIT Repositories Concepts	18
	9-Jun-20	17:00 - 19:00	Container Intro	
	14-Jun-20	17:00 - 19:00	REST APIs	
	16-Jun-20	17:00 - 19:00	Microservices and Service Mesh	
	21-Jun-20	17:00 - 19:00	BPM Implementation	
	23-Jun-20	17:00 - 19:00	Postman/K6	
	28-Jun-20	17:00 - 19:00	IOT: ESP/NodeMCU	
	5-Jul-20	17:00 - 19:00	IOT: MQTT Protocol	
	12-Jul-20	17:00 - 19:00	IOT: Node Red	

Why use a Version Control System

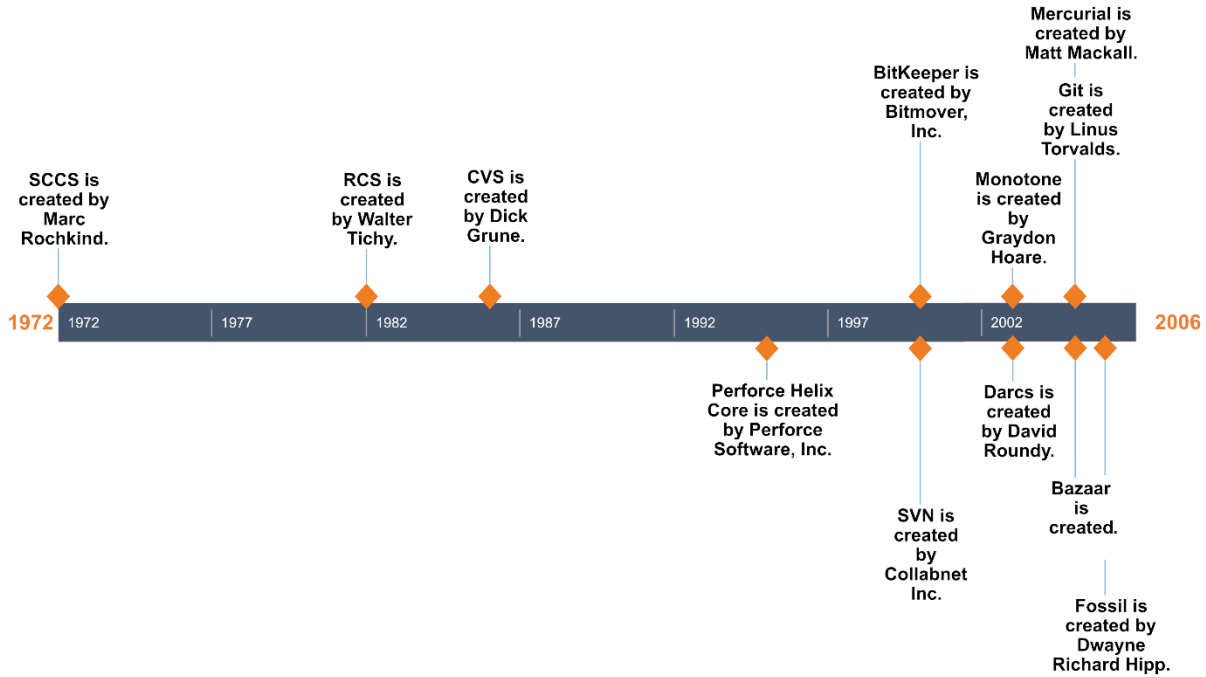
- A complete long-term change history of every file. If a mistake is made, developers can turn back the clock and compare earlier versions of the code
- Branching and merging, Traceability. Multiple developers can work on different parts of code in parallel
- Ability to restore code when developer machine is corrupted. Doing code without VCS is a huge risk that no professional team would be advised to accept

https://en.wikipedia.org/wiki/List_of_commercial_video_games_with_available_source_code#Games_with_reconstructed_source_code

https://www.atlassian.com/git/tutorials/what-is-version-control

Version Code Control History

- First Generation VCS
  - Cannot Handle Multiple files or misses checksum step for file integrity
- 2nd Generation
  - Central code repository
  - Repository may get corrupted during commits
  - Store deltas only
- 3rd Generation
  - Distributed Repository
  - Checksums
  - Compression to save space



https://initialcommit.io/blog/Technical-Guide-VCS-Internals#cvs

What is GIT

- Created by Linus Torvalds, creator of Linux, in 2005
  - Came out of Linux development community
  - Designed to do version control on Linux kernel
- Goals of Git
  - Speed
  - Support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects efficiently

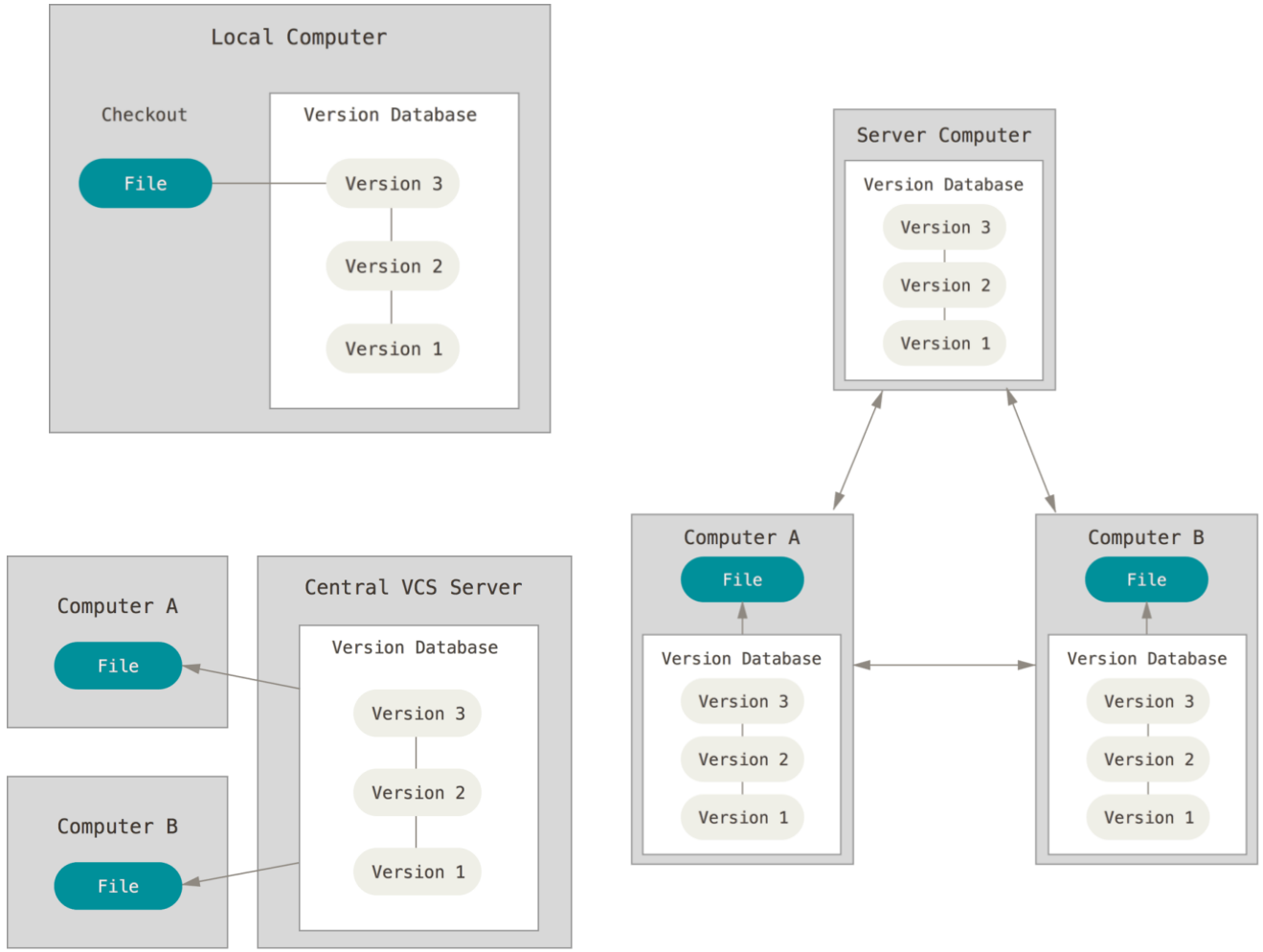
(A "git" is a cranky old man. Linus meant himself.)

https://courses.cs.washington.edu/courses/cse403/13au/lectures/git.ppt.pdf

Why GIT?

Distributed VCS vs Centralized VCS

- In Subversion, CVS
  - the server maintains the sole version history of the repo
  - When you checkout your changes are not versioned
  - When you're done, you "check in" back to the server your checkin increments the repo's version
- In git, mercurial
  - you "clone" repo and "pull" changes from it
  - Your local repo is a complete copy of everything on the remote server, local repo keeps version history
  - When you're ready, you can "push" changes back to server

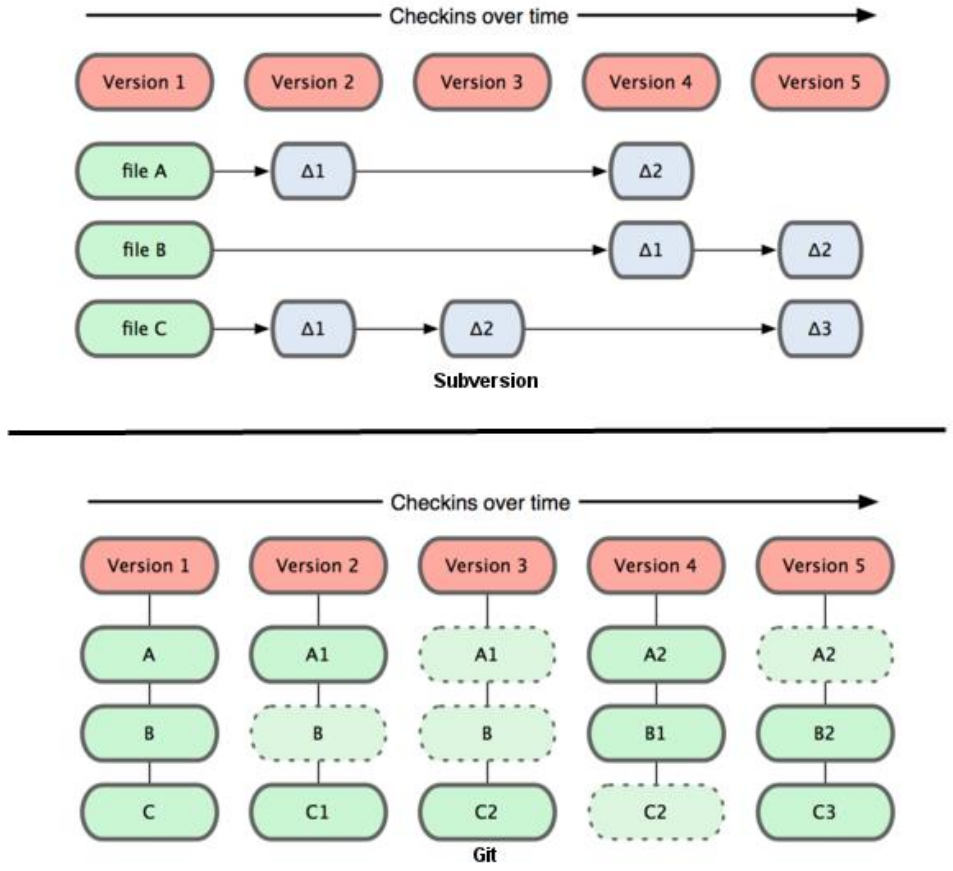


https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Why GIT?

Deltas, Snapshots

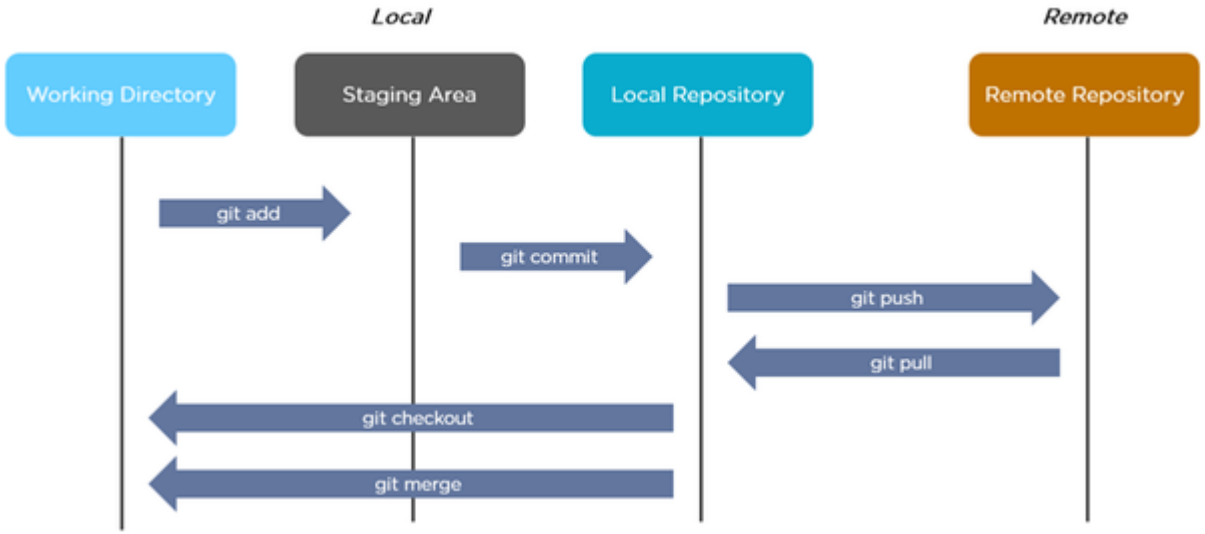
- In Subversion, CVS
  - Centralized VCS like Subversion track version data on each individual file
- In git
  - Git keeps "snapshots" of the entire state of the project.
  - Each checkin version of the overall code has a copy of each file in it.
  - Some files change on a given checkin, some do not. – More redundancy, but faster.



https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Basic Git workflow

- Modify files in your working directory.
- Stage files, adding snapshots of them to your staging area.
- Commit, which takes the files in the staging area and stores that snapshot permanently to your Git directory.



Git commit checksums

- In Subversion each modification to the central repo increments the version # of the overall repo
  - In Git, each user has their own copy of the repo, and commits changes to their local copy of the repo before pushing to the central server.
  - So Git generates a unique SHA-1 hash (40 character string of hex digits) for every commit.
  - Refers to commits by this ID rather than a version number.
- Often we only see the first 7 characters:
  - 1677b2d Edited first line of readme
  - 258efa7 Added line to readme
  - 0e52da7 Initial commit

Setup GIT Server

- There are many GIT servers available, we are going to use github for simplicity as you can try to install your favorite GIT server later:
  - Bitbucket (free up to 5 users)
  - GitBlit
  - GitLab
  - All of them have cloud based instances where you can easily start using them but you can also install locally any of these servers

Install GIT

- https://git-scm.com/downloads
- Additional Common Tools: Tortoise GIT, Source Tree, VS Code
- Initial Git configuration:
  - Set the name and email for Git to use when you commit
    - git config --global user.name “Ahmed Mohamad"
    - git config --global user.email ahmed@mohamed.com
- Creating a Git repo
  - Navigate to desired directory and use shell extension to init GIT shell
  - Create readme.md file in that directory
  - git init
  - git status
  - git add readme.md
  - git commit -m “my first git commit”

Practical GIT

- Cloning a repo
  - Navigate to desired directory and use shell extension to init GIT shell
  - git clone https://github.com/rwieruch/node-express-server-rest-api <foldername>
  - Modify some files using your favorite IDE / editor
  - Stage your files using tool or git add
  - Commit using tool or git commit –m “test change 1”

Branching and Merging

- To create a new local branch
  - git branch name
- To list all local branches
  - git branch
- To switch to a given local branch
  - git checkout branchname
- To merge changes from a branch into the local master
  - git checkout master
  - git merge branchname

Push Pull Remote Repo

- Whenever you try to push your code, if your code is not the latest commit in the branch you will have to pull first, so always pull before pushing
- Never try to force push, if you force push you will overwrite any code written by your colleagues
- To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:
  - git pull origin master
- To put your changes from your local repo in the remote repo:
  - git push origin master
- When you pull remote changes, the remote repository changes have to be merged with your local one
  - In case you are working on the same set of features with your colleagues you will most probably fall into merge conflicts, that is when the git merge tool couldn’t automatically merge the code it will leave these conflicts for you to resolve manually

Resolving a conflict

- Using visual studio code it is quite easy to manage and resolve conflicts
- Conflicts are written in following pattern

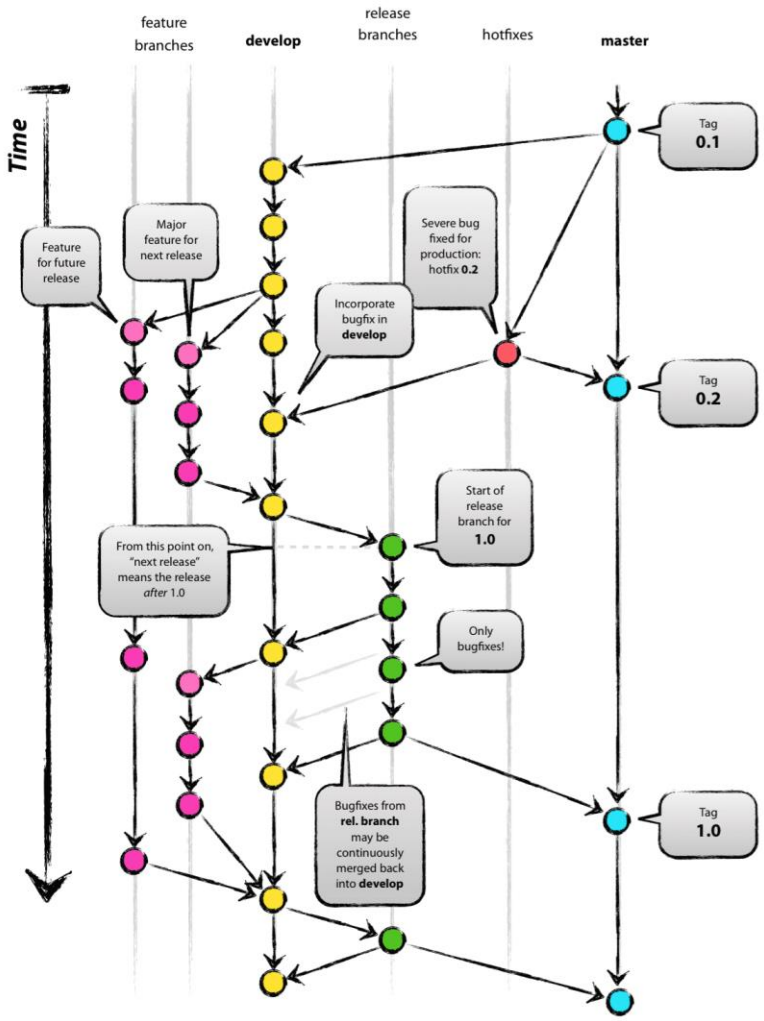
```
<<<<<<< HEAD:index.html
<div id="footer">todo: message here</div>
=======
<div id="footer">
  thanks for visiting our site
</div>
>>>>>>> SpecialBranch:index.html
```

branch 1's version

branch 2's version

GIT Recommended Branching Strategy

- Best Practice doesn't always mean it will fit any situation, you may have to change if you have multiple projects running in parallel in different environments
- Master branch contains stable code
- Develop branch is where we all developers work on it
- Feature branches are for new features
- Release branches for any planned releases
- Hot fixes always originate from master
- Tags are created for every production deployment to reserve an immutable reference of the code deployed



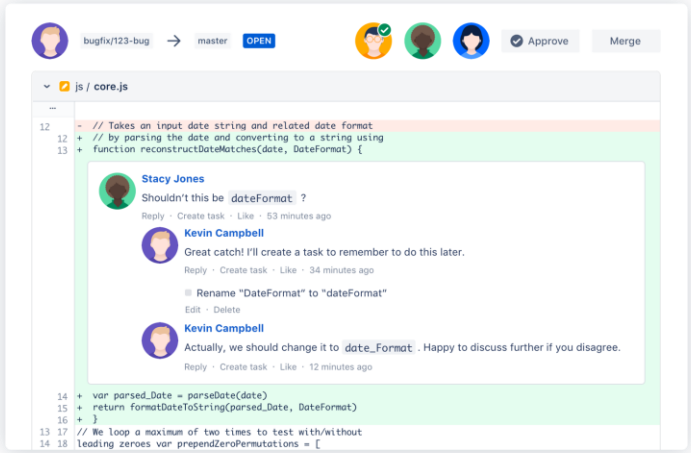
https://nvie.com/posts/a-successful-git-branching-model/

GIT Cheat Sheet

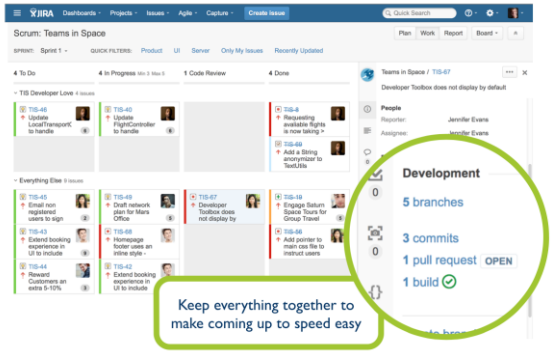
- https://education.github.com/git-cheat-sheet-education.pdf

Pull Requests

- A vital part of team work is to review each other's code, this increases awareness amongst team members and exchanges ideas.
- You will have to take any comment on your code constructively
- To perform pull requests:
  - You will have to fork the repository with your own version
  - When you push your code it will be held for review and then merged by code reviewers



Important Tools to work as a Team



- You always need a bug tracker, there are many tools available but the most complete tool that also manages your agile sprints is JIRA, open source alternative is mantis or redmine
- You need to track code quality, a tool called SonarQube does the job and scans for security risks too
- You need to setup build pipelines to automate repetitive tasks, best tool known is Jenkins
- Jira integrates with every other known tool in the market, that is why it is favored and used by most although it is not open source
- You need a wiki, as part of Atlassian products you can use confluence, but opensource alternatives exist such as mediawiki
- All these tools help a new concept being followed in companies called DevOPS

Home Work

- Create a Jira Cloud Instance – 10min
- Create a Bitbucket instance – 10min
- Write your project scope in the read me file – 15min
- Create Tasks and describe your tasks business wise what needs to be done – 30mins
- GIT Basics – 10 min
  - Pull code from favorite repository into your separate repository
  - Modify a file
  - Push to your remote bitbucket repository
  - Resolve a merge conflict