Version Control

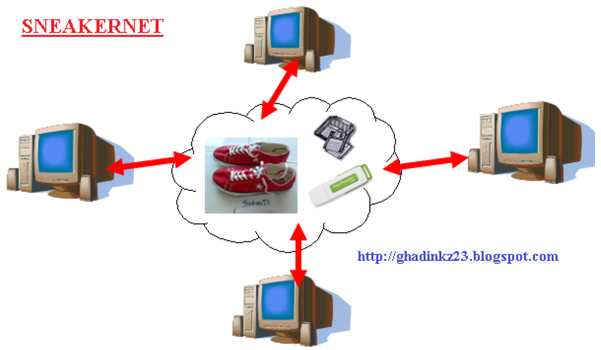
Project Contribution

- There are open-source projects hosted in repositories on GitHub
- Your task is to contribute your own code to one of these projects

- **But how?**

How to Share Code

- The entire class is effectively one large team
  - We need to share code with other
- How?
- Email attachments?
- Sneakernet?
- What if two people edit the same file at the same time?
- Google Docs? Dropbox?
  - No merging in Dropbox
  - GoogleDocs not meant for code
    - No syntax highlighting, etc.



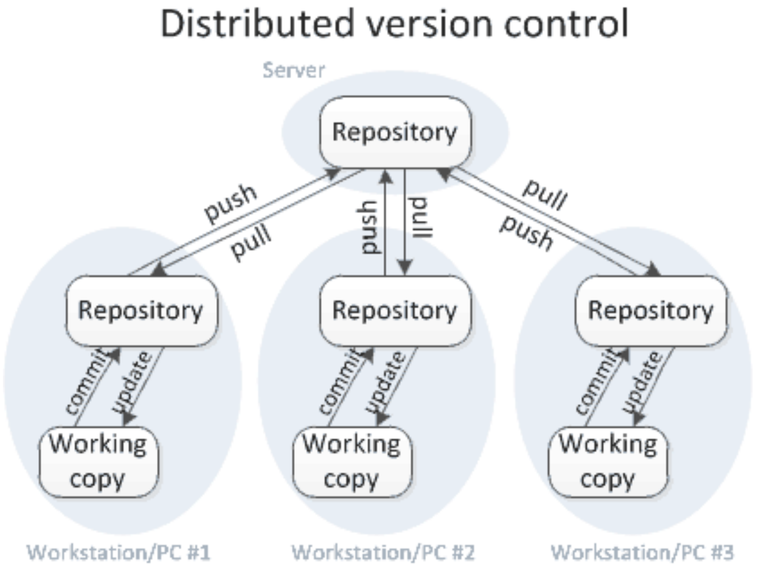
https://en.wikipedia.org/wiki/Sneakernet

git

- Designed for code
- Use for the examples repo and all homework handouts
- What git does for you:
  - Version control
    - Track numbered releases (eg. v2.1.19)
  - Source control
    - Tracks all changes made to the codebase
    - Create working branches without affecting stable releases
  - Can merge when two people change the same file

git

- Distributed version control software
- Remote server hosts your repository (GitHub)
- Every user maintains a local copy of the entire repository



https://homes.cs.washington.edu/~mernst/advice/version-control.html

(some) git Commands

- Make a repo
  - init
- Update a repo
  - add
  - commit
  - push
- Connect to a remote repo
  - clone
- Update local repo
  - pull

To Make a Repository

- init
  - Creates a new git repo in the current directory
- Or create a repo in GitHub or IntelliJ
  - GitHub/IntelliJ will call init

- clone
  - Downloads an existing repository

Updating the Repo

- add
  - Tell git which files it should track
  - Don't add files not related to the code (Ex. .idea files. Libraries)
- commit
  - Updates changes to your local repository
  - Add a meaningful commit message
- push
  - Updates the remote repo with the changes committed to your local repo
  - Lets your team access your changes
  - Will warn you to pull first if there are remote changes

Commit messages

- Make your messages meaningful and descriptive
  - Your future self and contributors will thank you!
  - Especially as you move on to bigger and better projects

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

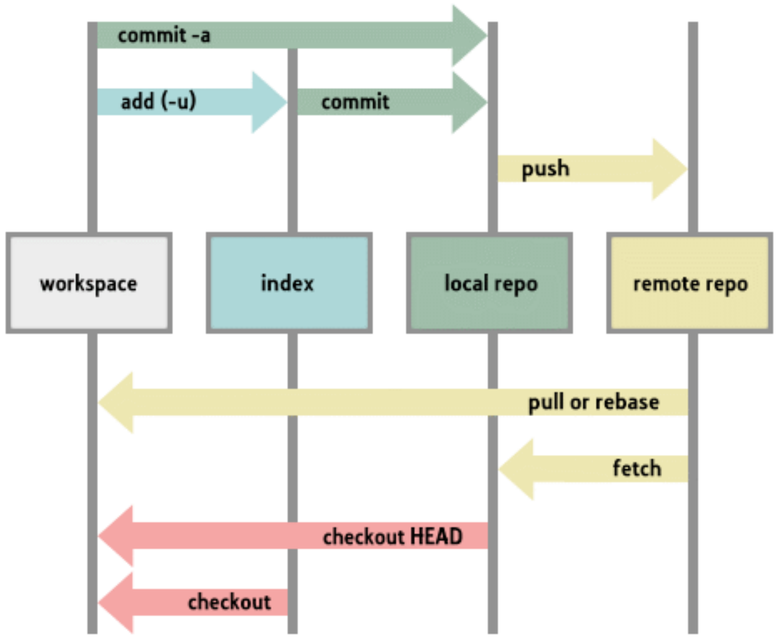
https://xkcd.com/1296/

Updating Your Copy

- pull
  - Copies the most up-to-date code from the server to your local copy of the repo
  - Can cause a merge conflict if you changed files that are changed on the server

- pull often!

git Structure



https://www.sonassi.com/knowledge-base/our-magento-git-guide-and-work-flow/

GitHub

- Provides servers to host git repositories
  - Has an web interface to view and interact with repositories

- GitHub != git

- git is the version control software
- GitHub is a web service that hosts git repositories

Branching

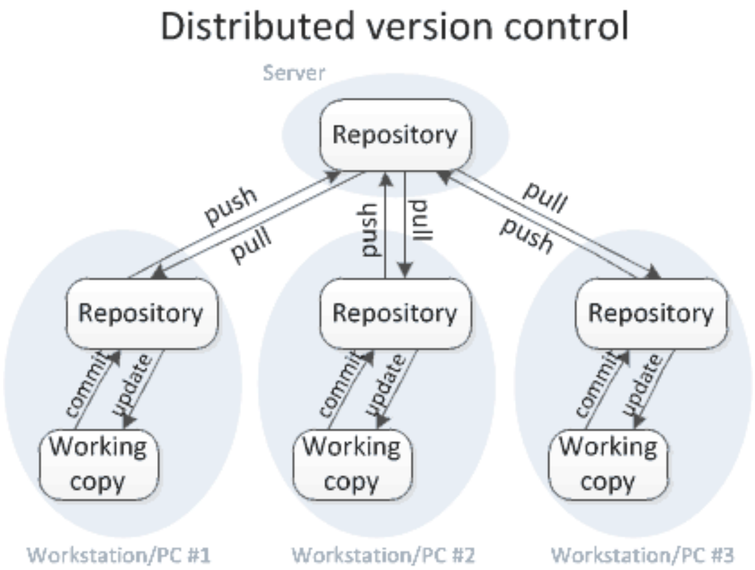
Version Control - Branching

- A way to write code without affecting the rest of your team

- A branch in a repository is a "copy" of all the code
  - This copy can be edited without affecting the rest of the team

- Merge branches to integrate your changes into the master branch

Version Control with git - Recall



https://homes.cs.washington.edu/~mernst/advice/version-control.html

Branching

- New commits are only added to the current branch
  - Can implement experimental code without affecting others
- When the life of the branch is over
  - Merge it into the primary codebase
- Branching is crucial on large teams

Merging

- Once the changes are complete and stable
  - Merge the branch back to the codebase
- Procedure:
  - Initiate a pull request
  - Other developers perform code reviews
  - If the new code is acceptable the pull request is accepted
  - Code is merged into the project

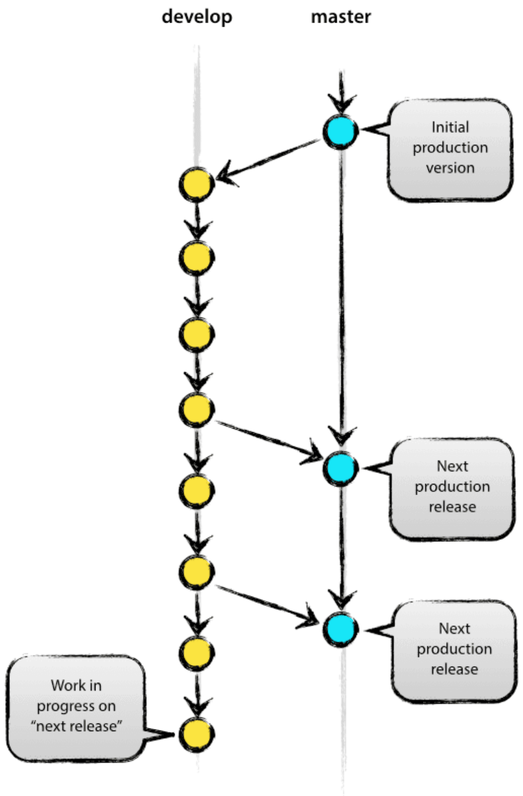
A Branching Model

Master and Develop

- The two primary branches for the codebase

- Master is the default branch in git
  - Used for stable releases
  - This is the state of the latest release available to users

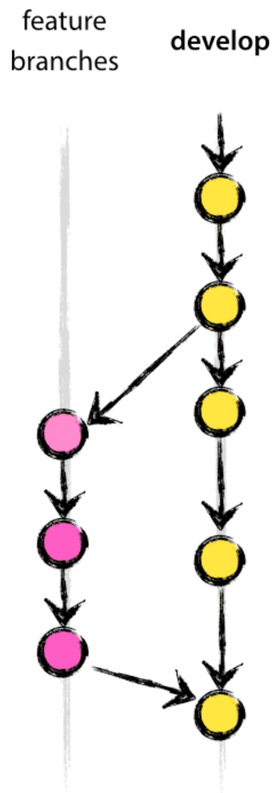
- Develop branch
  - The current state of the project
  - Contains all completed features (Includes unreleased features)



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

Feature Branch

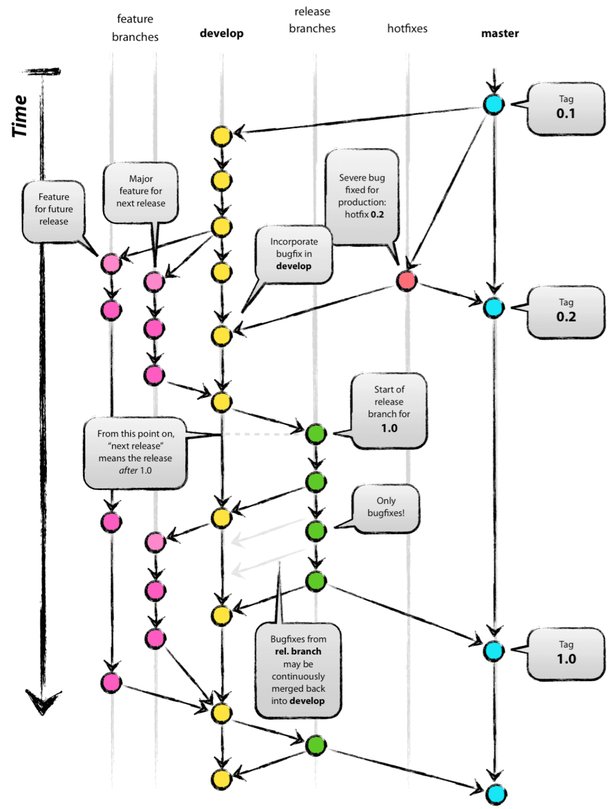
- Temporary branches
- Branch off of develop
- Primary working branches for an individual(s)
- When a new feature is finished
  - Merge into develop
  - Hope it doesn’t break develop (it might)
- Can be many feature branches being developed in parallel
- If the feature isn't finished
  - Delete the branch without merging into develop



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

Versioning

- Tag the current state of the code
- Can easily work with a previous version if needed
- Use versioning on master



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

Forks

- Fork to make a complete copy of a repository
  - As opposed to branding which creates a new copy of the code within that repo
- Typically fork a repo when you do not have push access
  - Forking can simulate having your own branch of a repo
  - Used to contribute to open-source repositories

Open-Source Contributions

- Procedure:
  - Fork the repository of the project to which you want to contribute [On GitHub]
  - Clone the repository [In IntelliJ]
  - Edit the code with your contribution
  - Commit and push your changes to your forked repository
  - Create a pull request from your forked repo into the appropriate branch [develop branch] of the original repository [On GitHub]

Pull Request Demo