Version Control with Git and GitHub

A Short Introduction
What is version control?

• Aka revision control, source control
• Version control: the management and tracking of changes to source code, documents, data, etc.
• Allows collaborative development
• Keeps track of who made a change, when the change was made, and what the change was
• Permits reverting any change and rolling back to a previous state

• Many systems available: CVS, Subversion (SVN), Perforce, git, Mercurial,...
What is git?

- **Distributed** revision control system
  - Speed
  - Data integrity
  - Distributed, non-linear work flows

- Created in 2005 by Linus Torvalds to support the Linux kernel development

- Main characteristics:
  - the entire code and history is kept on the client (user) machine
  - users can work (make changes to code) even without internet connection
  - internet connection required only for pushing and pulling from remote server (remote repository)
Git basics – commits

- Git keeps track of code history in snapshots
  - record of what all files look like at a given point in time
- User decides when to take a snapshot (commit) and what files should be included
- Allows going back and visiting any past snapshot
  - later snapshots are not lost
- A project is made out of a series of commits
- Each commit contains:
  1. information on how the files changed from previous commit (diff)
  2. a reference to the previous commit (parent commit)
  3. a hash code name
Git basics – repositories

- A repository (or ‘repo’) is a collection of all the files and their commit history
  - contains all commits
  - can be local or remote
- Copying a repository from a remote server is called cloning
- Cloning allows teams to develop collaboratively
- Pulling: downloading commits that do not exist on the local machine from a remote repository
- Pushing: adding local changes (commits) to a remote repository
Git basics – branches

• All commits in a repository live in some branch
• The main branch in a repository is called the master branch
• A project can have many branches
  • For example, a project that follows GitFlow will have a master branch, a develop branch, feature branches, hotfix branches, release branches
• Branches allow maintaining parallel and separate development tracks in a single project
• Development tracks can be
  • branched off
  • merged
What is GitHub?

• Largest web-based git repository hosting service  https://github.com/
• Founded in 2008
• Promotes open source, but also has an Enterprise Edition for businesses

• Offers all Git distributed version control functionality

• Additional functionality:
  • User interface
  • Documentation
  • Bug tracking
  • Feature requests
  • Pull requests
GitHub basics – forking
GitHub basics – pull requests

- Public Repo
- Mirrored Repo
- Local Repo
- Pull Request (PR)
- Pull Request
- Fork
- Merge
- Update/Commit/Push
**Sourcetree GUI client**

- **Current project**
- **Local branches**
- **Remote branches**
- **Log message for selected commit**
- **Selected file in commit**
- **Commit hash code**
- **Who made the commit?**
- **When was the commit done?**
- **Selected commit**
- **File differences (wrt parent commit)**
GitFlow

- Proposed by Vicent Driessen
  http://nvie.com/posts/a-successful-git-branching-model/
- A development model based on Git
Git/GitHub resources

• Git official website: https://git-scm.com
• GitHub guides: https://guides.github.com
• Git tutorials and training: https://www.atlassian.com/git/tutorials/
• Git cheat-sheet: https://services.github.com/kit/downloads/github-git-cheat-sheet.pdf
• GetFlow cheat-sheet: http://danielkummer.github.io/git-flow-cheatsheet/

• Git GUI clients: many options (see https://git-scm.com/download/gui/linux)
  • Windows/Mac: Sourcetree (by Atlassian)
  • Linux/Windows/Mac: SmartGit, git-cola