Introduction to VCS and Git

Christos Kotselidis & Bijan Parsia
COMP61511 Week 3
October 2018
Background

- Version Control Systems (VCS) allow us to manage our code in a structured way
- They enable historic snapshots of our code in the form of changesets
- We can revert our changes to a previous state
- Increase productivity and collaboration through shared repositories and branches
- Many variations (CVS, SVN, Mercurial, Git)
Git

- Git is the state-of-the-art in distributed VCS
- Started in 2005 to support Linux development
- Large ecosystem with various online systems hosting Git
  - Github, Gitlab, Bitbucket, etc.
- We will use Gitlab in our course
Git core concepts

- **Repository** is the location where our code resides
  - Local and/or Remote

- **Changeset** is a record/state/snapshot of our code
  - Has an associated hashcode with it
  - Can be created by committing our code

- **Branch** is a line of development of our code (collection of changesets)
  - The initial branch we work on is usually called the *master* branch
  - We can create multiple branches
  - Typically used for *feature* development

- **Merge** is the act of combining the changesets of two branches
  - It can be conflict-free or not
Git core concepts

Remote Repo (in GitLab)
By issuing `git clone` we create our local mirror of the remote repo

Local Repo (in your computer)
By issuing `git push` we publish our local changes to the remote repo

How our code looks like?
Git core concepts

Working with branches
  • Branches allow us to create separate paths of work
  • Separation between stable branch (master) and other branches used for:
    • Hotfixes, Features, etc.
  • Enable better collaboration and integration between developers and teams
  • Example: Add more unit tests in miniWC on a separate branch

```
git checkout -b unittest
```
Git core concepts

Time to merge our changes back to the master branch
1. Switch our current working branch to master: `git checkout master`
2. Merge our unittest branch: `git merge unittest`
3. (optional) Delete our unittest branch: `git branch --d unittest`
Making our branch public

- Initially the newly created branch reside in our local repository
- In order to make our branch public and share it with other team member we need to publicize it
- We achieve that by issuing `git push -u origin` from within the branch we want to publish
- Other collaborators who have cloned the same public repo must issue:
  - `git fetch` and `git pull` to get the new branch
• Git can become very complex to use
• **Key** is to understand basic concepts
• **Advise:** To avoid merge conflicts try to develop new features always in separate branches and then merge to master
• However, this might not always be possible
  – Manual merging is needed in case of conflicts
Additional Resources

Git Fundamentals

- https://agripongit.vincenttunru.com
- https://www.slideshare.net/HubSpot/git-101-git-and-github-for-beginners

Advanced Git

- Gitflow methodology: https://datasift.github.io/gitflow/IntroducingGitFlow.html
- Advanced Git podcast from Dr. Foivos Zakkak: https://video.manchester.ac.uk/faculties/de9a2b394122e0a46b94a3df893620a9/4785c4d1-5f0f-4d1a-92ae-8cf6f91ea434/