Version Control

EECS 2311 - Software Development Project

January 11, 2017

YORK UNIVERSITÉ UNIVERSITY

Scenario 1

- You finished the assignment at home
- You get to York to submit and realize you did not upload it

- Has this ever happened to you?

Scenario 2

- Your program works pretty well
- You make a lot of improvements ...
  - ...but you haven't gotten them to work yet
- You need to demo your program *now*

Scenario 3

- You are working on the 2.0 version of “your great app.” But 2.0 does not quite compile yet… and customer finds a critical bug in 1.0, which must be fixed ASAP.
- If you're smart, you have a copy of your 1.0 source. You make the change and release, but how do you merge your changes into your 2.0 code?
- If you're not so smart, you have NO source code saved. You have no way to track down the bug, and you lose face until 2.0 is ready.

Scenario 4

- You change one part of a program - it works
- Your teammate changes another part - it works
- You put them together - it does not work

- What were all the changes?

Scenario 5

- You make a number of improvements to a class
- Your teammate makes a number of *different* improvements to the *same* class

- How can you merge these changes?

A poor solution

- There are a number of tools that help you spot changes (differences) between two files, such as diff
- Of course, they won't help unless you kept a copy of the older version
- Differencing tools are useful for finding a *small* number of differences in a *few* files
- A better solution…

Version control systems

- Keep multiple (older and newer) versions of everything (not just source code)
- Request comments regarding every change
- Display differences between versions
- Allow merging of changes on the same file

Centralized Version Control

- Traditional version control system
  - Server with database
  - Clients have a working version
- Examples
  - CVS
  - Subversion
- Challenges
  - Multi-developer conflicts
  - Client/server communication

Distributed Version Control

- Authoritative server by convention only
- Every working checkout is a repository
- Get version control even when detached
- Backups are trivial
- Examples
  - Git
  - Bitkeeper

Terminology

- A **repository** contains several branches
- The main branch is called the **master**
- **Branches** break off from the master to try something new, e.g. a new feature, code restructuring etc.
- Branches can be merged with other branches or into the master
- **Tags** are usually official releases that have to be supported

Let’s work with git

- We need to do the following:
  - Create a repository online
  - Create a local repository, add a project to it, and push it to the online repository
  - All team members get the online repository
  - Changes pushed by one team member can now be pulled by all

Once per team

- Go to github.com
- Sign up for a new account
- Create a new repository
- Copy the URL to access your repository

Once per team

- Run Eclipse
- Create a new project called ProjectWithGit that contains a main method that prints “Fun with Git”
- Go to Window -> Preferences -> Team -> Git -> Configuration
- Click Add Entry, add the pair [user.name, yourname]
- Click Add Entry, add the pair [user.email, youremail]
- These should be the same as the ones used at github.com
- Click Apply, then OK

Once per team

- Rightclick on ProjectWithGit, and select Team->Share Project…
- Select Git, and hit Next
- Click on Create…
- Provide a name for your local repository, and click Finish
- Your **local** repository is now setup.

Once per team

- Rightclick on ProjectWithGit and select Team -> Commit…
- Provide name, email
- Add a commit message
  - It is important that you add a message every time you commit, makes it much easier to find a version later
- Select all files, and click Commit
- Close editors, reopen, make a change to the output of your program and Commit again

Once per team

- Rightclick on ProjectWithGit, select Team -> Remote -> Push…
- Copy the URL from github.com in the URI field
- Enter your github.com username and password, click Next
- Select master from the Source ref pull down menu
- Click on Add All Branches Spec
- Click Finish, then OK
- You should be able to see ProjectWithGit in github.com

Once per team

- Due to an Eclipse bug, it is now easier to delete the local repository, and re-get it from github.com along with the other team members
- Rightclick on ProjectWithGit in Eclipse, and select Delete.
- Select to delete project contents on disk, and click OK.

All team members

- Go to File -> Import -> Git -> Projects from Git
- Click Next, select Clone URI, click Next
- Copy the URL from github.com on the URI field
- Keep clicking Next, and finally Finish
- You now have a copy of the project in your local repository
- To push changes to the remote repository, you will need a github.com account that is added as a collaborator

Push

- Make some changes to any of the classes in the project
- Rightclick on any element that has changes (could be the whole project), and select Team -> Commit
- Add a commit message
- If you do not want to publish the changes yet, click Commit
- If they are ready to be published, click Commit and Push

Pull

- To get changes published by other team members, rightclick on the project, and select Team -> Pull

Lab Task

- Get git working for every team member
- This should be for code / documents / notes etc.
- Demonstrate that everybody can pull / push code on Monday's lab