How can 4 (or 4000) developers work on a product at once?

Revision Control

Revision Control

• Revision Control:
  – Also called version control, source control, software configuration management.

• Motivation:
  – Need to coordinate changes made by multiple developers.
  – Need a reliable system to ensure changes are..
Git Basics
Local Topology Simplified

- Local Machine has a...
- The latest code in the repo can be checked-out into the working directory.
  - Head: the latest version of the code.
- Changes to files in the working directory are committed to the local repo.
Remote Topology Simplified

- Remote Server has a Git Repo
  - Server accessed by multiple developers
  - Local repo syncs up with remote
Distributed

• Distributed Version Control
  – Git has..
    each “local repo” is a full and complete repo.
  – Can work off-line (on a plane) and still commit to the local repo. Later sync up with the remote repo.

• Git Servers
  – Often the remote repo is a dedicated Git server such as GitHub or GitLab.
  – These systems add extra team collaboration and discussion tools (more later).
Work Flow 1: Setup

• Associate your local repo to a remote repo by either:
  – Create a repo in GitLab (gitlab.cs.sfu.ca) and push some existing code to it; or
  – .. an existing repo to your local PC.
Work Flow 2: Changes

- Do some work in working directory
  - create new files, change files, delete files, etc.
- Stages the changes as being ready to commit.
  - Also used for adding files to Git (tracking them)
- Commit all staged changes to local repo.
  - Sometimes termed “Check-in”
- Send committed changes to remote repo.
- View the state of local file changes
Work Flow 3: Other’s Changes

- Other team members will push some changes to the repo which you then want
  - May be new / changed / deleted files

- Get changes from remote repo and apply them to local repo and working directory (move to head).
  - If there are any conflicting changes, may need to do a *merge* (more later).

- At any time, can view the changes people have made.
Git Tools

• Command Line
  - Git is very often accessed via its command-line tools
  - Git commands look like:
    
git clone git@csil-git1.cs.surrey.sfu.ca:myTeam/daProject.git
  
git commit

• GUI Integrated Tools
  - .. but low-level understanding is required!
  - Can be inside IDE: Android Studio
  - Can be integrated into file system: TortoiseGit
  - Lecture: command line to understand the tool;
    Assignments: IDE for convenience (likely).
Command-line Demo

- Git Command Demo
  
  `[create repo on csil-git1.cs.surrey.sfu.ca]`
  
  - `git clone <git@csil-git1.cs....>`
    
    `[now edit file hello.txt]`
    
    - `git status`
    
    - `git add hello.txt`
    
    - `git commit`
    
    - `git push`
    
    - `git log`
    
    - `git pull`
Git Details
Basic Git Sequence for Editing Code

0. Have a working directory with no changes
1. ..
   – will "fast-forward" without any conflicting changes
2. ..
   – cannot pull with some uncommitted changes
3. ..
4. ..
   – automatically merges files without conflicting changes
   – manually merge conflicts when required
5. ..
   – cannot push if others have pushed code:
     “current branch is behind master”, “unable to fast-forward”
Merge Conflict Demo

• Show demo of conflicting changes being made by two team members at once
  – Pulling with uncommitted conflicts fails
  – Pushing before merging fails
  – Commit my changes
  – Pull to trigger merge
  – When merge done then add/commit/push

• Android Studio has VCS --> Update Project
  – Which works with uncommitted conflicts
  – It automatically stash changes to get around having to do extra commit
.gitignore

- .gitignore File
  - Lists file types to exclude from Git:..

- Example:
  Exclude .bak, build products, some IDE files
Commit Messages

• A good commit message is required!
  - Line 1: ..
    (<=70 characters)
    Capitalize your statement
    Use imperative: "Fix bug..." vs "fixed" or "fixes"
  - Line 2: ..
  - Line 3+: ..; wrap your text ~70 characters

Example: Make game state persist between launches and rotation.
          Use SharedPreferences to store Game's state. Serialize using Gson library and Bundle for rotation.

• 276 Pair Programming
  - If pair programming, add pair’s user ID at start:
    “[pair: bfraser] Make game state persist ....”
Reverting Changes

• ‘git checkout’ to revert files
  – ..
  – Overwrite file in working directory with one from local repo.

• Revert with Caution
  – Will lose all uncommitted changes in the file.
  – Normally Git does not let you lose changes.
  – If in doubt, grab a backup copy (ZIP your folder) then revert.
    • Just make sure you don’t commit the backup!
Delete, Rename

• Delete file
  – Delete file normally via the OS/IDE,
    
    ..
    
    Git records it's now deleted.
  – Will be deleted on everyone else's system when they pull your changes.

• Rename file
  – Rename file normally via the OS/IDE, then "add" it to Git
  – Git tracks files by their content, not by their name.
Revision Control
Generalities
Merge vs Lock

2 Competing ways revision control protects files:

- **Checkout-Edit-Merge**
  - Merge support allows concurrent access to a file so multiple developers can work on same code at once
  - But can lead to...

- **Lock-Edit-Unlock**
  - Locking prevents merge conflicts by..
    - "I can't make any changes until Bob finish!"
  - Adds pressure to make changes quickly..
    - "I need that file now!"
Revision Control Features

- Atomic operations
  - Change is applied all at once: no other changes applied while you're checking in.

- Tag
  - Mark certain versions of certain files as a group. Ex: "Files for Version 1.0 of product".
  - Able to easily.. of the files later to fix bugs etc.
    - "Get all files exactly as the were in Version 1.0 (three year ago)".
Team Work

- Minimum requirement to committing code:
  - When you check in, the full system must compile and run.
  - Only under exceptional circumstances should you ever check in something which breaks the build.
Committing Frequency

• Expected Commit Frequency
  – Commit little changes to local repo very often
  – Once some work is more stable, push all the changes at once to remote repo.

• CMPT 276
  – Committing / pushing this frequently gives visibility to your contributions; helps for marking discussions!
  – In a ‘professional’ project, you would tailor your commits/pushes to the work you are doing, and squash small commits together into bigger more meaningful ones.
Coding with Source Control

- // Removed Jan 2002 for V1.01
  // cout << "Dave; I wouldn't do that, Dave.";
- Put meaningful comments into checkins!

- 
  #if 0
  // Unneeded, but left 'cuz someone may want it...
  ......
  #endif

- 
  // Written by Dr. Evil
  ....
Summary

• Revision control a critical tool for development.
  – Git is a distributed revision control system.

• Operations:
  – clone, add, commit, push, pull, merge (later)

• Git Details
  – Merge conflicting changes as needed.
  – .gitignore, revert (git’s checkout)

• Basic Features
  – Atomic operations, tags/Label

• Rules to Code By
  – Commit often, don’t break the build