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

1. Login to Linux (if in Windows, reboot) while booting select to boot Linux
2. For Notes and Videos tinyurl.com/ssssGitWorkshop

Revision Control

• Revision Control
  – a system to manage changes to electronic documents.
  – Also called version control, source control, software configuration management.

• Motivation
  – Need to coordinate changes made by multiple developers.
Git Graph / Log / History

source: https://www.atlassian.com/git/tutorials/using-branches/git-merge
Git Basics
Local Topology Simplified

- Local Machine has a Git repository (Repo)
  - Usually in `.git/` directory
  - Checkout code from repo to working directory.
Remote Topology Simplified

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

Local Computer

Working Directory

Local Repository

Remote GitLab Server

Remote Repository
Git Details
SSH Key

- GitLab verifies you via an SSH key (no passwords)
  - Generate the key on each machine you use
    (all CSIL machines will share your SSH key)
  - Open terminal and run:
    $ ssh-keygen -t ed25519
    (press enter until done)
  - View key; highlight and copy:
    $ cat ~/.ssh/id_ed25519.pub

- On GitLab (gitlab.cs.sfu.ca)
  click avatar (top right) --> Settings --> SSH keys
  - paste SSH key; give title “CSIL”; and add it.

- Now GitLab will allow you access!
  $ ssh -T git@csil-git1.cs.surrey.sfu.ca
Git Graph / Log / History

source: https://www.atlassian.com/git/tutorials/using-branches/git-merge
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
  - Clone an existing repo to your local PC.
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”
Your Turn!

1) Create *empty* repo on gitlab.cs.sfu.ca
2) Create Java project in IntelliJ; add a Readme.txt
3) Commit to local repo (this adds and commits)
4) Push to remote repo
   Set origin to git@csil-git1.cs.surrey.sfu.ca.____.git
   (get ____ from GitLab repo’s “clone” button)

   If you mistakenly created a non-empty repo, it’s easiest to create a new empty repo (no readme even!) and push to it.

5) Make another change, commit, push
Working in a Team
# Let’s try it with a partner

<table>
<thead>
<tr>
<th>Person A</th>
<th>Person B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Add ‘B’ to your repo</td>
<td>2. Clone repo</td>
</tr>
</tbody>
</table>
| 3. Add hello.java, push it (loop to print 10 ‘hellos’) | 4. VCS --> Update  
Edit hello.java & push |
| 1. Pull | 4. Change hello.java at bottom |
| 2. Change hello.java at top | 5. Push (fails) |
| 3. Push | 6. VCS --> Update |
| | 7. Push (succeeds!) |
Merging with Partner

<table>
<thead>
<tr>
<th>Person A</th>
<th>Person B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Pull</td>
<td>1. Pull</td>
</tr>
<tr>
<td>2. Change hello.java’s loop (for/while/do-while)</td>
<td>2. Change hello.java’s loop (for/while/do-while)</td>
</tr>
<tr>
<td>4. Push (fails)</td>
<td>3. Push</td>
</tr>
<tr>
<td>5. VCS --&gt; Update</td>
<td></td>
</tr>
<tr>
<td>6. Resolve merges</td>
<td></td>
</tr>
<tr>
<td>7. Push</td>
<td>8. Pull</td>
</tr>
</tbody>
</table>
.gitignore / delete / add / rename

• .gitignore File
  – Lists file types to exclude from Git:
  – Example:
    Exclude .bak, build products, some IDE files

• Delete / Add / Rename Files
  – Just delete / create the files in working directory
  – Then execute Git commands:
    • “add” changed files
    • “commit”
    • “push”
Commit Messages

• A good commit message is required!
  - Line 1: Short summary (<70 characters)
  - Line 2: Blank
  - Line 3+: Details.. ; 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.
Reverting Changes

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

• Revert with Caution
  – Will lose all uncommitted changes in the file.
  – If in doubt, grab a backup copy (ZIP your folder) then revert.
    • Just make sure you don’t commit the backup!
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.
Feature Branches
Issues and Branches

• GitLab Issues
  - Used to track feature changes and bugs

• Feature Branch
  - Separate from master branch: allows you to develop and push your code without it going into master.
  - Used for most changes in bigger projects.

• Process
  - Create a GitLab Issue with a branch
  - Checkout branch on your PC.
  - Code, then push changes.
  - Do a GitLab Merge Request
GitLab Workflow
Feature Branch, Merging Changes, Merge Request

Create GitLab Issue.
1. Pull to update.
2. Checkout local feature branch.
Change files, commit, push.

1. Commit any changes.
2. Pull to update.
3. Merge Master to Feature Branch.
4. Resolve conflicts.
5. Build and test.
6. Commit/push changes.
7. Switch to Master branch.

GitLab closes associated issue
Create & accept Merge Request

1. Pull.
2. Remove local feature branch.

1. Assign issue to self
2. Create feature branch

Change files, commit, push.

Legend
In GitLab
In Android Studio
Teammate Feature Branches
Master
My Feature Branch
Create & accept Merge Request

Sequence of Events
Feature Branch Workflow

• Do a Feature Branch
  – GitLab: Create issue
  – GitLab: Create branch
  – IntelliJ: Pull / switch to branch
  – IntelliJ: Code, add-commit-push (repeat!)
  – IntelliJ: Merge master to branch; push
  – GitLab: Merge Request

See videos on GitLab Workshop Page for more!