Principles of Software Construction: Objects, Design, and Concurrency

Software engineering in practice

Git, software development workflows, and monorepos

Josh Bloch    Charlie Garrod    Darya Melicher
Administrivia

- Homework 6 due next Wednesday
  - Checkpoint deadline Monday night
Key concepts from Tuesday
Compare to the Facebook release cycle
DevOps: Development / Operations
Configuration management (CM)

• Definition (Pressman): Configuration management “is a set of tracking and control activities that are initiated when a software engineering projects begins and terminates when software is taken out of operation.”
SVN (left) vs. Git (right)

- SVN stores changes to a base version of each file
- Version numbers (1, 2, 3, ...) are increased by one after each commit
- Git stores each version as a snapshot
- If files have not changed, only a link to the previous file is stored
- Each version is referred by the SHA-1 hash of the contents

A brief Git history...

Linux kernel source tree

- torvalds / linux

- 797,570 commits
- 1 branch
- 582 releases
- ∞ contributors

Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

Latest commit 60b5482 17 hours ago

- torvalds Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net...
- Documentation 21 hours ago
- LICENCES a month ago
- arch 17 hours ago
- block 15 days ago

...
Today

- Practical Git
- Common workflows using Git
- Developing at scale
git commit

Graphics by https://learngitbranching.js.org
`git branch newImage`
git commit
git checkout newImage; git commit
Activity: Make a new branch named bugFix and switch to that branch
Three ways to move work around between branches

1) git merge bugFix (into master)
git checkout bugfix; git merge master (into bugFix)
Move work from bugFix directly onto master

2) git rebase master
But master hasn't been updated, so:

```
git checkout master; git rebase bugFix
```
Copy a series of commits below current location

3) `git cherry-pick C2 C4`
Activity:
Ways to undo work (1)

```
git reset HEAD~1
```

HEAD is the symbolic name for the currently checked out commit
Ways to undo work (2)

**git revert HEAD**

**git reset does not work for remote branches**
Activity:
Highly recommended

Today

- Practical Git
- Common workflows using Git
- Developing at scale
1. Centralized workflow

- Central repository to serve as the single point-of-entry for all changes to the project
- Default development branch is called master
  - all changes are committed into master
  - doesn’t require any other branches
2. Git feature branch workflow

- All feature development should take place in a dedicated branch instead of the master branch
- Multiple developers can work on a particular feature without disturbing the main codebase
  - master branch will never contain broken code (enables CI)
  - Enables pull requests (code review)
3. GitFlow workflow

- Strict branching model designed around the project release
- Uses two+ branches
  - master stores the official release history; tag all commits in the master branch with a version number
  - develop serves as an integration branch for features
GitFlow feature branches (from develop)
GitFlow release branches (eventually into master)
GitFlow hotfix branches
Today

- Practical Git
- Common workflows using Git
- Developing at scale
Pre-2017 release management model at Facebook
Diff lifecycle: local testing
Diff lifecycle: CI testing (data center)

App and Build Configuration Matrix

<table>
<thead>
<tr>
<th></th>
<th>Facebook</th>
<th>Messenger</th>
<th>Groups</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>arm</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>x86</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Diff lifecycle: diff ends up on master
Release every two weeks
Quasi-continuous web release

- Release
- Test
- Build
- Deploy
Quasi-continuous push from master (1,000+ devs, 1,000 diffs/day); 10 pushes/day
Google: similar story, huge code base

**Google repository statistics**
As of Jan 2015

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of files*</td>
<td>1 billion</td>
</tr>
<tr>
<td>Number of source files</td>
<td>9 million</td>
</tr>
<tr>
<td>Lines of code</td>
<td>2 billion</td>
</tr>
<tr>
<td>Depth of history</td>
<td>35 million commits</td>
</tr>
<tr>
<td>Size of content</td>
<td>86 terabytes</td>
</tr>
<tr>
<td>Commits per workday</td>
<td>45 thousand</td>
</tr>
</tbody>
</table>

*The total number of files includes source files copied into release branches, files that are deleted at the latest revision, configuration files, documentation, and supporting data files.
Exponential growth?

Millions of changes committed (cumulative)

[Graph showing exponential growth from 2000 to 2015]
Google Speed and Scale

- >30,000 developers in 40+ offices
- 13,000+ projects under active development
- 30k submissions per day (1 every 3 seconds)

- All builds from source
- 30+ sustained code changes per minute with 90+ peaks
- 50% of code changes monthly
- 150+ million test cases / day, > 150 years of test / day
- Supports continuous deployment for all Google teams!

2016 numbers
Google code base vs. Linux kernel code base

Some perspective

Linux kernel
- 15 million lines of code in 40 thousand files (total)

Google repository
- 15 million lines of code in 250 thousand files \textit{changed per week, by humans}
- 2 billion lines of code, in 9 million source files (total)
Managing a huge monorepo

- Automated testing...
- Lots of automation...
- Smart tooling...
Version control for a monorepo

• Problem: even git is slow at Facebook scale
  – 1M+ source control commands run per day
  – 100K+ commits per week
Version control for a monorepo

- Use build system's file monitor, Watchman, to see which files have changed \( \rightarrow 5x \) faster “status” command
Version control for a monorepo

• Sparse checkouts $\Rightarrow$ **10x faster clones and pulls**
  – clone and pull download only the commit metadata, omit the files
  – When a user performs an operation that needs the contents of files (such as checkout), download the file contents on demand