CS 312 Software Development

Introduction to DevOps

Deployment: Closing the loop

- *Programs that are never deployed have not fulfilled their purpose. We must deploy!*

- But we must answer:
- Is our application in a working state?
- Do we have the necessary HW/SW resources?
- How do we actually deploy?

Continuous Integration (CI)

- Maintain a single repository
  - *With always deployable branch*
- Automate the Build (Build is a proper noun)
  - *And fix broken builds ASAP*
- The Build should be self testing
- Everyone integrates with master frequently
  - *Small "deltas" facilitate integration and minimize bug surface area*
- Automate deployment
  - *Practice "DevOps" culture*

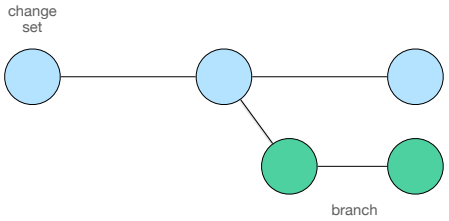
Martin Fowler "Key practices of Continuous Integration"

DevOps

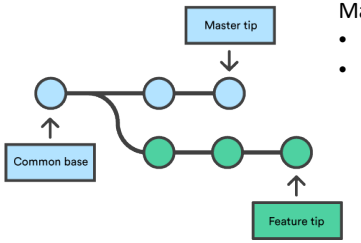
- Involvement of the operations function in each phase of a system's design and development
- Heavy reliance on automation versus human effort
- The application of engineering practices and tools to operations tasks

Version control systems

- git
- Mercurial
- bazaar
- subversion
- csv
- Perforce
- BitKeeper
- etc…



Git workflow for CI

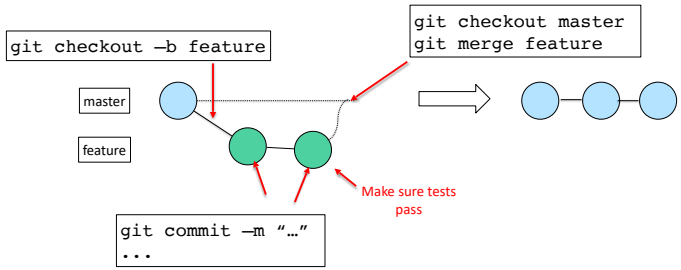


Master is always “deployable”
- Tests pass
- No incomplete features

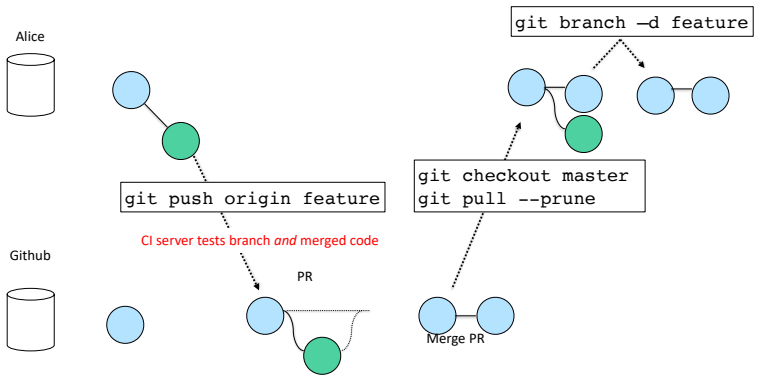
Short-lived branch for single feature

- Branching is cheap in Git
- We will use features branches to segregate changes until integration
- The “master” branch remains deployable

Git “solo” branching workflows



Git/GitHub workflow with CI



Student advice: Branch-per-feature

- "Aggressive branch-per-feature minimized merge conflicts"
- "With this many people you NEED branch-per-feature to avoid stepping on each other"

Our goal is to work efficiently as a project team. *Practice now the processes you will need in your project!*

Adapted from Berkeley CS169