Git for Grown-ups

Emma Jane Westby

Twitter: emmajanehw
[http://twitter.com/emmajanehw]

slides: https://github.com/emmajane/gitforteams
[https://github.com/emmajane/gitforteams]

Hello! My Name is Emma
[http://en.wikipedia.org/wiki/Emma_Jane_Hogbin]

I have been using version control for 10+ years and had the great misfortune of teaching CVS to arts majors before distributed version control was a thing.

Warning!

This is not a talk about all the commands you can run in Git.

Resources for Commands:

- Introduction to Git
  [http://drupalize.me/series/introduction-git-series]
  video lessons
- Git Documentation
  [http://git-scm.com/doc]
- Pro Git
  [http://git-scm.com/book]
- ungit
  [https://github.com/FredrikNoren/ungit]
  web UI / visualization tool

More Warning!

This talk is kind of about how Joe and I decided to incorporate rebasing into our workflow.

For the record: I still think rebasing is fundamentally wrong.

And I have the mic.

Final Warning!

Git makes me angry inside. You can read why [http://24ways.org/2013/git-for-grownups/] ...if you want.

tl;dr: it is arrogant, poorly documented software.

How We Normally Teach Tech Topics

"Here's a list of 23893467 commands!
You should memorize use them!"

We tend to lead with the code. This is not adult education best practices. This is not best practices for adult education. "Andragogy" tells us adults want information which is relevant to their job, and immediately actionable. Adult learners are selfish.

My Goal for this Presentation

By the end of this session you should be able to:

- Determine a permission strategy for your project.
- Determine a branching strategy for your project.
- Create documentation which outlines how your team members will use version control.

We are going to examine these three concepts throughout this presentation.

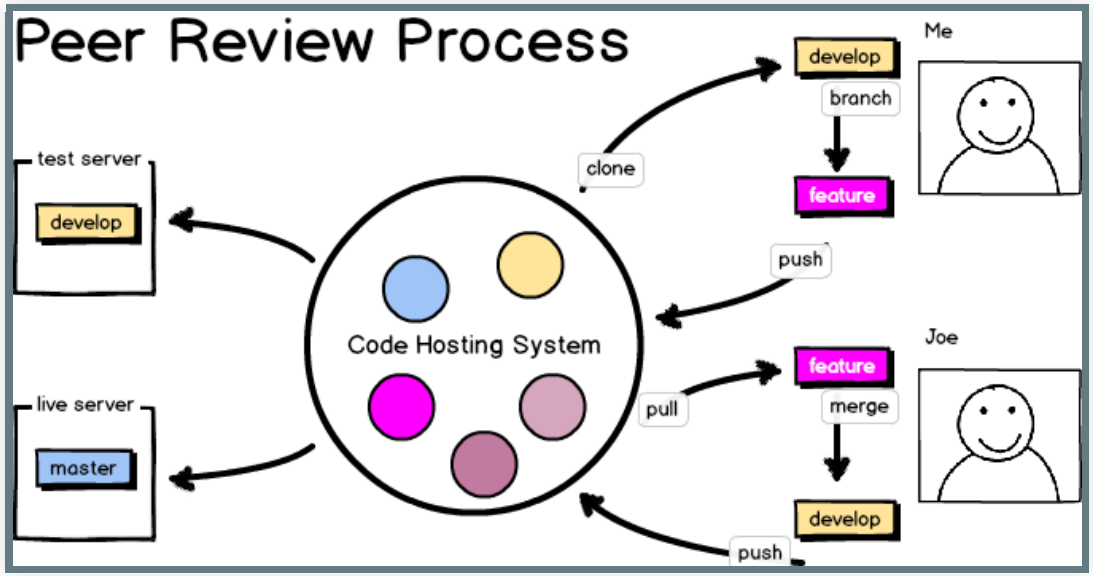
Workflow Solves Hard Problems with Trivial Questions

- Who has permission to commit code?
- At what point(s) does the code need to be reviewed and approved?
- How often do you deploy code?

Assumption Alert!

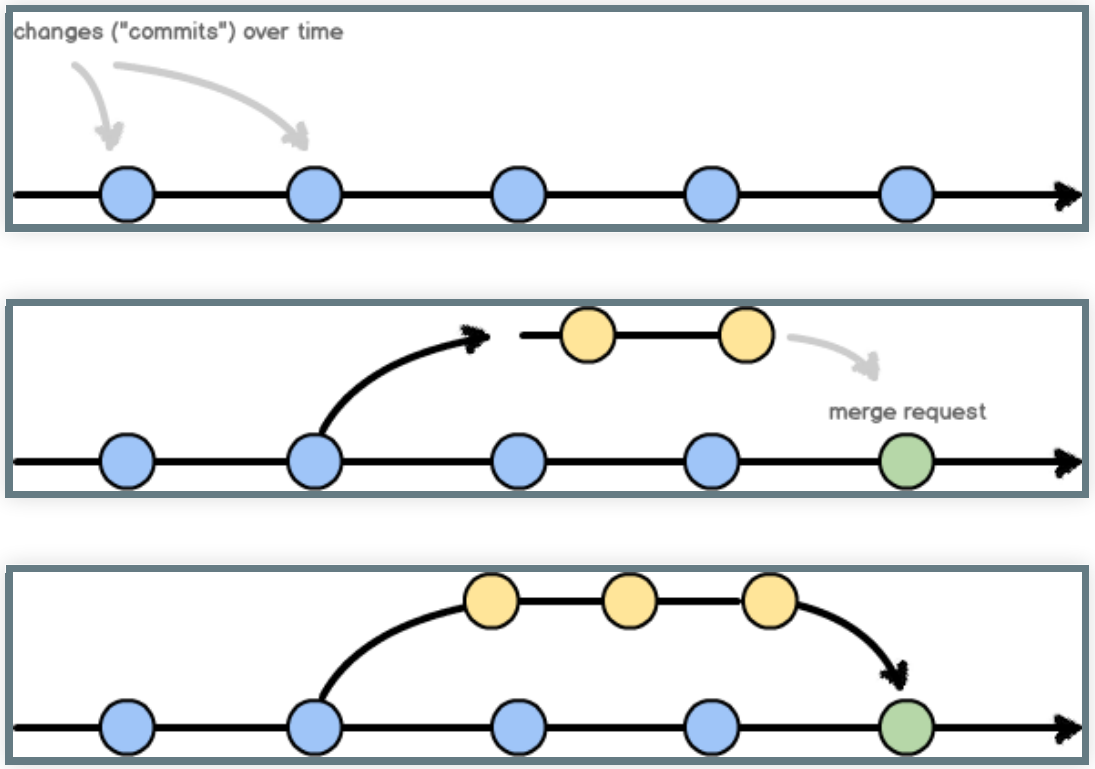
1. Inconsistency leads to mistakes.
2. Anything that is arbitrary must follow a convention.
3. Conventions should be documented.
4. Only tested code is deployed.
5. Faster is better.

Workflow =
Actions + Locations + Permissions



this is where we want to end up by the end of today. You know where each branch lives. You know how / where a branch is closed.

Permission Strategies



Centralized: everyone works in master from the same disk. **Branching** how we work. Anyone can check into master. **Forking** how most FOSS projects work; also for CI where the testbot gets final approval into master.

Patching

Everyone has read access. Very few have write access. Suggested changes are presented as a patch file for review.

- Pro: Forces a review process.
- Con: Patches need to be rerolled to stay up-to-date.
- Example: Drupal

Forking

Project forks give full permissions to developers so they can do work. New work is added to the main project through a request to upstream project.

- Pro: Forces a review process.
- Pro: "Modern" way of doing patches.
- Pro: Encourages experimentation (dev controls their own project clone)
- Example: joind.in

Branching

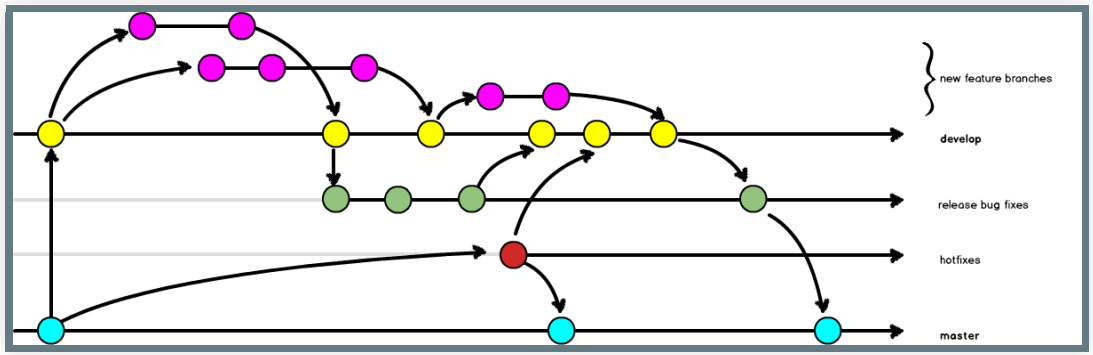
- Pro: Ensures clean/working master (good for CI)
- Pro: Encourages experimentation (cheap to branch)
- Pro: Reduces overhead of forking workflow
- Con: Encourages code review (does not require)
- Example: your internal project (probably)

Branching Strategies

- Scheduled Release: Gitflow [http://nvie.com/posts/a-successful-git-branching-model/] or Simplified Gitflow [http://drewfradette.ca/a-simpler-successful-git-branching-model/]
- Continuous Deployment: Branch Per Feature [https://www.acquia.com/blog/pragmatic-guide-branch-feature-git-branching-strategy] or GitHub Flow [http://scottchacon.com/2011/08/31/github-flow.html]
- Hybrid: Squash Workflow [http://reinh.com/blog/2009/03/02/a-git-workflow-for-agile-teams.html]

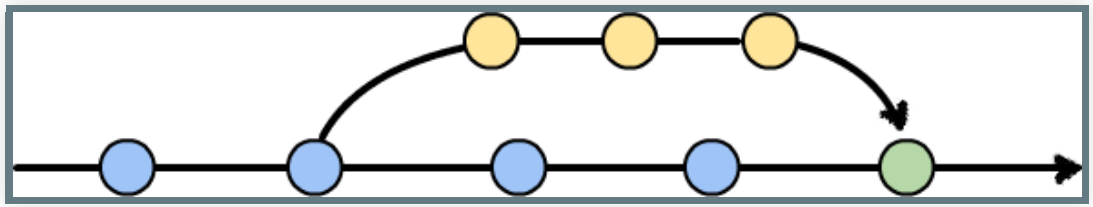
Scheduled Release

- Incorporates human-reviews, and possibly automated tests.
- Allows you to collate many smaller changes into a single release.



Continuous Deployment

- Code is deployed faster than scheduled releases.
- Requires (trusted) test coverage.
- Typically uses a mechanical gatekeeper to check in code to the master branch.
- Fewer branches to maintain / keep updated.

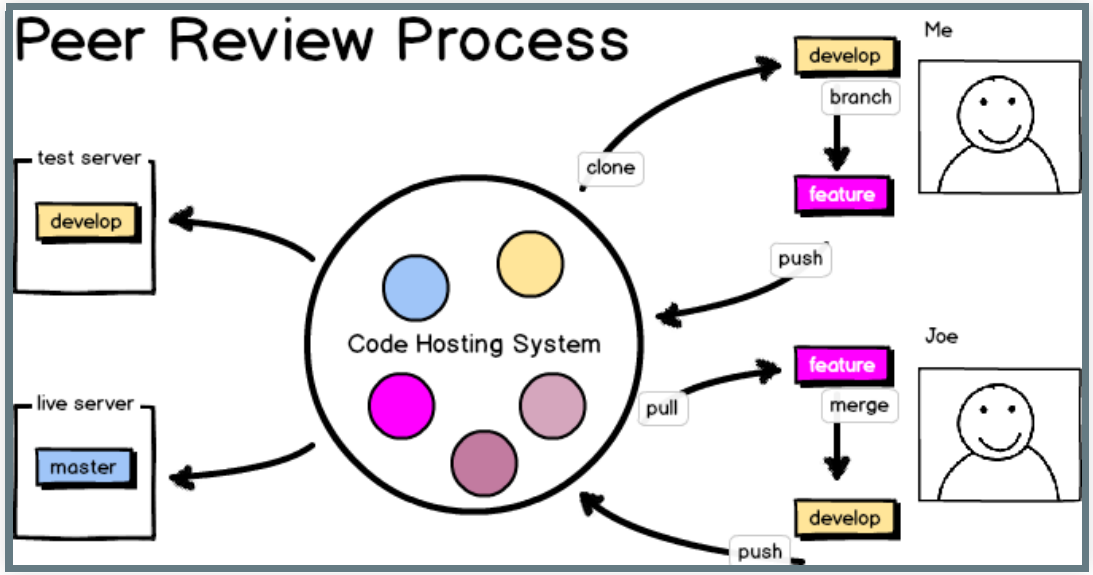


Example Worklows

- These examples are pulled from Drupalize.Me.
- This is a product with no external stakeholders.
- YMMV, YOLO, etc.

these are both in the resources for the repository

How We Work



this is the star wars sprintflow. There are more layers for the WP workflow.

Note on Naming

- Use terms which resonate with your team (MVP -> LBB).
- Giving a descriptive name to projects and processes allows you to change the meaning by changing the name.
- There are a lot of Ewoks.
- There are more My Little Ponies.

During the Upgrade

- Drupal 6 -> Drupal 7 upgrade
- Aiming for speed of code work, not stability.
- Changes were not being deployed to the live server.
- Total time: 18 months.
- Star Wars Sprintflow [../../resources/workflow-sample-starwars.md]

Post-Launch

- Aiming for stability first, speed second.
- We do not have complete test coverage.
- Changes are collated weekly onto a QA server, and deployed from there.
- Whispering Pines Weekly Workflow [../../resources/workflow-sample-whisperingpines-code.md]
- Release philosophy [../../resources/workflow-sample-whisperingpines-releasecycle.md]
- Deployment [../../resources/workflow-sample-whisperingpines-deployment.md]

Resources

- Managing Chaos: Digital Governance by Design [http://www.rosenfeldmedia.com/books/web-governance/]
- Workflows and Permissions Strategies [https://www.atlassian.com/git/workflows]
- Scheduled Release: Gitflow [http://nvie.com/posts/a-successful-git-branching-model/] (Cheatsheet [http://danielkummer.github.io/git-flow-cheatsheet/]) or Simplified Gitflow [http://drewfradette.ca/a-simpler-successful-git-branching-model/]
- Continuous Deployment: Branch Per Feature [https://www.acquia.com/blog/pragmatic-guide-branch-feature-git-branching-strategy] or GitHub Flow [http://scottchacon.com/2011/08/31/github-flow.html]
- Hybrid: Squash Workflow [http://reinh.com/blog/2009/03/02/a-git-workflow-for-agile-teams.html]


WORKFLOW?
GIT DON'T CARE
memegenerator.net

Thanks!

Feedback Welcome!

@emmajanehw
[http://twitter.com/emmajanehw]

https://github.com/emmajane/gitforteams
[https://github.com/emmajane/gitforteams]