Git Workflow and Branching Strategy

- Git Review
- Merge vs. Rebase (plus 'git pull –rebase')
- My workflow and recommendation
- Branching Recommendations
- Strategies / Discussion
- Cool Tools / Tips

Git Review

- ## Why use SCM? Why use Git?

http://sandofsky.com/blog/git-workflow.html

The first is to help the act of writing code.

You need to sync changes with teammates, and regularly back up your work. Emailing zip files doesn't scale.

The second reason is configuration management.

This includes managing parallel lines of development, such as working on the next major version while applying the occasional bug fix to the existing version in production. Configuration management is also used to figure out when exactly something changed, an invaluable tool in diagnosing bugs.

Traditionally, these two reasons conflict.

When prototyping a feature, you should make regular checkpoint commits. However, these commits usually break the build.

In a perfect world, every change in your revision history is succinct and stable. There are no checkpoint commits that create line noise. There are no giant, 10,000 line commits. A clean history makes it easy to revert changes or cherry-pick them between branches. A clean history is easy to later inspect and analyze. However, maintaining a clean history would mean waiting to check in changes until they're perfect.

So which approach do you choose? Regular commits, or a clean history?

If you're hacking on a two man pre-launch startup, clean history buys you little. You can get away with committing everything to Master, and deploying whenever you feel like it.

As the consequences of change increase, be it a growing development team or the size of your user base, you need tools and techniques to keep things in check. This includes automated tests, code review, and a clean history.

Feature branches seem like a happy middle ground. They solve the basic problems of parallel development. You're thinking of integration at the least important time, when you're writing the code, but it will get you by for some time.

When your project scales large enough, the simple branch/commit/merge workflow falls apart.

The time for duct-tape is over. You need a clean revision history.

Merge vs. Rebase

- http://gitready.com/intermediate/2009/01/31/intro-to-rebase.html
- http://progit.org/book/ch3-6.html
- http://darwinweb.net/articles/the-case-for-git-rebase
- http://blog.woobling.org/2009/05/git-rebase-considered-awesome.html
- The man himself: Linus Torvalds: http://kerneltrap.org/Linux/Git_Management

  As long as you never _ever_ expose your rewriting of history to anybody else, people won't notice or care, because you basically guarantee that nobody can ever see all those _other_ "simpler" histories, and they only see the one final result. That's why 'rebase' is useful for private histories.

- This quote helped me immensely to understand "why" to use rebase:

  What rebase really says is to consider the changes that are about to be made to the branch (i.e. master)... FIRST get the changes that have already been COMMITTED to the branch, and then re-apply MY changes on top of that, irrespective of when the changes were made. So prior to merging to master, my changes are ON TOP of the latest of master.

- git pull –rebase vs. git pull: (remember: git pull = git fetch; git merge)...

Git pull –rebase: pros/cons

Argument against git pull --rebase:

Branches and merges do contain valuable information: they record the flow used to develop the software, they clearly state that commit x depends on commit y and typically records the exact version which has been used by the developer to write commit x. This is a huge advantage over subversion and the like, which pretend in their history that the software was always written with absolutely no parallelism.

When you rebase or cherry pick, you lose that valuable information. This can be OK in some very specific situations, when this is the result of a voluntary deliberate non-default action. But rebasing by default when pulling... please just don't advise that to innocent developers. I admit this might work and present some advantages in some very specific environments, like different peoples working on very different components with little and well defined interactions between said components, and everybody knowing very precisely what others are doing. But this situation is not the general case. It's even very far from what I would consider a typical situation in the software industry.

And the argument against this argument:

you all are saying it contains information. I disagree... when all you are doing is updating your remote tree a merge contains absolutely no information. and if you've patched it and you aren't rebasing you've made a merge more required to pull into the mainline. merges are good when they contain useful information... like a deliberate fork of the code for a feature, or fix, not just when you are doing your daily update from the remote repositories. although git is distributed there isn't a need to actually treat everyone's code base like it is there own private fork... better to treat it as a base off the mainline. I like to see intentional patches and merging... e.g. cherry-pick, rebase, and merge are good. if you do it too much those merges just becomes noise.

Rebase vs. Merge – A Picture

- Before:

```
* b2be05e 2012-03-27 | deleted remaining French references. Bon Voyage! [Z[redacted]] [Dan [redacted]]
* 58b8ac3 2012-03-27 | Ensure alerts in the event_chain (called from workers) are internationalized [#[redacted]] [Scott [redacted]]
* 5945005 2012-03-27 | Remove French as a supported translation and locale. Updated specs to use PT instead of FR [Z[redacted]] [Dan [redacted]]
* 93a03b7 2012-03-27 | Ensure that alerts called in workers are internationalized with locale from account. [#[redacted]] [Scott [redacted]]
* 4063edc 2012-03-27 | put resque connection management back in an after_fork callback [Eric [redacted]]
* acbd8e0 2012-03-27 | fixed bug in thermostat daily processor worker [Eric [redacted]]
* 1164ed1 2012-03-27 | White space cleanup [Jim [redacted]]
```

- After:

```
* ae0332a 2012-03-27 | deleted remaining French references. Bon Voyage! [Z[redacted]] [Dan [redacted]]
*   bddd7fe 2012-03-27 | Merge branch 'release-2.2' of m[redacted].sc[redacted].com:s[redacted]l into release-2.2 [Duane [redacted]]
|\
| * 58b8ac3 2012-03-27 | Ensure alerts in the event_chain (called from workers) are internationalized [#[redacted]] [Scott [redacted]]
* | df783ae 2012-03-27 | Javascript fix in translation of fan mode) [Duane [redacted]]
|/
* 5945005 2012-03-27 | Remove French as a supported translation and locale. Updated specs to use PT instead of FR [Z[redacted]] [Dan [redacted]]
* 93a03b7 2012-03-27 | Ensure that alerts called in workers are internationalized with locale from account. [#[redacted]] [Scott [redacted]]
* 4063edc 2012-03-27 | put resque connection management back in an after_fork callback [Eric [redacted]]
* acbd8e0 2012-03-27 | fixed bug in thermostat daily processor worker [Eric [redacted]]
* 1164ed1 2012-03-27 | White space cleanup [Jim [redacted]]
```

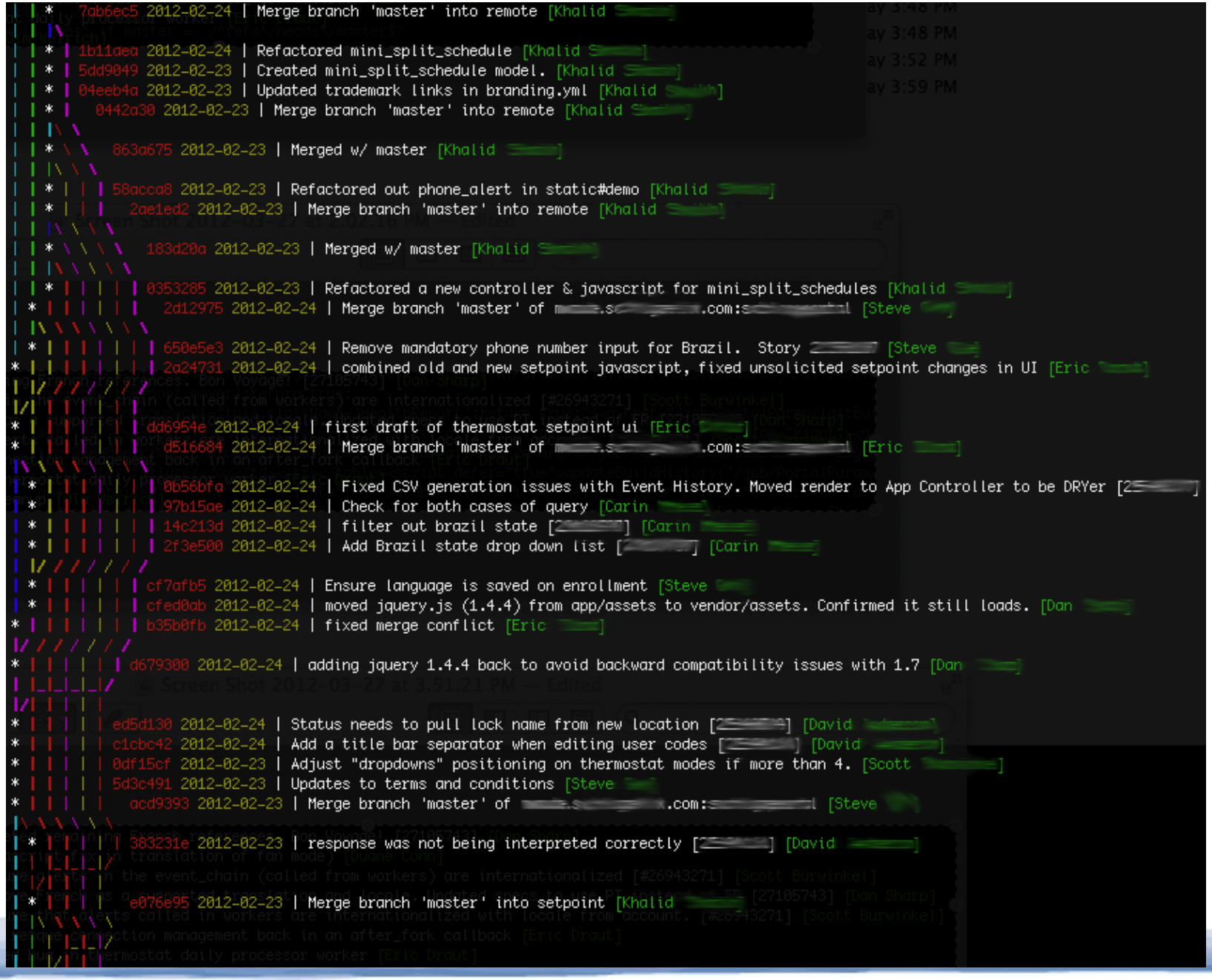
- Better:

```
* 415288b 2012-03-27 | deleted remaining French references. Bon Voyage! [Z[redacted]] [Dan [redacted]]
* 0612ac3 2012-03-27 | Javascript fix in translation of fan mode) [Duane [redacted]]
* 4974f92 2012-03-27 | Ensure alerts in the event_chain (called from workers) are internationalized [[redacted]] [Scott [redacted]]
* 5945005 2012-03-27 | Remove French as a supported translation and locale. Updated specs to use PT instead of FR [Z[redacted]] [Dan [redacted]]
* 93a03b7 2012-03-27 | Ensure that alerts called in workers are internationalized with locale from account. [[redacted]] [Scott [redacted]]
* 4063edc 2012-03-27 | put resque connection management back in an after_fork callback [Eric [redacted]]
* acbd8e0 2012-03-27 | fixed bug in thermostat daily processor worker [Eric [redacted]]
* 1164ed1 2012-03-27 | White space cleanup [Jim [redacted]]
```

But Dan...

- That's not so bad, right?
- Plus, it's good to know that Duane was on a branch...
  - (except he wasn't!)
- And it's good to know that his edit happened before Scott's...
  - (except it may not have!)
  - (plus, why does that matter? Only the time it landed on master should count)
- And, it's only a little blip, right?
  - er....

Ick!


* 7ab6ec5 2012-02-24 | Merge branch 'master' into remote [Khalid
* 1b11aea 2012-02-24 | Refactored mini_split_schedule [Khalid
* 5dd9049 2012-02-23 | Created mini_split_schedule model. [Khalid
* 04eeb4a 2012-02-23 | Updated trademark links in branding.yml [Khalid
* 0442a30 2012-02-23 | Merge branch 'master' into remote [Khalid
* 863a675 2012-02-23 | Merged w/ master [Khalid
* 58acca8 2012-02-23 | Refactored out phone_alert in static#demo [Khalid
* 2ae1ed2 2012-02-23 | Merge branch 'master' into remote [Khalid
* 183d20a 2012-02-23 | Merged w/ master [Khalid
* 0353285 2012-02-23 | Refactored a new controller & javascript for mini_split_schedules [Khalid
* 2d12975 2012-02-24 | Merge branch 'master' of
[Steve
* 650e5e3 2012-02-24 | Remove mandatory phone number input for Brazil. Story
[Steve
* 2a24731 2012-02-24 | combined old and new setpoint javascript, fixed unsolicited setpoint changes in UI [Eric
* dd6954e 2012-02-24 | first draft of thermostat setpoint ui [Eric
* d516684 2012-02-24 | Merge branch 'master' of
[Eric
* 0b56bfa 2012-02-24 | Fixed CSV generation issues with Event History. Moved render to App Controller to be DRYer [2
* 97b15ae 2012-02-24 | Check for both cases of query [Carin
* 14c213d 2012-02-24 | filter out brazil state [
] [Carin
* 2f3e500 2012-02-24 | Add Brazil state drop down list [
] [Carin
* cf7afb5 2012-02-24 | Ensure language is saved on enrollment [Steve
* cfed0ab 2012-02-24 | moved jquery.js (1.4.4) from app/assets to vendor/assets. Confirmed it still loads. [Dan
* b35b0fb 2012-02-24 | fixed merge conflict [Eric
* d679300 2012-02-24 | adding jquery 1.4.4 back to avoid backward compatibility issues with 1.7 [Dan
* ed5d130 2012-02-24 | Status needs to pull lock name from new location [
] [David
* c1cbc42 2012-02-24 | Add a title bar separator when editing user codes [
] [David
* 0df15cf 2012-02-23 | Adjust "dropdowns" positioning on thermostat modes if more than 4. [Scott
* 5d3c491 2012-02-23 | Updates to terms and conditions [Steve
* acd9393 2012-02-23 | Merge branch 'master' of
[Steve
* 383231e 2012-02-23 | response was not being interpreted correctly [
] [David
* e076e95 2012-02-23 | Merge branch 'master' into setpoint [Khalid

My Git Workflow

Basically taken from: http://gitguru.com/2009/02/03/rebase-v-merge-in-git/

- Create new branch B from existing branch A
- Add/commit changes on branch B
- Rebase updates from branch A
- Merge changes from branch B onto branch A

http://ariejan.net/2009/06/08/best-practice-the-git-development-cycle

http://www.randyfay.com/node/91

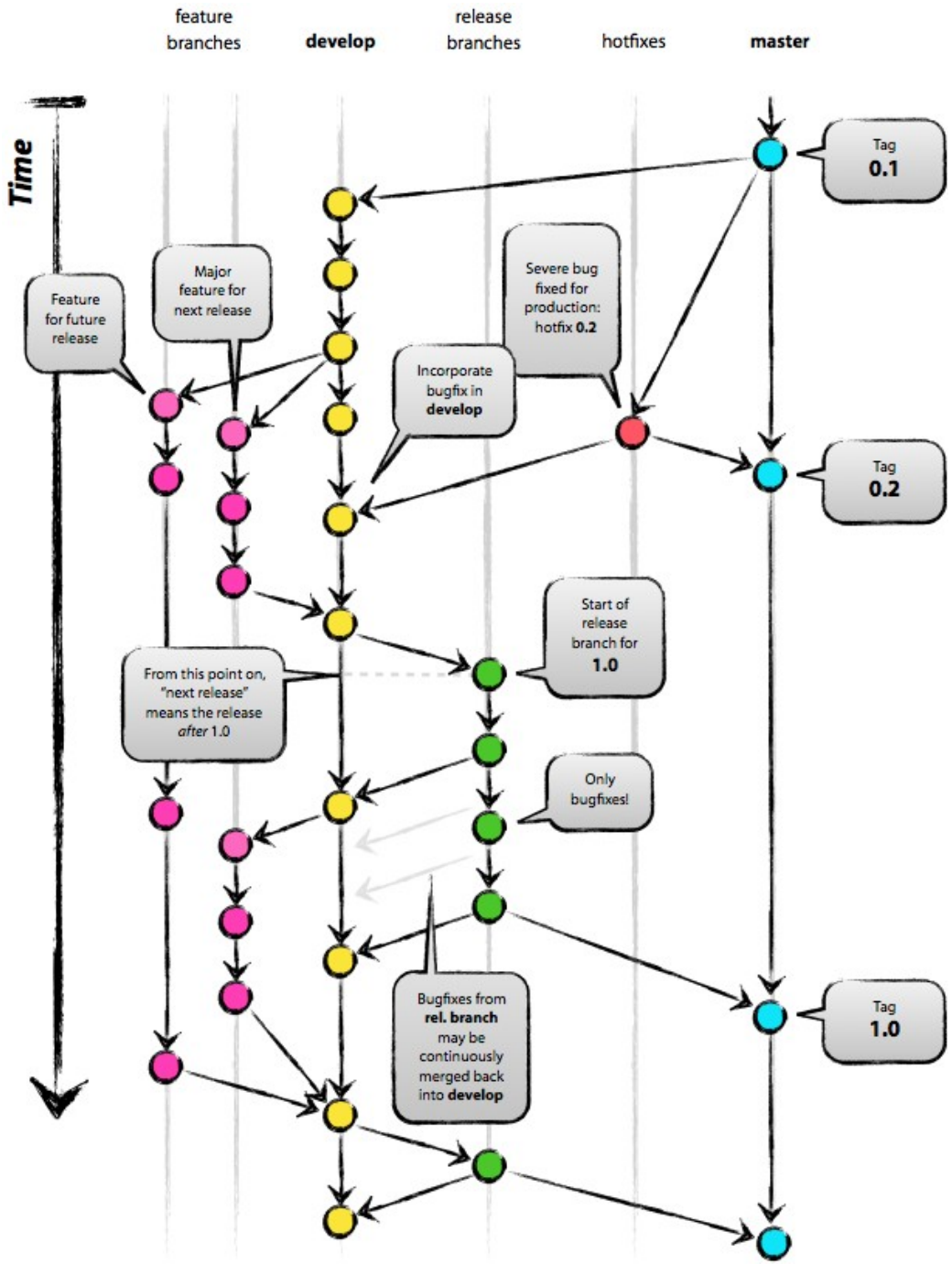
http://stackoverflow.com/questions/457927/git-workflow-and-rebase-vs-merge-questions

Git / Branch Recommendations

- Always use a local branch for local work. **Not master**!
- Name the branch so you know what it is for:
  - dan_refactor_transaction_locking
- If you may need to deploy it or share it, prefix “name_”
- If it is long-lived or shared, **don't rebase**!
- If it is short-lived and private, **rebase**!
- Don't use 'git pull'. Use 'git pull –rebase' Why?
  'Merge branch 'master' of xxx.yyy.com:zzzzzz'
  ^^ wasted message and useless commit!
  Also: http://randyfay.com/node/103
- When done, merge it into master:
  git checkout master
  git merge dan_refactor_transaction_locking
  git push origin master
- All this == Clean History! == Happy Panda!

Strategies / Discussions

- Longer lived branches?
  - development + master vs. just master?
  - Pros: better deploy/release control
  - Cons: more merges. Requires “gatekeeper”
- Feature branches? Rebase? Dangers
- Spikes? When? Why? Naming?
- Release branches? Role of Master?
- Tagging? Cleaning up?


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Tips / Tricks / Tools

- git merge –no-ff (always create a merge commit)
- git rebase -p (preserve merges during rebase)
- git rerere (**re**member **re**base **re**solutions)
- HEAD, HEAD~#, HEAD^
- rewriting private history:
  - git rebase -i
  - git merge –squash
  - git reset –soft
  - git cherry-pick
- git bisect (binary tree good/bad resolution)
- **Beware!**
  - git push –force
  - git rebase, or git rebase -i

More Tips / Tricks / Tools

- My tools: resync, clean, push, pushf, glog
- Git-Smart: https://github.com/geelen/git-smart
- Git Flow: http://jeffkreeftmeijer.com/2010/why-arent-you-using-git-flow/
- Git Rake: http://blog.bitfluent.com/post/27983389/git-utilities-you-cant-live-without
- Git Up: https://github.com/aanand/git-up
- Gup: http://jasoncodes.com/posts/gup-git-rebase
- Git: the chainsaw of software development!