Trunk Based Development

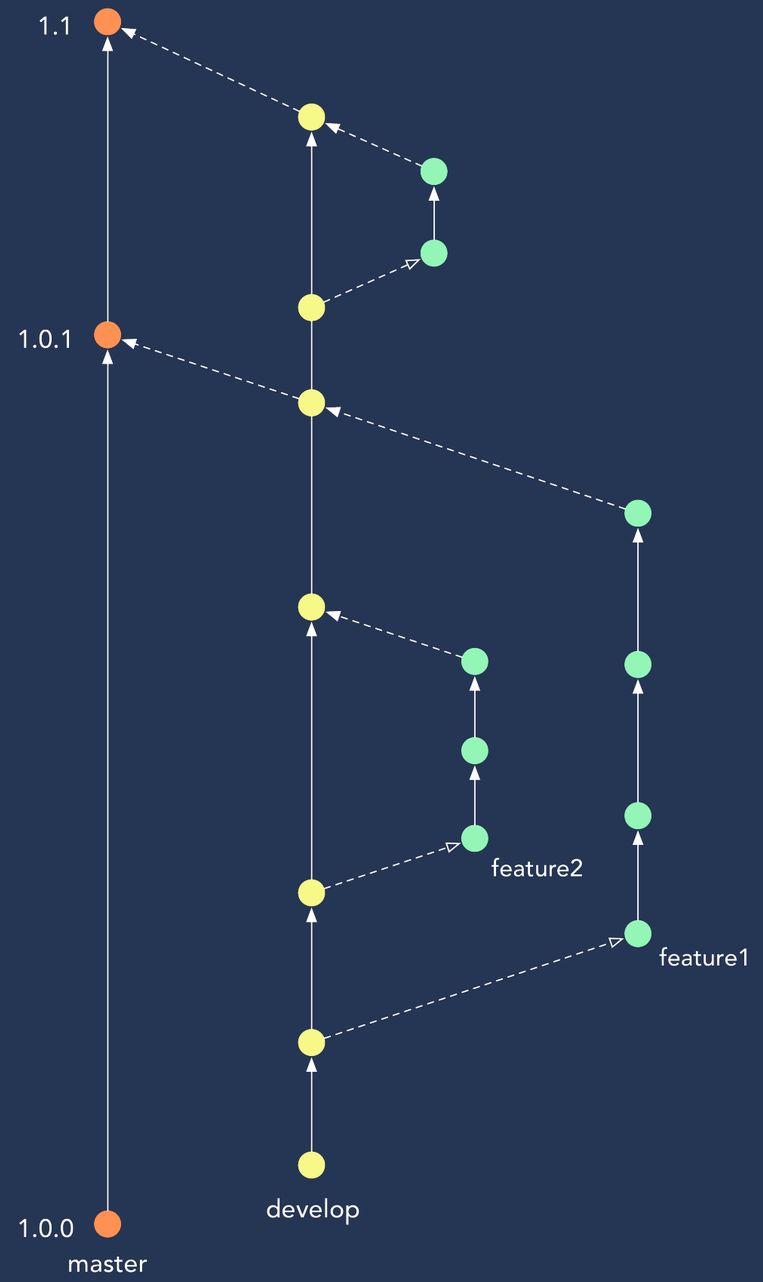
Because no one likes to merge

I am

- **@sebreh** (Sebastian Rehnby)
- Managing the **mobile team at Podio**
- 3.5 years with **Podio + Citrix**
- **Swedish**

Who has **heard of TBD?**

Lets start with something familiar



Lets build a feature

```
$ git checkout -b booster-rockets
```

Lets build a feature

```
$ git checkout -b booster-rockets
```

```
$ git commit -m "WIP just trying something out"
```

Lets build a feature

```
$ git checkout -b booster-rockets

$ git commit -m "WIP just trying something out"

$ git commit -m "WIP fixed a few tests"
```

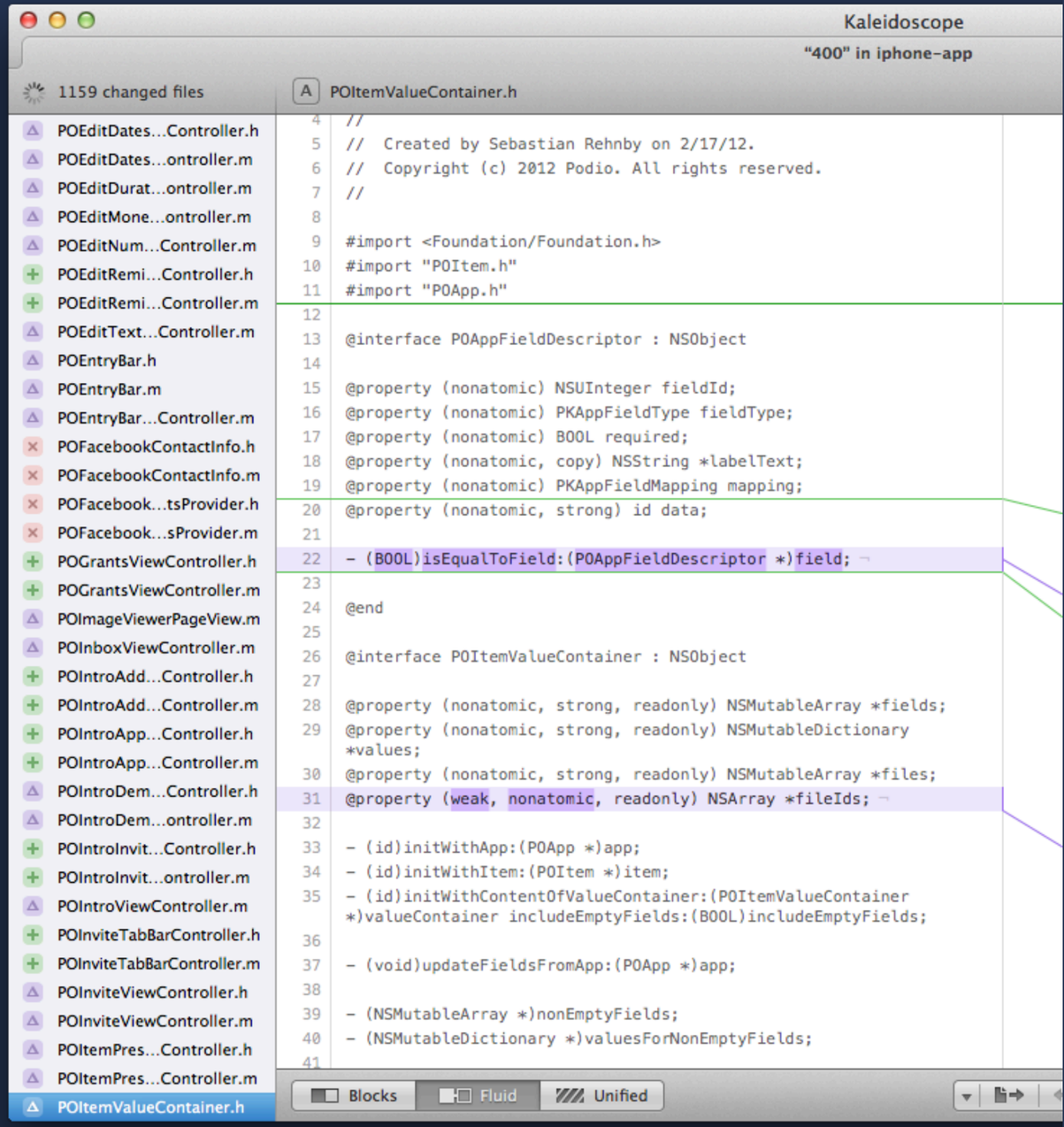
Lets build a feature

```
$ git checkout -b booster-rockets

$ git commit -m "WIP just trying something out"

$ git commit -m "WIP fixed a few tests"

$ git commit -m "Tests pass, yay"
```

Let's ship it

```
$ git merge develop
```

Hmmm...


Kaleidoscope
"400" in iphone-app
1159 changed files
POItemValueContainer.h
POEditDates...Controller.h
POEditDates...ontroller.m
POEditDurat...ontroller.m
POEditMone...ontroller.m
POEditNum...Controller.m
POEditRemi...Controller.h
POEditRemi...Controller.m
POEditText...Controller.m
POEntryBar.h
POEntryBar.m
POEntryBar...Controller.m
POFacebookContactInfo.h
POFacebookContactInfo.m
POFacebook...tsProvider.h
POFacebook...sProvider.m
POGrantsViewController.h
POGrantsViewController.m
POImageViewerPageView.m
POInboxViewController.m
POIntroAdd...Controller.h
POIntroAdd...Controller.m
POIntroApp...Controller.h
POIntroApp...Controller.m
POIntroDem...Controller.h
POIntroDem...ontroller.m
POIntroInvit...Controller.h
POIntroInvit...ontroller.m
POIntroViewController.m
POInviteTabBarController.h
POInviteTabBarController.m
POInviteViewController.h
POInviteViewController.m
POItemPres...Controller.h
POItemPres...Controller.m
POItemValueContainer.h
// Created by Sebastian Rehnby on 2/17/12.
// Copyright (c) 2012 Podio. All rights reserved.
#import <Foundation/Foundation.h>
#import "POItem.h"
#import "POApp.h"
@interface POAppFieldDescriptor : NSObject
@property (nonatomic) NSUInteger fieldId;
@property (nonatomic) PKAppFieldType fieldType;
@property (nonatomic) BOOL required;
@property (nonatomic, copy) NSString *labelText;
@property (nonatomic) PKAppFieldMapping mapping;
@property (nonatomic, strong) id data;
- (BOOL)isEqualToField:(POAppFieldDescriptor *)field;
@end
@interface POItemValueContainer : NSObject
@property (nonatomic, strong, readonly) NSMutableArray *fields;
@property (nonatomic, strong, readonly) NSMutableDictionary *values;
@property (nonatomic, strong, readonly) NSMutableArray *files;
@property (weak, nonatomic, readonly) NSArray *fileIds;
- (id)initWithApp:(POApp *)app;
- (id)initWithItem:(POItem *)item;
- (id)initWithContentOfValueContainer:(POItemValueContainer *)valueContainer includeEmptyFields:(BOOL)includeEmptyFields;
- (void)updateFieldsFromApp:(POApp *)app;
- (NSMutableArray *)nonEmptyFields;
- (NSMutableDictionary *)valuesForNonEmptyFields;
Blocks
Fluid
Unified

Also, this guy

Oh no a bug!

Romain already fixed it in his branch, I will just cherry pick it

```
$ git cherry-pick f4fbd8d8
```

My experience from git-flow like branching

- Long running, **diverging branches**
- Hard to know in **which branch** a certain commit/fix is in
- **Cherry picking** anarchy
- **Merges** from hell
- **Doesn't scale** with team size

Either we suck at it, or it sucks.

I choose to belive the latter.

What is Trunk Based Development

- Everyone develops on **master**
- The master branch should always be **releasable**
- We **branch in code** through the use of **feature flipping** (runtime or compile time)
- We use **feature levels** to control availability

Lets build a feature...again

```
let flipper = Flipper(["landing-gear"    : .Release,
                       "heat-shield"     : .Beta,
                       "booster-rockets" : .Development])
```

Lets build a feature...again

```
let flipper = Flipper(["landing-gear"    : .Release,
                       "heat-shield"     : .Beta,
                       "booster-rockets" : .Development])

...

if flipper.isEnabled("booster-rockets") { // => False for release builds, True for development builds
  // Use the booster rockets
} else {
  // Use whatever old thing
}
```

Lets build a feature...again

```
$ git commit -m "Feature flag for 'booster-rockets'"
```

Lets build a feature...again

```
$ git commit -m "Feature flag for 'booster-rockets'"
$ git pull --rebase
```

Lets build a feature...again

```
$ git commit -m "Feature flag for 'booster-rockets'"
$ git pull --rebase
$ git push origin master
```

Lets build a feature...again

```
$ git commit -m "Feature flag for 'booster-rockets'"
$ git pull --rebase
$ git push origin master

$ git commit -m "Attached booster rockets"
```

Lets build a feature...again

```
$ git commit -m "Feature flag for 'booster-rockets'"
$ git pull --rebase
$ git push origin master

$ git commit -m "Attached booster rockets"
$ git pull --rebase
```

Lets build a feature...again

```
$ git commit -m "Feature flag for 'booster-rockets'"
$ git pull --rebase
$ git push origin master

$ git commit -m "Attached booster rockets"
$ git pull --rebase
$ git push origin master
```

Rinse and Repeat

Many small merges, instead of one big one

TBD in practice

- How to know if a commit is good or bad? **Continous integration** is your friend.
- Invest in **unit and functional testing**. Make it easy to write tests.
- **Most things can be feature flipped** if you just try.

Don't break master

You break it, you fix it

Old code is bad code

Our process

- Central Jenkins server runs unit and functional tests of every commit
- Nightly build for beta/release, pushed to HockeyApp
- Never code older than 3 days on local machine

Why do TBD?

Because Google and Facebook does it!

Seriously...Advantages of TBD

- If something breaks, we know immediately
- Every feature is in the app, always
- Conditional feature availability based on build configuration
- Remote control of features possible
- Code ownership leads to better commits
- It scales!

Force features

```
flipper.flipOn("booster-rockets")

// or...

flipper.flipOff("booster-rockets")
```

Force features

- Staged rollouts
- A/B testing

Challenges of TBD

- **Uncomfortable at first** a.k.a. I-will-just-make-a-small-branch-for-this-syndrome. Trust in TBD, and branching will start to feel weird.
- **Stability of master** Risk of regressions. Invest in automated testing. Branching is just a false sense of security, still needs to be integrated at a later point. Possibility to use release branches.
- **Code reviews** (There are tools available)

Bla bla...Does it work?

- 25(ish) devs doing it (iOS, Android, frontend and API)
- We deploy our website every day
- We release the apps every 3 weeks from master branch. No release branches.
- Very little time, if any, is spent on integrating changes.
- I couldn't imagine going back.

Try it!

github.com/sebreh/Flipper

More reading...

- *What is Trunk Based Development* by Paul Hammant
- *Feature Toggles* by Martin Fowler

That is all