ThoughtWorks® IT大咖说

Feature Branching

To branch or not to branch, that's the question!

Benoit Del Basso

Thoughtworks Open Party 武汉 2017.7.1

Hello, I'm 小贝!

CTO @ My Little Paris

2012-2017

A newsletter for women, e-commerce box,
« good deal » apps and many many more…

Growth from 8 people to a media group
of 150 people / 30 M€ annual turnover

Business model: native advertising

Go French Yourself

ThoughtWorks | IT大咖说 知识分享平台

Why am I in Wuhan?

In 💑 with a 武汉人

Six-month break to get my kid learning Chinese

Meanwhile enjoying 热干面 and working on my new startup https://devflow.io

So, let's talk about

Feature Branching ?

Feature Branching ?

A strategy about the flow of commits

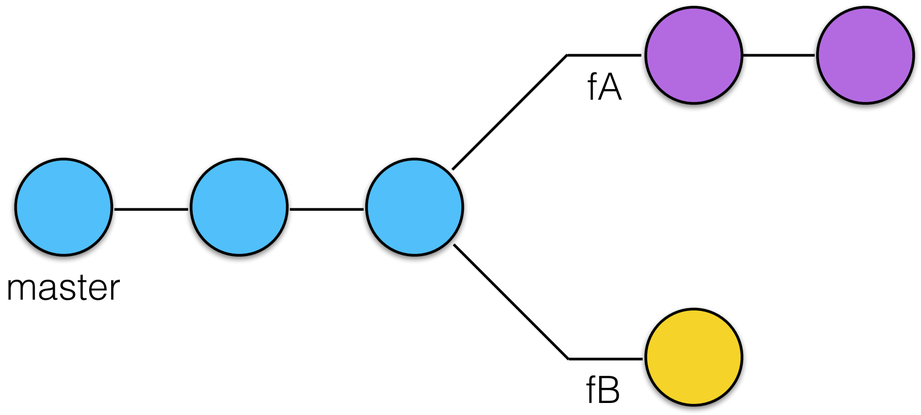
Often controversial
(有争议)

Feature Branching ?

An idea, something that brings
value to the users

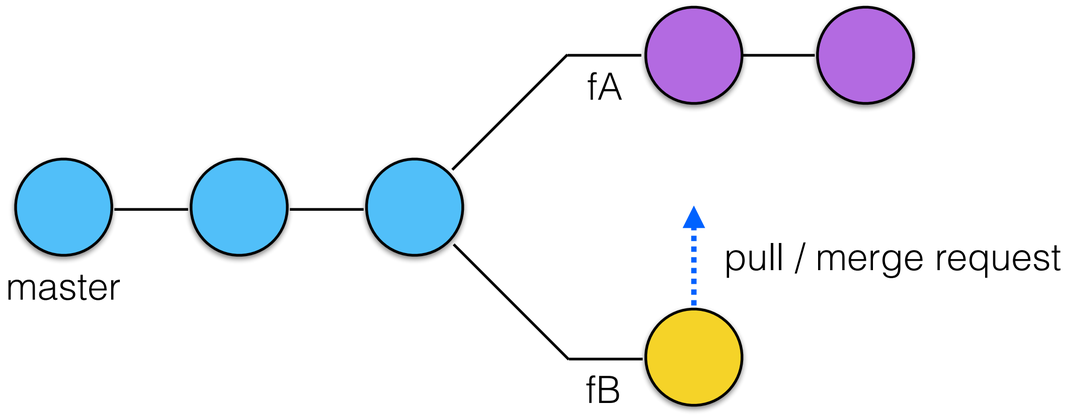
Feature Branching

Creating a branch for every feature you work on



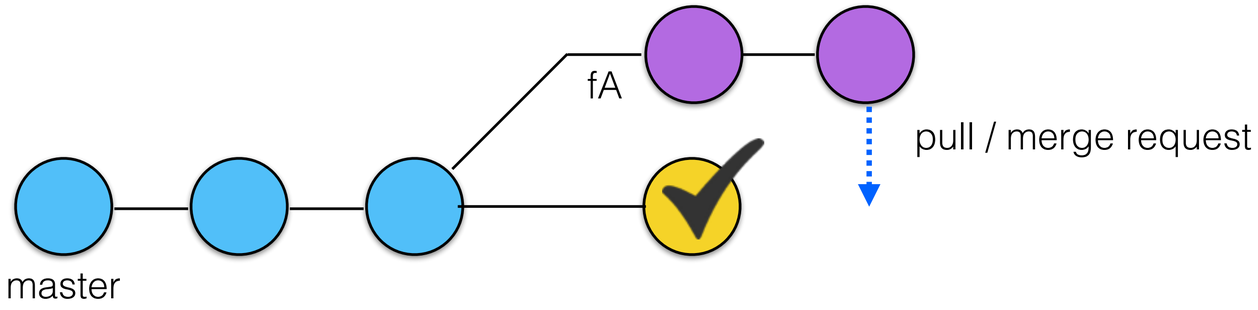
Feature Branching

Creating a branch for every feature you work on



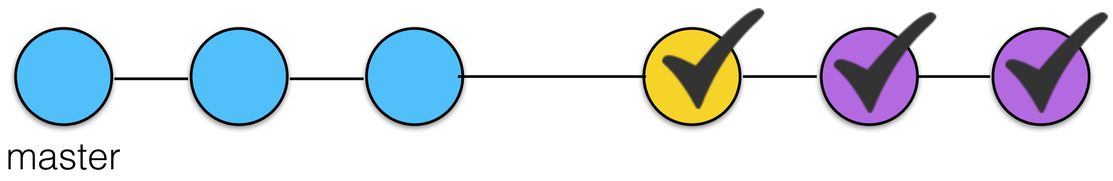
Feature Branching

Creating a branch for every feature you work on



Feature Branching

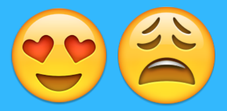
Creating a branch for every feature you work on



And merge only after it is reviewed

(I'm not talking about build outcome)

1. Both popular and controversial 有争议



2. Can be valuable when combined with other practices

3. Common challenges & solutions

4. A tour of SAAS tools

Why is it popular? 😍

- Lifecycle of an idea : Review process per feature
- Release flexibility 灵活发布
- Used in open-source projects
- Popular as « Github flow [link] »
- More and more support/tools around the concept of branch 更多工具

1. Both popular and controversial

ThoughtWorks | IT大咖说 知识分享平台

Very controversial 😩 (有争议)

Vivid 撕逼 debate since 2011 :

« The death of continuous integration »

Some would say it is an « organizational anti－pattern » [1] (Steve Smith)

While others write « why does Martin Fowler not understand feature branches » [2] and that the two concepts can go together

So what is Continuous Integration?

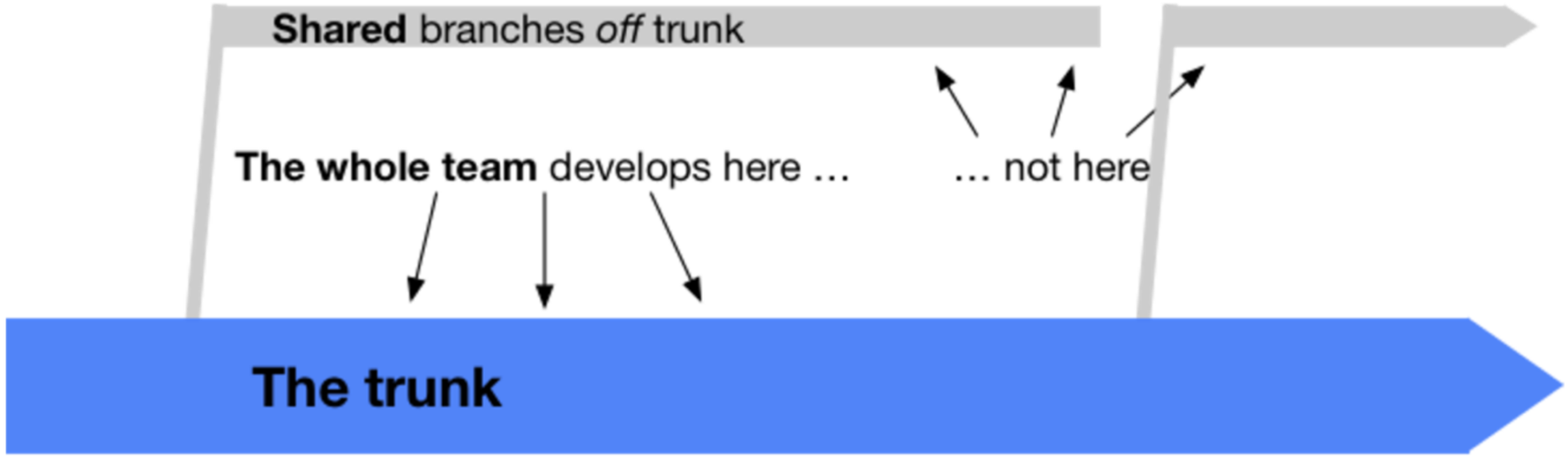
Martin Fowler:
« practice where members of a team integrate their work frequently, usually each person integrates at least daily »

Continous Integration:
It's a *non-event*

Differed integration (end of project)
Integration hell

So what is Continuous Integration?

Most often associated to *trunk-based development* (everybody commits to master)
主分支策略



Credits: trunkbaseddevelopment.com

* *master*, in Git nomenclature

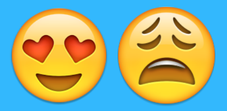
Feature Branching + Continuous Integration ?

In FB, you never commit directly to master, only merge to master

=> How often do you merge to master?

=> The system does not force you to integrate, it depends on your discipline

1. Both popular and controversial



2. Can be valuable when combined with other practices

3. Common challenges & solutions

4. A tour of SAAS tools

A combination of practices

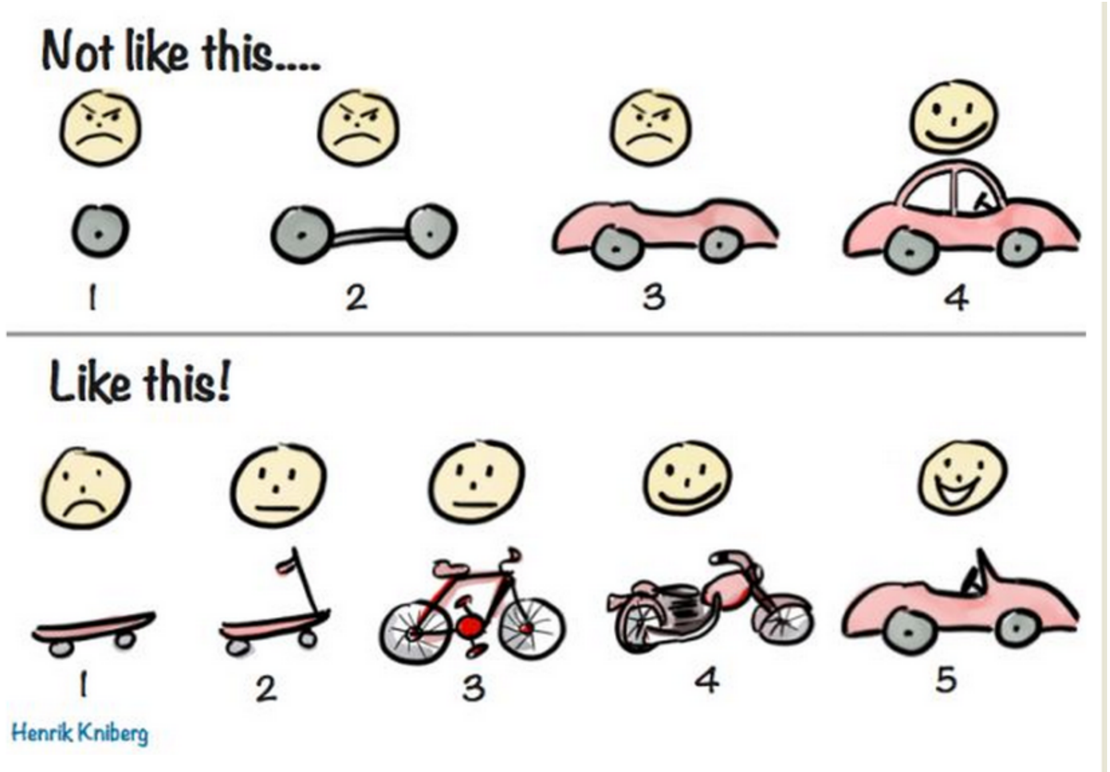
- Lean « Feature Branching »
- Automated (acceptance) testing
- A disciplined review process

Lean « Feature Branching »

#1

Always a working product:

- branch/and merge back to master
- no sub-branches

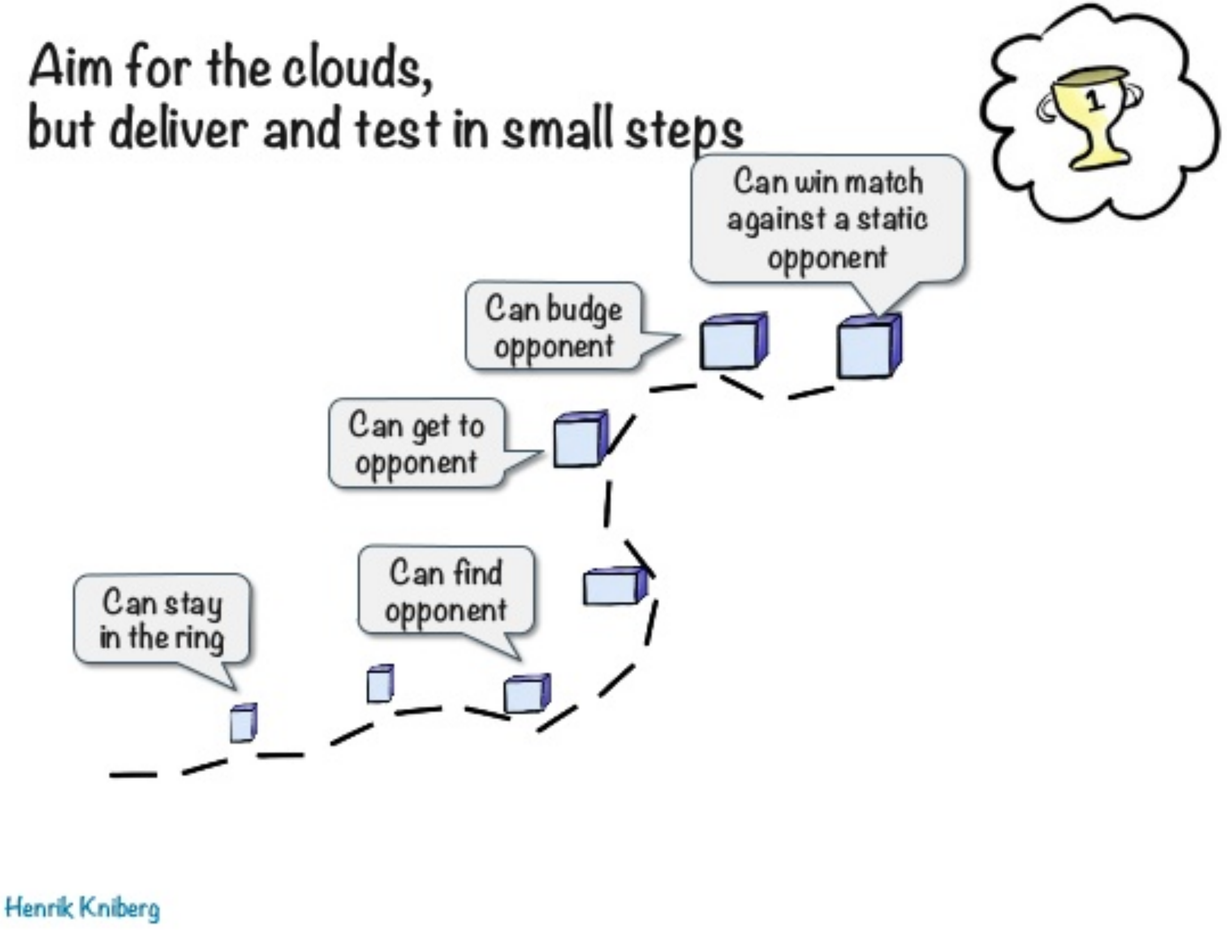


Lean « Feature Branching »

#2

Small steps ! (« Stories »)

A step = a day's work



Lean « Feature Branching »

If integration happens everyday => CI is not dead!

But it requires much <u>discipline</u>

Automated (acceptance) testing

#3

If you don't do it already => you should!

Build on every commit/every branch

<u>Not</u> only on the master branch

Gives confidence in refactoring
You known quickly when you've broken master

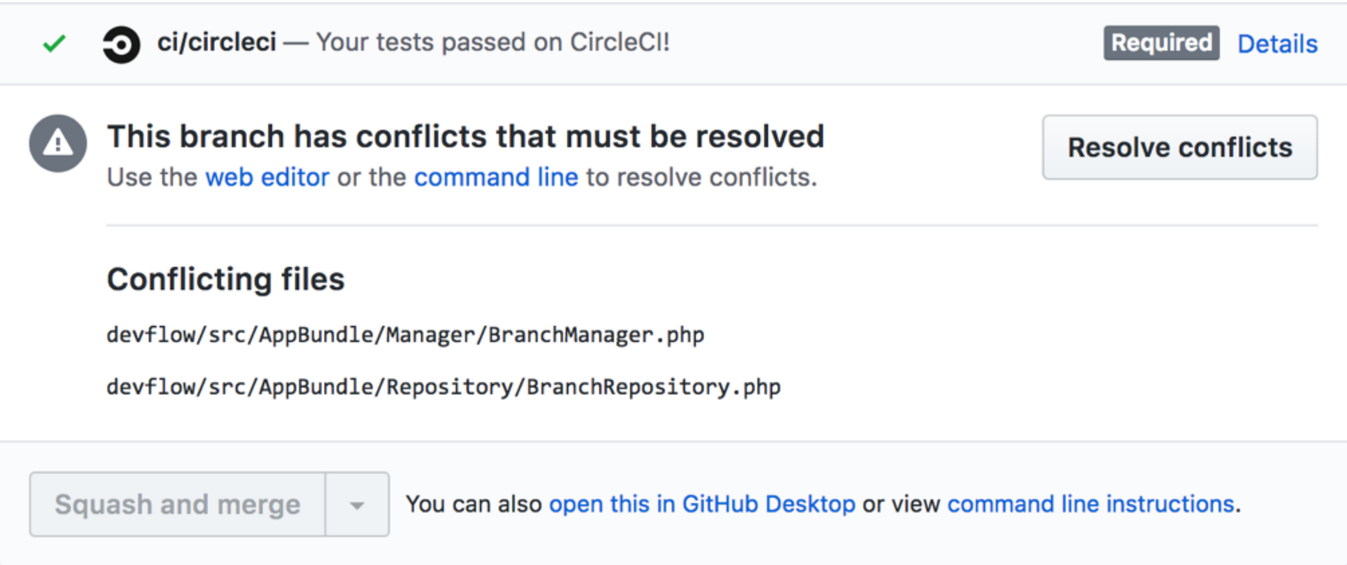
ThoughtWorks | IT大咖说 知识分享平台

A disciplined review process

#4

Define minimal review timeslots every day

e.g. every half-day before starting work



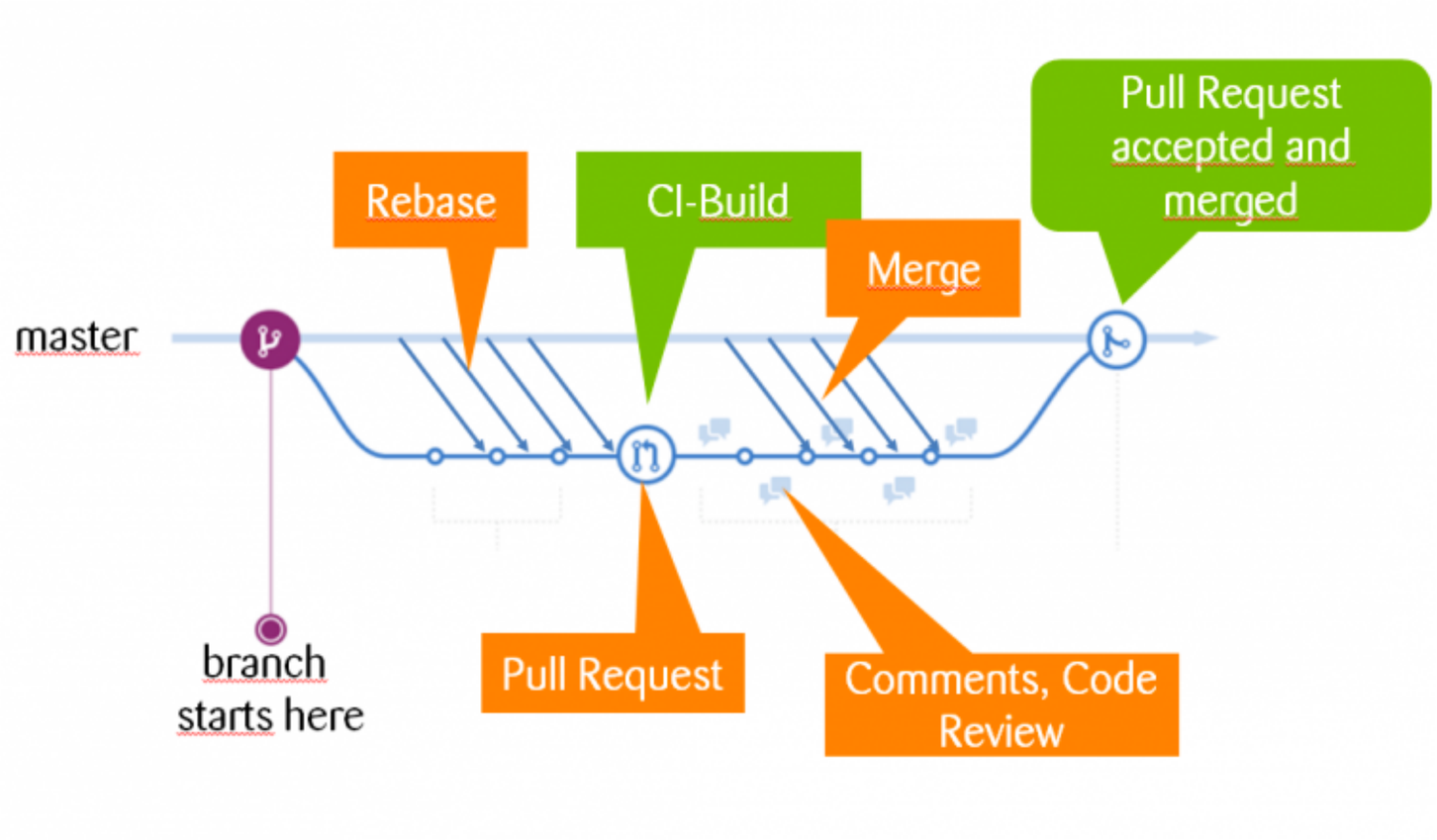
A disciplined review process

#5

Monitor long-lasting branches (2-3 days) and act!

- Git rebase
- Merge but act to hide in the UI
- Feature toggles « *branch by abstraction* »

Pull-Request / Review in detail



ThoughtWorks | IT大咖说 知识分享平台

Pull-Request / Review in detail

Discussion | Commits 1 | Files Changed 2

tildedave opened this pull request just now

Revert "Merge pull request #1 from tildedave/pull_request_demo"

Edit

Open

+ 1 addition

- 4 deletions

No one is assigned

No milestone

This reverts commit `6912d58`, reversing changes made to `98b9ef7`.

1 participant

Dave King added a commit — just now

Dave King `Revert "Merge pull request #1 from tildedave/pull_request_demo"` ··· `cf09b44`

You can add more commits to this pull request by pushing to the `revert_pull_request_1` branch on **tildedave/git-learning**

This pull request can be automatically merged. — Merge pull request

Pull Request 5: adding new features

.index.html [+]

```
1  <!DOCTYPE html>
2
3  <html lang="en">
4  <head>
5      <meta charset="utf-8" />
6      <title>TypeScript HTML Application</title>
7      <link rel="stylesheet" href="app.css" type="text/css" />
```

F — Can we use the newer CSS here?

ftottendev - less than a minute ago - reply — Status: Active

```
8      <script src="app.js"></script>
9  </head>
10 <body>
11     <h1>TypeScript Web Application</h1>
12
13     <div id="content"></div>
14 </body>
15 </html>
16
```

Advantages / Drawbacks

(vs trunk-based development)

- Review process. Online, asynchronous 异步
- Master is always deployable
- Leverage SAAS tools

- Small inventory of WIP commits
- Requires discipline (a lot)
- Won’t scale

Which one to try?

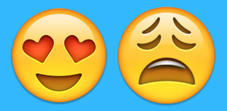
Feature Branching

- Open-source
- Small teams with big portfolio of projects
- When pair programming not possible
- Freelancer / web agency

Trunk-based development

- Co-located Commercial software development
- Growing teams (7+ people)
- Strong product/release management

1. Both popular and controversial



2. Can be valuable when combined with other practices

3. Common challenges & solutions

4. A tour of SAAS tools

Challenge 1: Epic feature (« too big to split »)

It's NEVER too big to split

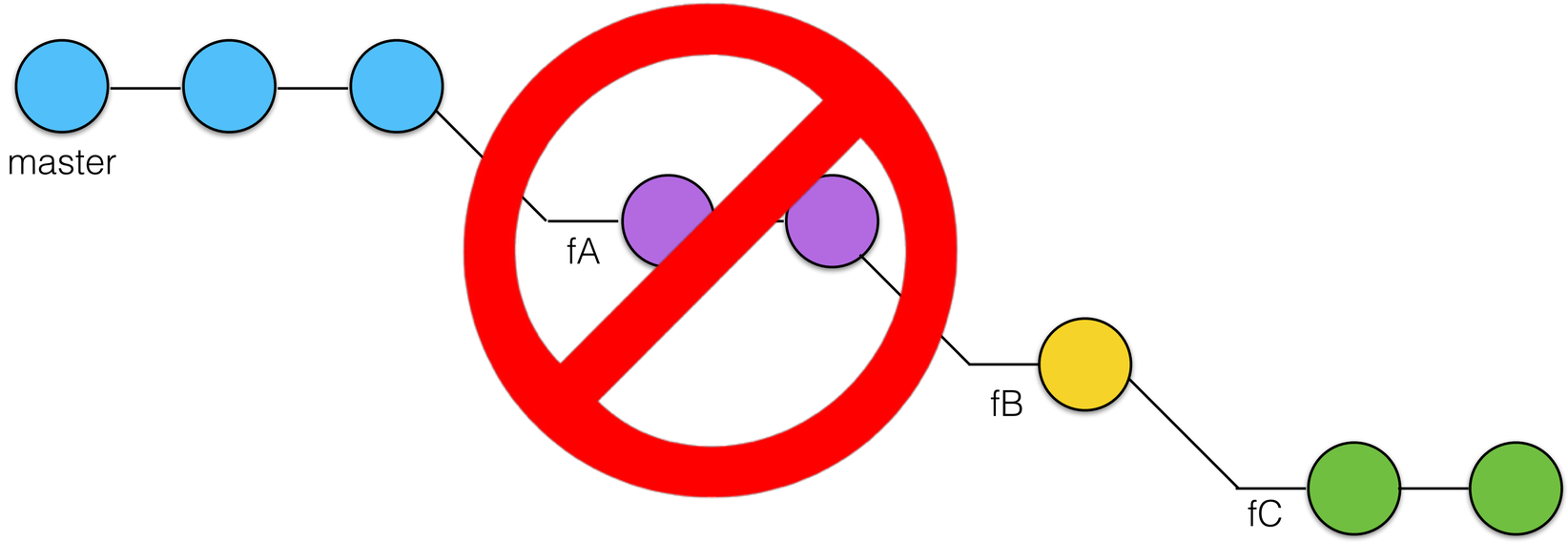
Grow the code with TDD in several simple features

Branching by abstraction « feature flags »

Challenge 2: Branching hell

Remember to branch only <u>from master</u>

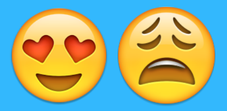
Need to re-use code? **refactor - rebase - continue**



Challenge 3: where to do manual testing / demo?

- On developer's laptop?
- On a shared test environment?
- On the staging after merging?
- On a farm of test environments?
- On a disposable environment per branch?

1. Both popular and controversial



2. Can be valuable when combined with other practices

3. Common challenges & solutions

4. A tour of SAAS tools

Why SAAS tools?

Free tier, fast to configure
Well integrated with Github/Gitlab
Handles the server management, updates and scaling
You can always switch to in-house/open-source solution later

Code hosting & reviewing

Github

GitLab

Running automated tests

circleci

TravisCI

SAUCELABS

One environment per branch

Standalone

runnable

DEVFLOW BETA

In hosting offering

platform.sh

PANTHEON® Website Management Platform

Open-source

Gitlab Community Edition (dynamic env.)

Code quality & security

Scrutinizer

DependencyCI

Thank you for listening!

How do you do branching?
Any pain points with your current branching strategy?

I'd love to hear about it.