ThinkingLabs
Ghent, Belgium
On the left, Korenlei
On the right, Graslei
Feature Branching is Evil
Thierry de Pauw | consulting CTO
takes questions at the end of the session
@tdpauw@mastodon.social
thinkinglabs.io

Being vulnerable ...

I'm **shy** and **introverted**

A tale of 2 teams

"Like all powerful tools, there are many ways you can use them (DVCS), and not all of them are good."

-- On DVCS, continuous integration, and feature branches, Jez Humble

Iqbal Ahmed
@propattern

Convincing people about the immeasurable rewards of Trunk Based Development is not an overnight task.

The idea that developers should work in small batches off master or trunk rather than on long-lived feature branches is still one of the most controversial ideas in the Agile canon, despite the fact it is the norm in high-performing organizations such as Google.[3]

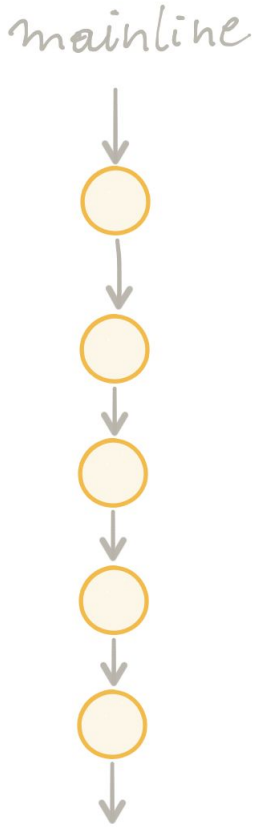
-- 2016 State of DevOps Report

Some definitions ...

ThinkingLabs

Mainline is the line of development which is the reference from which the builds of your system are created that feed into your deployment pipeline.

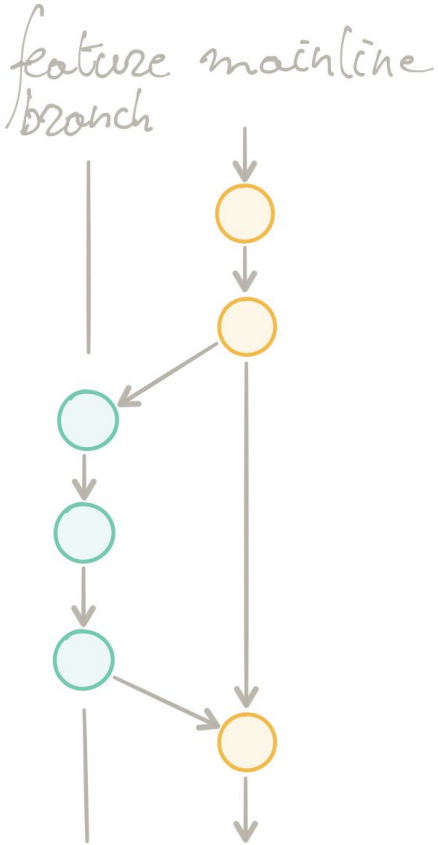
-- Jez Humble



ThinkingLabs

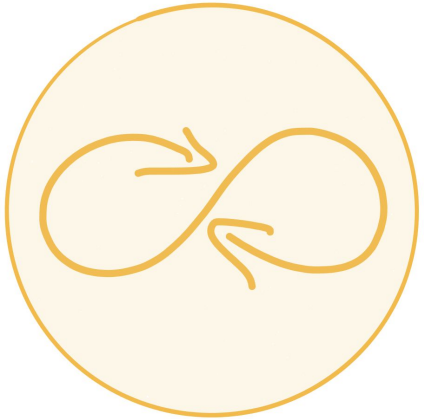
Feature Branching is a practice where people do not merge their code into mainline until the feature they are working on is "done" (but not “done done”).

-- Jez Humble



Continuous Integration is a practice where members of a team integrate their work frequently - usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build [...].

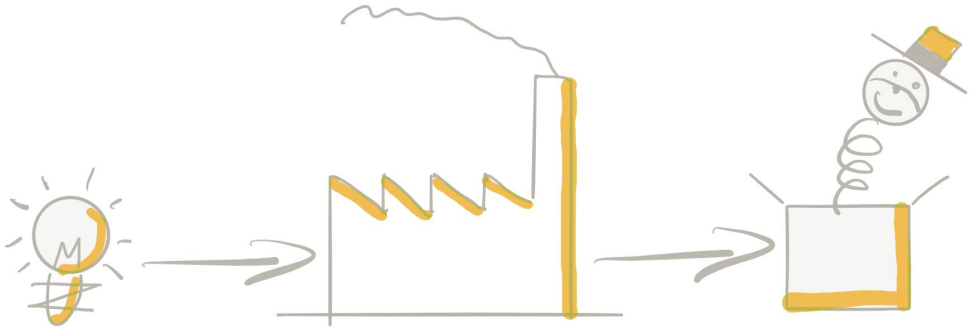
-- Martin Fowler



ThinkingLabs

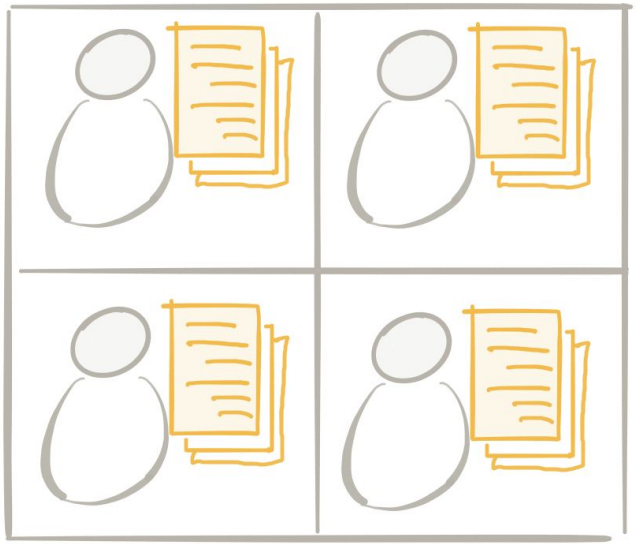
The goal of an Organisation is

to **sustainably minimise** the **lead time**
to create positive **business impact**.



Why long-running branches ?

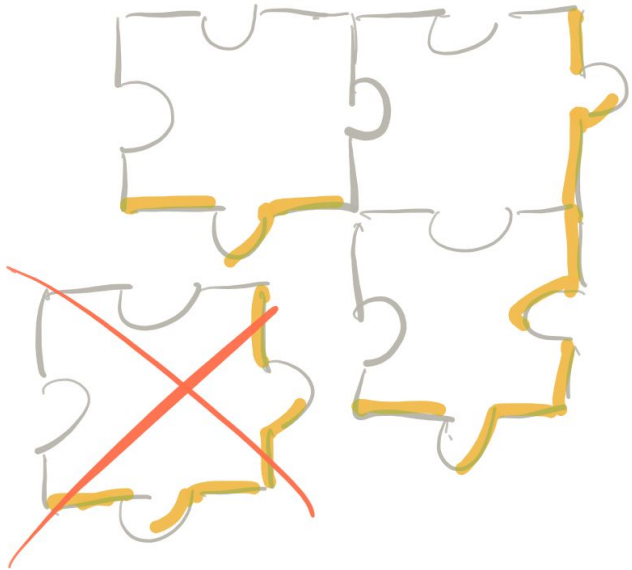
It allows us to work in isolation.
Therefore we are more productive.



*"Developing in isolation can help an **individual** go faster but it does not help a **team** go faster. Merge time and rework cannot be estimated and will vary wildly, and the team can only go as fast as the slowest merge."*

-- Steve Smith

If a refactoring goes nowhere
we can just delete it.



“A spike solution is a very simple program to explore potential solutions. Build the spike to only address the problem under examination and ignore all other concerns. Most spikes are not good enough to keep, so expect to throw it away."

-- extremeprogramming.org, Don Wells

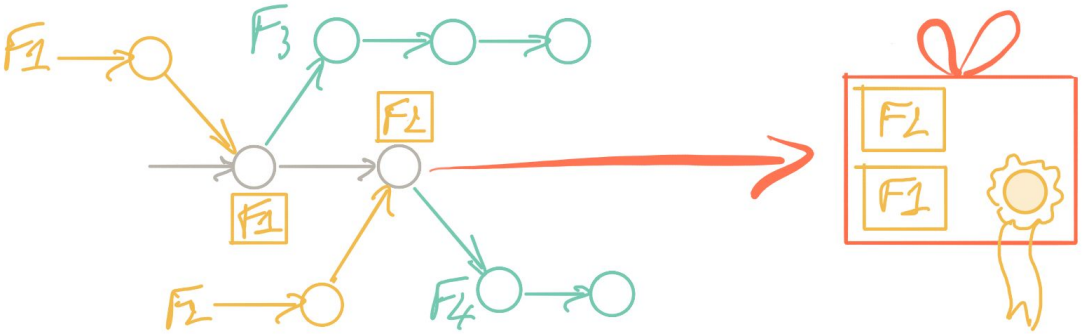
It allows us to control the quality of what goes into production.

"The objective is to eliminate unfit release candidates as early in the process as we can ...
You are effectively prevented from releasing into production builds that are not thoroughly tested and found to be fit for their intended purpose."

-- Continuous Delivery,
Jez Humble and Dave Farley

ThinkingLabs

It allows us to control which features get into production.



"Feature Branching is a poor man's modular architecture, instead of building systems with the ability to easy swap in and out features at runtime/deploy-time they couple themselves to the source control providing this mechanism through manual merging."

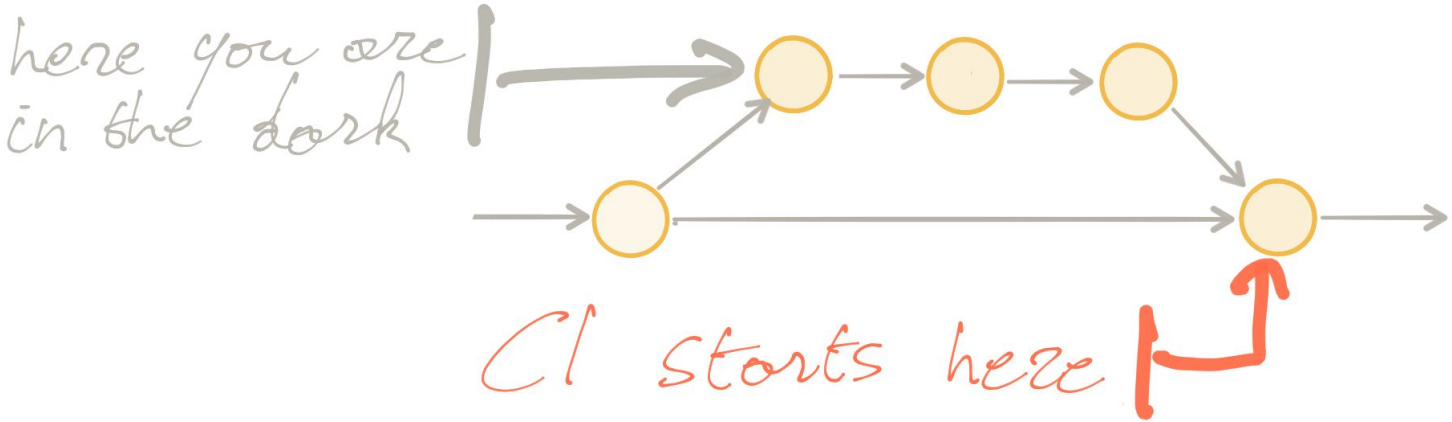
-- Dan Bodart

Why is this a problem ?

ThinkingLabs

Feature Branching **delays feedback**.

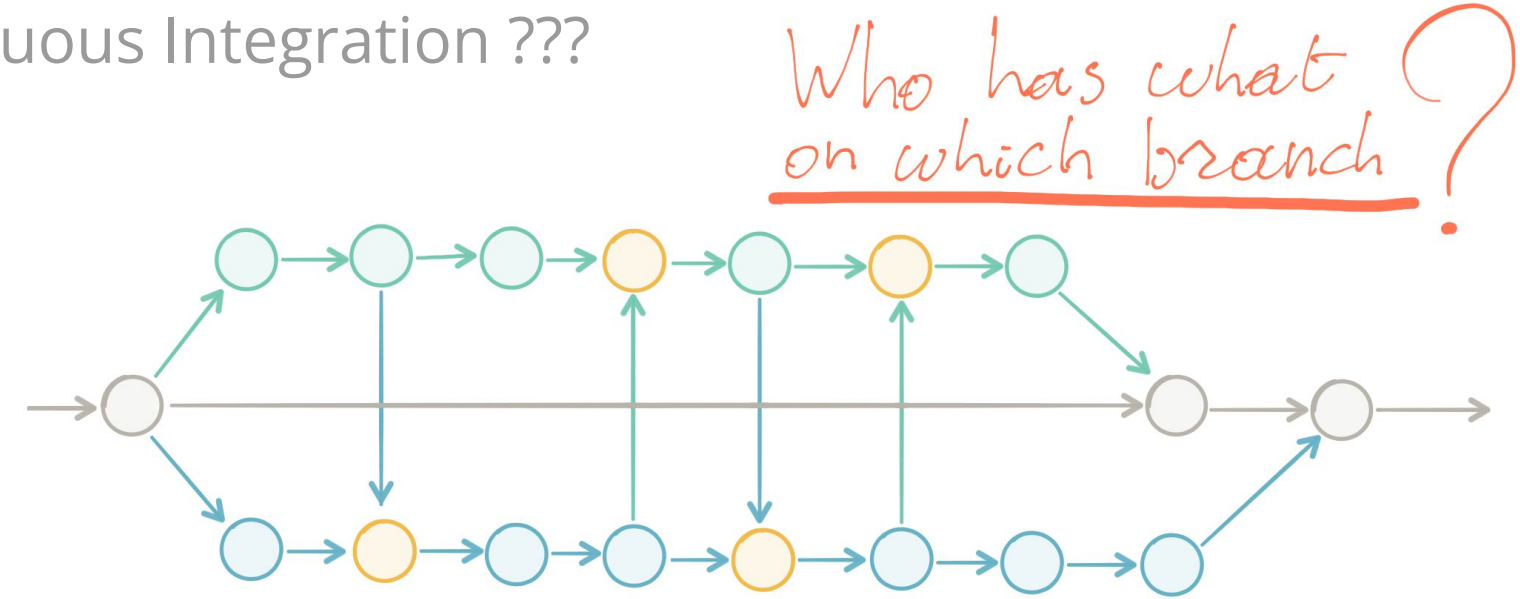
=> Continuous Isolation



ThinkingLabs

Feature Branching
hinders integration of features.

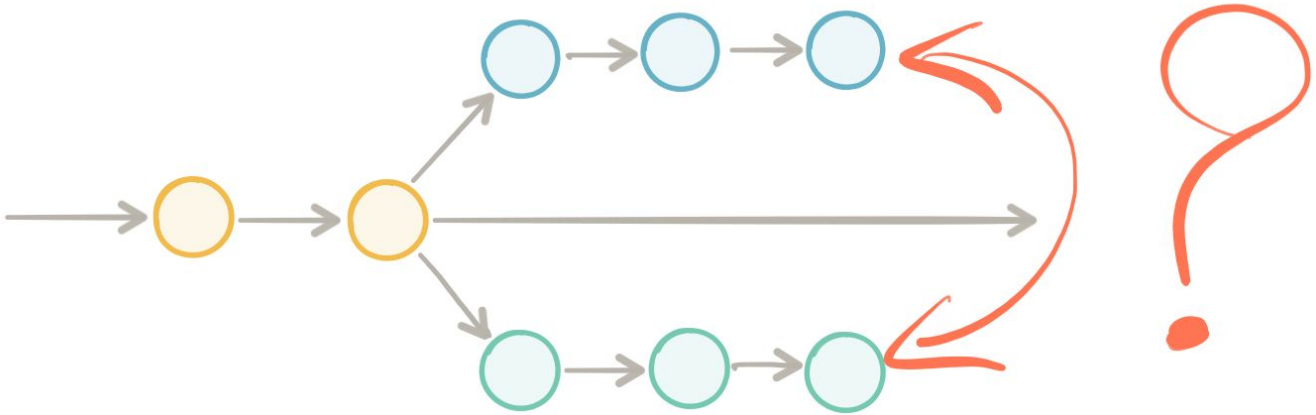
Promiscuous Integration ???



ThinkingLabs

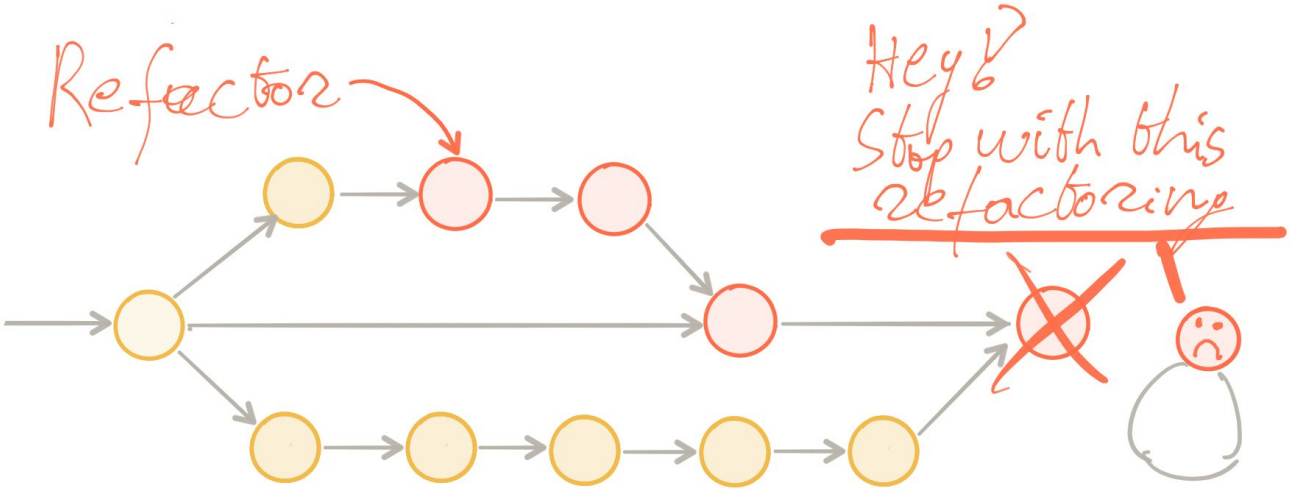
Feature Branching **hides work**
for the rest of the team.

frequently merging back to mainline
= communicating with your team



ThinkingLabs

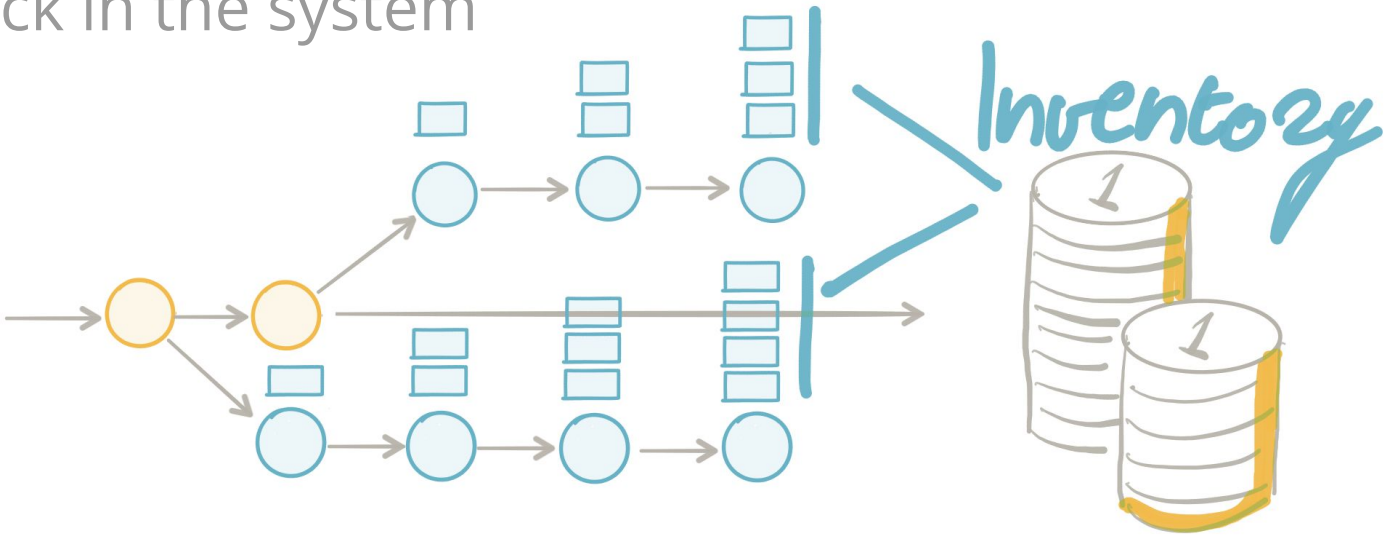
Feature Branching works
against refactoring.



ThinkingLabs

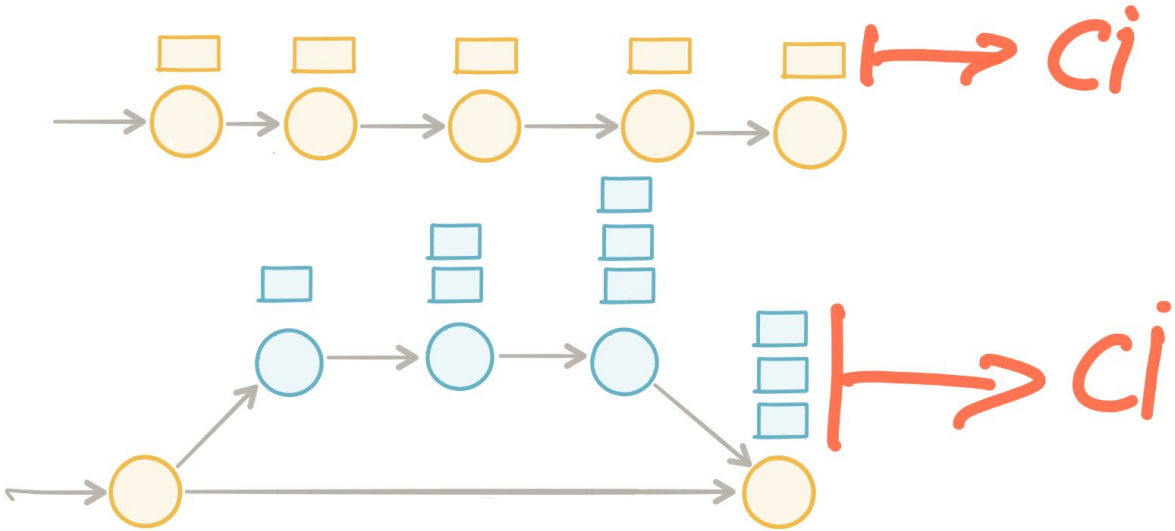
Feature Branching **creates inventory**.

inventory
= money stuck in the system



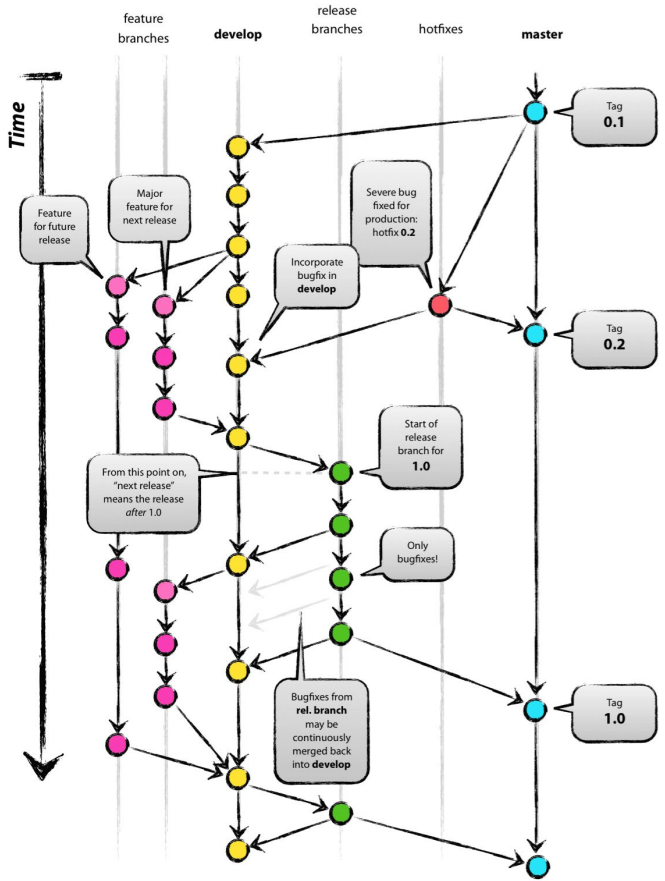
ThinkingLabs

Feature Branching **increases risk**.



ThinkingLabs

Feature Branching creates
cognitive overload.



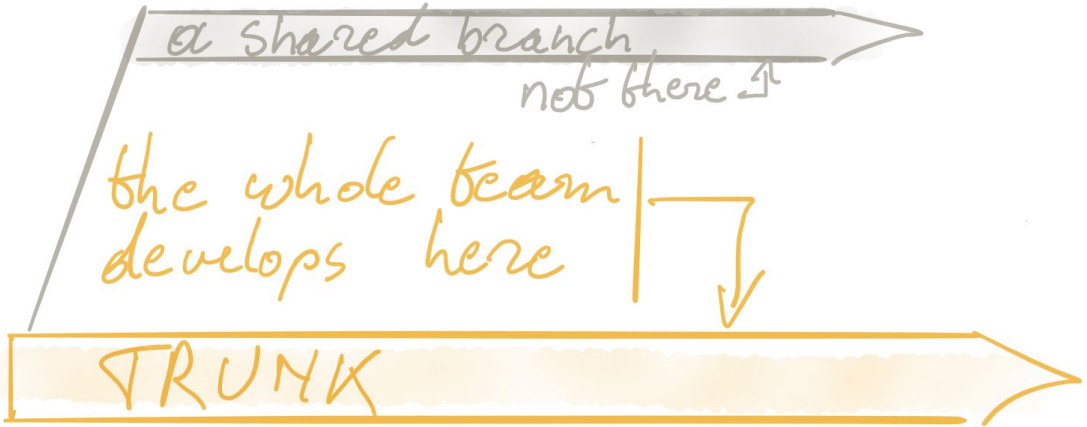
How can we avoid this ?

ThinkingLabs

Continuous Integration

Your application is always in a
releasable state on main line.

Trunk Based Development



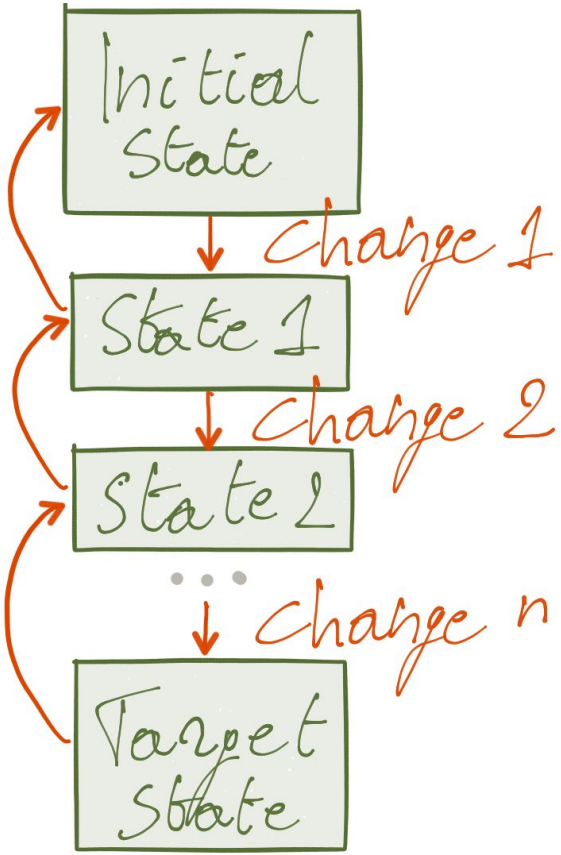
ThinkingLabs

Break large changes into a set of **small incremental changes**.

always commit on **Green**.

decoupled code base.

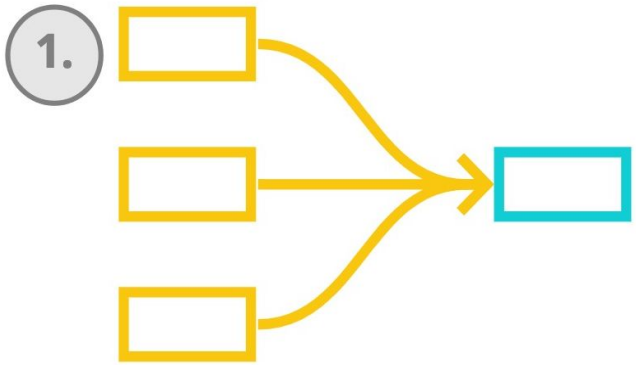
lots of **fast** tests.

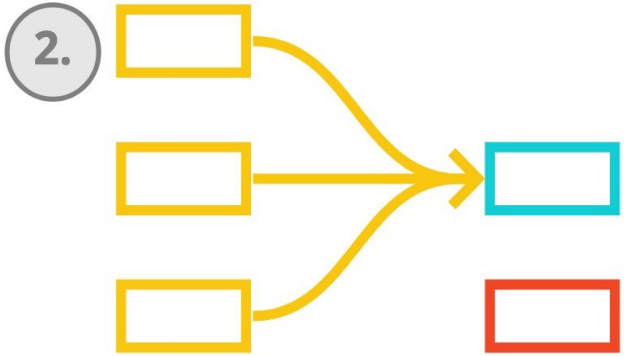


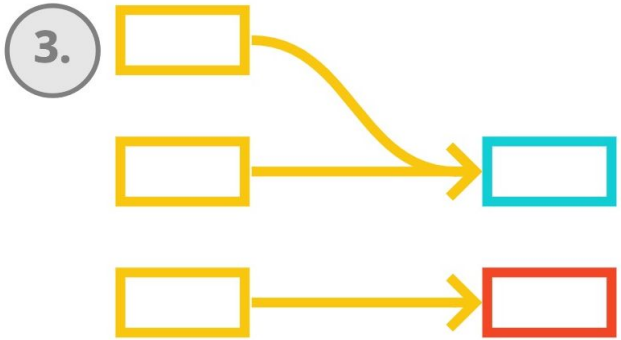
Hide unfinished new functionality.

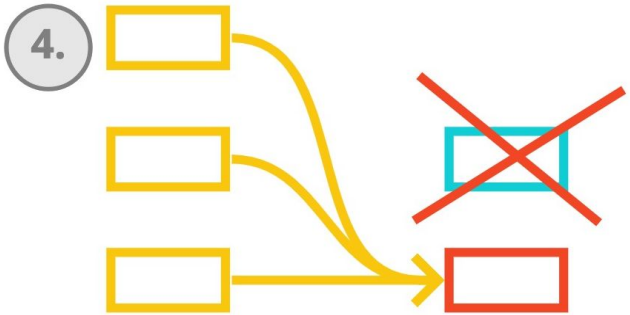
Use **Expand-Contract**
when performing large refactorings.

ThinkingLabs









Feature Toggles to decouple
feature release from code deployment.

Bouillier Clément @clem_bouillier · Jan 7

Replying to @tdpauw

1st part, I feel that we could also say #FeatureToggle is evil when badly applied, as #FeatureBranching is evil when badly applied

2 1

Feature Toggles done badly are **evil too** !

Bouillier Clément @clem_bouillier · Jan 6

Replying to @clem_bouillier @tdpauw

also it sounds like feature toggles are easy, but they are not...they also have their drawbacks (in code quality if not mastered)

1 1

A frequently asked question

How to perform Code Reviews ?

When mature enough **no Code Reviews**.

Pair Programming => Continuous Code Review

post-commit review

- pre-merge: short lived branches + Pull Request
- post-merge: review all commits on mainline

Benefits

More frequent commits to mainline

=> more frequent builds

=> more frequent deployments

=> **reduced Time to Market**

=> more experiments

=> **uncover more unmet needs**

More frequent builds

=> uncovers more problems earlier

=> fix problems immediately

=> build quality in

=> **better Stability & Quality**

Where is the evilness ?

In reply to Thierry de Pauw

code designer & more @alexboly · Nov 8

@tdpauw However, it's important to understand that long lived **feature branches are** a symptom. The **evil** hides beneath.

In reply to code designer & more

Bogdan Dumitrescu @bogdand84 · Nov 1

@alexboly They are evil when developers don't know how to develop in small increments. Unfortunately it's a rare skill.

Thierry de Pauw @tdpauw · Nov 10

@alexboly IMHO the evil lies in: too coupled code base, missing tests, or not being aware on how to apply small incremental changes

… and **slow builds**

"*Trunk-based development requires a big mindset shift. Engineers thought trunk-based development would never work, but once they started, they could not imagine ever going back.*"

-- Gary Gruver,
Directory of Engineering for HP's LaserJet Firmware division

ThinkingLabs

Hello, I am Thierry de Pauw

is chief imposter
likes dark chocolate
dark means > 50% cacao, prefers 70% and more

Engineer at the fintech startup Abbove.com

Founder of **Thinking**Labs, advisory firm on IT delivery

Article series:
https://thinkinglabs.io/on-the-evilness-of-feature-branching

ThinkingLabs

Resources

SCM Patterns (ch 4 Mainline; ch 5 Active Development Line), Stephen Berczuk and Brad Appleton
Growing Object Oriented Software guided by Tests, p172 Keyhole Surgery for Software, Steve Freeman and Nat Pryce
Continuous Delivery (ch 14 Advanced Version Control), Jez Humble and Dave Farley
The Role of Continuous Delivery in IT and Organizational Performance, Nicole Forsgren and Jez Humble
The State of DevOps Report 2016, Alanna Brown, Nicole Forsgren, Jez Humble, Nigel Kersten and Gene Kim
DevOps Handbook (ch 11 Enable and Practice CI), Gene Kim, Jez Humble, Patrick Debois and John Willis
Accelerate (ch 4 Technical Practices), Nicole Forsgren, Jez Humbe and Gene Kim
Measuring Continuous Delivery (ch 7 The Mainline Throughput indicator), Steve Smith
trunkbaseddevelopment.com
ThoughtWorks Technology Radar on GitFlow
Continuous Integration on a dollar a day, James Shore
On DVCS and Continuous Delivery, Jez Humble
Why software development methodologies suck, Jez Humble
Don't Feature Branch, Dave Farley
Feature Branch, Martin Fowler
Version Control Stragies series, Steve Smith

ThinkingLabs

More Resources

More Feature Branches means less Continuous Integration, InfoQ interview with Steve Smith
The Death of Continuous Integration, Steve Smith
Long-Running Branches Considered Harmfull, Jade Rubick
An e-mail conversation with Steve Smith on Trunk Based Development
Continuous Isolation, Paul Hammant
What is Trunk Based Development ?, Paul Hammant
Trunk Based Development, Jon Arild Tørresdal
You Are What You Eat, Dave Hounslow
Google's Scaled Trunk Based Development, Paul Hammant
Legacy App Reju venation, Paul Hammant
Why Google Stores Billions of Lines of Code in a Single Repository ?, Google
The history of “Taking Baby Steps”, Adrian Bolboaca
Baby Steps TDD approach, David Völkel
4 Simple Tricks to avoid Merge Conflicts, Robert Mißbach
From GitFlow to Trunk Based Development, Robert Mißbach
Short-lived branches, Corey Haines

Even More Resources

Introducing Branch by Abstraction, Paul Hammant
Branch by Abstraction, Martin Fowler
Make Large Scale Changes Incrementally with Branch by Abstraction, Jez Humble
branchbyabstraction.com
Feature Toggles, Pete Hodgson
#NoStaging - Pipeline Conf 2016, Dave Nolan
When Feature Flags go Wrong, Edith Harbaugh
Managing Feature Flag Debt with Split, Adil Aijaz
Continuous Delivery and Code Review from the Continuous Delivery Google Group
Theory X and Theory Y from Wikipedia
Continuous Review, Paul Hammant
Non-Continuous Review, Paul Hammant
Code Review: Why are we doing it ?, Sandro Mancuso
Code Reviews in Trunk Based Development, Robert Mißbach
A conversation in the SoCraTes Slack #codereview channel on ... Code Reviews and Trunk Based Development
A reply on Twitter by Michiel Rook regarding When code reviews would not be required