accu 2023

THE PRACTICES THAT MAKE CONTINUOUS INTEGRATION

THIERRY DE PAUW

Being vulnerable ...

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


ThinkingLabs
Feature Branching is Evil
Continuous Delivery Conference NL 2016
Thierry de Pauw, @tdpauw
@tdpauw
thinkinglabs.io

ThinkingLabs

Continuous Integration Test

everyone in the team commits at least once a day to Mainline

every commit to Mainline triggers an automated build and execution of all automated tests

whenever the build fails it gets fixed within 10 min

@tdpauw
@tdpauw@mastodon.social

Continuous Integration is a practice to ensure always working software and to get feedback within a few minutes to whether any given change broke the application.

– Jez Humble

ThinkingLabs

14 practices

Team Working for CI

Coding for CI

Building for CI

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Team Work for CI

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. When Broken Revert

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

allan kelly
@allankellynet

Replying to @SteveSmith_Tech and @davefarley77

Historically I meet 1 team a year on average who aren't using version/source code control. Common reasons:
- very immature team
- SQL
- PickOS derivative or ERM/CRM system

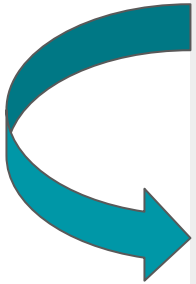
Last category may even lack tooling

5:34 PM · Nov 8, 2020 · Twitter Web App

1. **Version Control Everything**
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs



1. **Version Control Everything**
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. **Revert When Broken**
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

Without Version Control:

- no single source of truth
- hard to rollback a deployment
- all other CI practices fall flat

@tdpauw
@tdpauw@mastodon.social

First practice that requires a tool!

1. **Version Control Everything**
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

Version Control System is a communication tool.

1. **Version Control Everything**
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

From now on, our code in revision control will always build successfully and pass its tests.

– James Shore

@tdpauw
@tdpauw@mastodon.social

1. Version Control Everything
2. **Agree as a Team to never Break the Build**
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

From now on, our code in revision control will always build successfully and pass its tests.

– James Shore

1. Version Control Everything
2. **Agree as a Team to never Break the Build**
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. **Commit Only on Green**
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. **Run a Local Build**
13. **Have a Vast Amount of High Quality Automated Tests**
14. **Have a Fast Build**

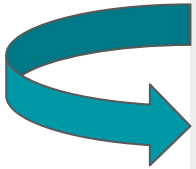
@tdpauw
@tdpauw@mastodon.social

Stop the line, fix immediately

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. **Do not Push to a Broken Build**
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs



1. Version Control Everything
2. **Agree as a Team to never Break the Build**
3. **Do not Push to a Broken Build**
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

The whole team owns the failure

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Precondition to Continuous Integration

Fix a broken build within 10 mins

Otherwise …
a whole team at stand still
disables on-demand releases

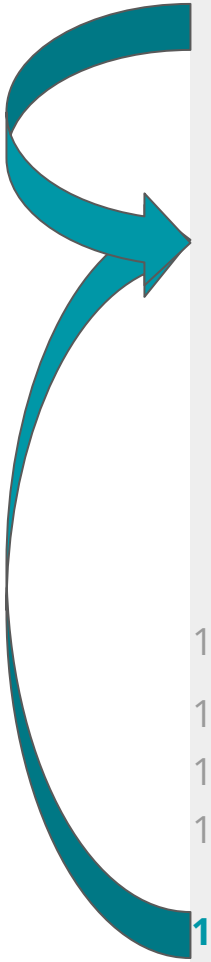
@tdpauw
@tdpauw@mastodon.social

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. **Revert When Broken**
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

ThinkingLabs

Why 10 mins?

because Have a Fast Build



1. **Version Control Everything**
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. **Revert When Broken**
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. **Have a Fast Build**

@tdpauw
@tdpauw@mastodon.social

Coding for CI

</>

5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Break large changes in a series of small incremental changes

=> keep the application always working

=> never tearing the application apart

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. **Make all changes in Small Increments**
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

This is hard work!

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. **Make all changes in Small Increments**
6. Commit Frequently
7. Commit Only on Green
8. **Decouple the Codebase**
9. **Adopt Expand-Contract**
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Continuous Integration
= integrate early and often

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken

5. Make all changes in Small Increments
6. **Commit Frequently**
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract

10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

When not Committing Frequently

- introduce batch work
- integrating becomes time-consuming
- prevents communication with the team

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

When Committing Frequently

- changes are small
- merge conflicts are less likely
- reverting a failing change is easier

@tdpauw
@tdpauw@mastodon.social

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. **Commit Frequently**
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

ThinkingLabs

Gentle design pressure to ...

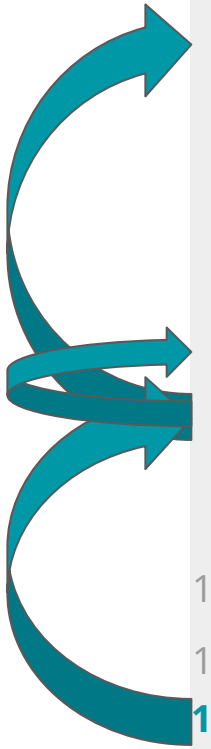
1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. **Make all changes in Small Increments**
6. **Commit Frequently**
7. Commit Only on Green
8. **Decouple the Codebase**
9. Adopt Expand-Contract
10. **Hide Unfinished Functionality**
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. **Have a Fast Build**

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Only commit when
the Local Build says SUCCESS

=> Test Driven Development



1. Version Control Everything
2. **Agree as a Team to never Break the Build**
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. **Commit Frequently**
7. **Commit Only on Green**
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. **Run a Local Build**
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Small increments
requires a decoupled codebase

=> improves quality

=> reduces engineering time

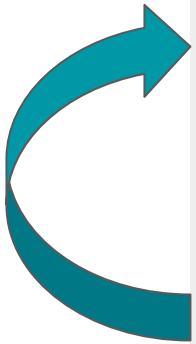
1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. **Make all changes in Small Increments**
6. **Commit Frequently**
7. Commit Only on Green
8. **Decouple the Codebase**
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Little unknown gem

Strong enabler for Continuous Integration



1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken

5. Make all changes in Small Increments

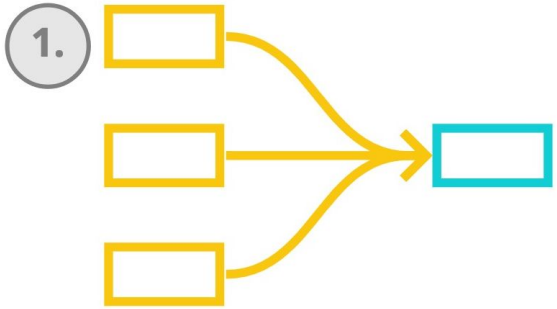
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase

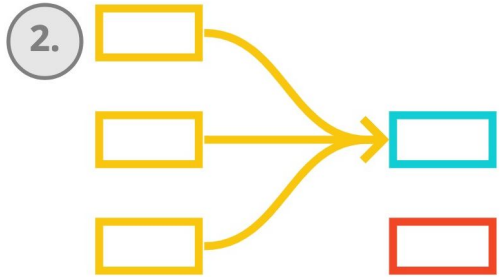
9. Adopt Expand-Contract

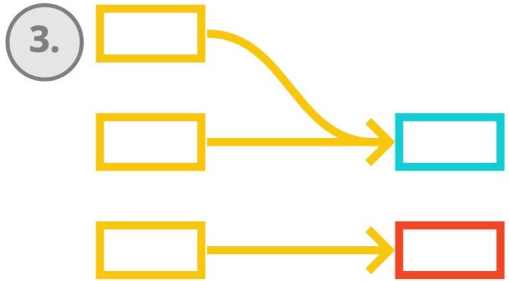
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

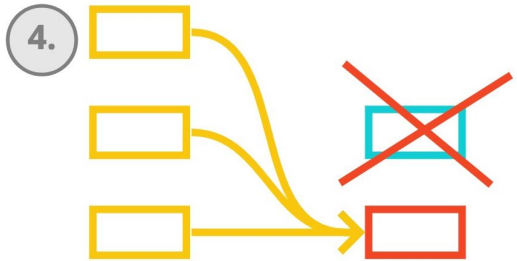
@tdpauw
@tdpauw@mastodon.social

ThinkingLabs









1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. **Adopt Expand-Contract**
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

What if a feature
takes too long to implement?

=> perfectly acceptable to have
unfinished functionality
in production

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. **Commit Frequently**
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. **Hide Unfinished Functionality**
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

Feature Toggles

enabler of Operability and Resilience

but ...

comes with their fair share of problems

@tdpauw
@tdpauw@mastodon.social

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. **Hide Unfinished Functionality**
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

ThinkingLabs

Building for CI

11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Build Script -> **SUCCESS** or **FAILURE**

Used by
Local Build and Commit Build

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. **Automate the Build**
12. **Run a Local Build**
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

Second and last practice that requires a tool!

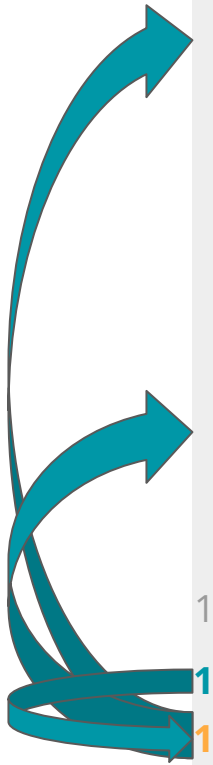
1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. **Automate the Build**
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Prevent broken build

=> run a Local private Build
before committing



1. Version Control Everything
2. **Agree as a Team to never Break the Build**
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. **Commit Only on Green**
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. **Automate the Build**
12. **Run a Local Build**
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build

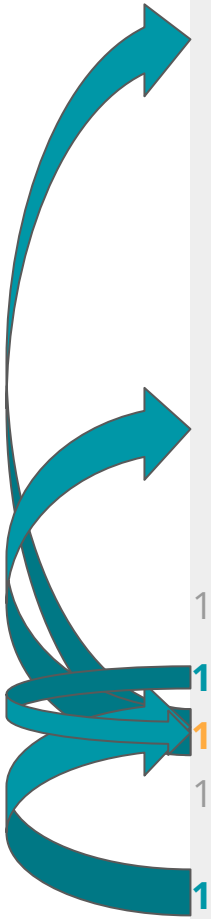
@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Commit to Mainline

=> triggers the Commit Build

Monitor the Commit Build!



1. Version Control Everything
2. **Agree as a Team to never Break the Build**
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. **Commit Only on Green**
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. **Automate the Build**
12. **Run a Local Build**
13. Have a Vast Amount of High Quality Automated Tests
14. **Have a Fast Build**

@tdpauw
@tdpauw@mastodon.social

No tests = no feedback

=> cannot Commit Frequently

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. **Have a Vast Amount of High Quality Automated Tests**
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

Gain confidence

- enough automated tests
- and of high-quality

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. **Have a Vast Amount of High Quality Automated Tests**
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Types of tests:

- Unit Tests
- Integration Tests
- Automated Acceptance Tests
- Smoke Tests

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. **Have a Vast Amount of High Quality Automated Tests**
14. Have a Fast Build

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

Commit Frequently

=> Fast Build

1. Version Control Everything
2. **Agree as a Team to never Break the Build**
3. Do not Push to a Broken Build
4. **Revert When Broken**
5. Make all changes in Small Increments
6. **Commit Frequently**
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. **Run a Local Build**
13. Have a Vast Amount of High Quality Automated Tests
14. **Have a Fast Build**

@tdpauw
@tdpauw@mastodon.social

When the Build is slow

- not executing the Local Build
- execute the Local Build less often

@tdpauw
@tdpauw@mastodon.social

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. **Have a Fast Build**

ThinkingLabs

What is fast?

- 10 min is the limit
- under 5 min is the focus
- 30s is plain bonus for engineers

@tdpauw
@tdpauw@mastodon.social

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. **Have a Fast Build**

ThinkingLabs

Non Reliable Tests (aka Flaky Tests)

=> increases delivery lead time

Credits to Maaret Pyhäjärvi (@maaretp)

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken
5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build
15. **Have Reliable Tests**

@tdpauw
@tdpauw@mastodon.social

ThinkingLabs

How can we fix Flaky Tests?
=> put them in Quarantine

1. Version Control Everything
2. Agree as a Team to never Break the Build
3. Do not Push to a Broken Build
4. Revert When Broken

5. Make all changes in Small Increments
6. Commit Frequently
7. Commit Only on Green
8. Decouple the Codebase
9. Adopt Expand-Contract

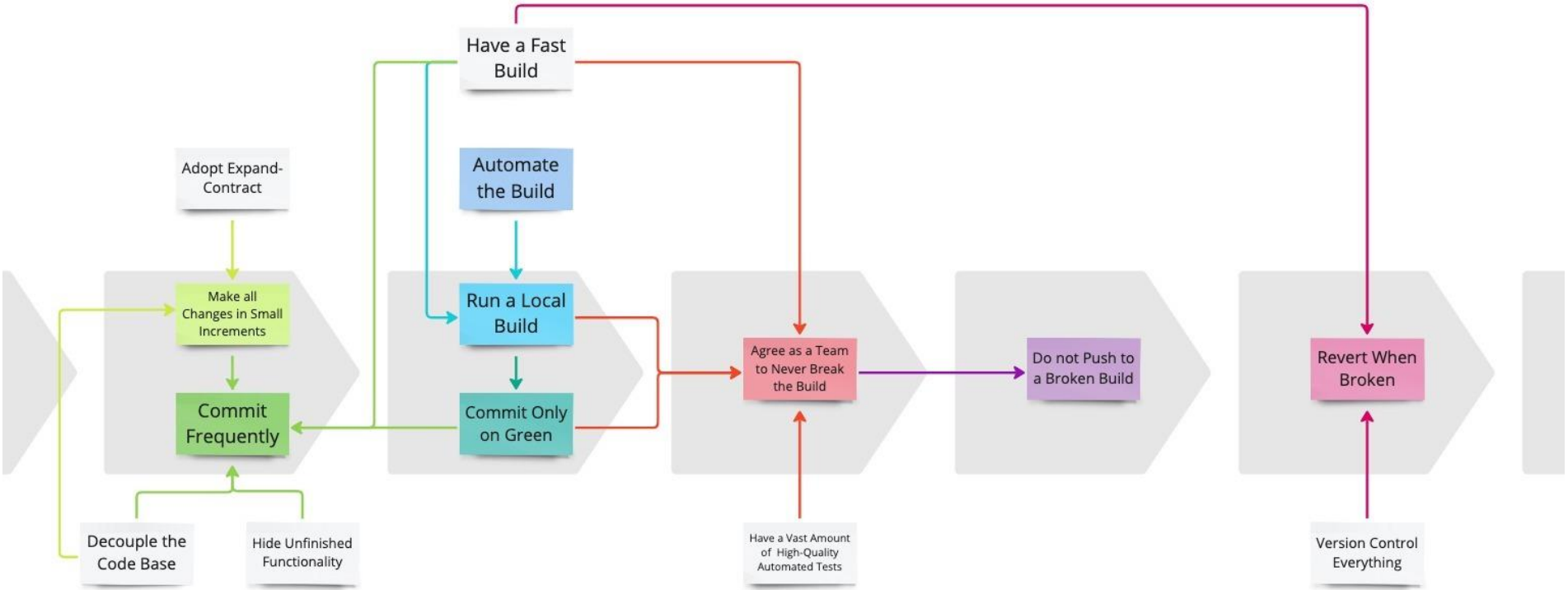
10. Hide Unfinished Functionality
11. Automate the Build
12. Run a Local Build
13. Have a Vast Amount of High Quality Automated Tests
14. Have a Fast Build
15. **Have Reliable Tests**

@tdpauw
@tdpauw@mastodon.social

Only two tools!

- Version Control System
- Automated build

ThinkingLabs

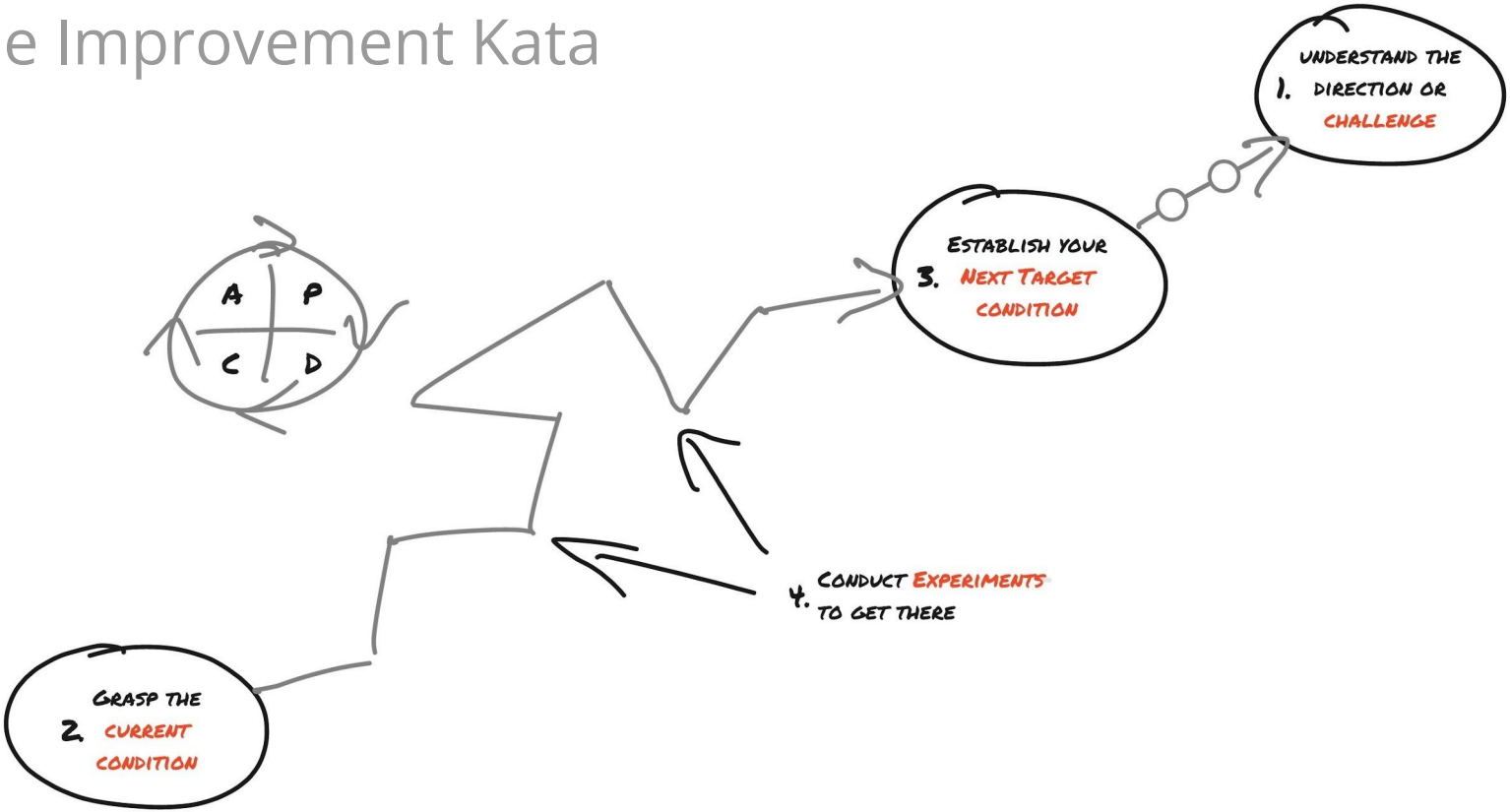


Adapted from Michael Lihs (@kaktusmimi), ThoughtWorks

Where do we start?

ThinkingLabs

The Improvement Kata



@tdpauw
@tdpauw@mastodon.social

Continuous Integration

together with trunk-based development

predicts

higher throughput and quality.

Hello, I am Thierry de Pauw

The article series: The Practices that make Continuous Integration

https://thinkinglabs.io/the-practices-that-make-continuous-integration

Acknowledgments:

Els, the one I love!

Lisi Hocke (@lisihocke), Seb Rose (@sebrose) and Steve Smith (@SteveSmith_Tech) for their thorough reviews of the article series.

Martin Van Aken (@martinvanaken), Martin Dürrmeier (@md42), Aki Salmi (@rinkkasatiainen), Nelis Boucke (@nelisboucke), Karel Bernolet (@BernoletKarel) for reviewing the slides.