



**ORLANDO
FLORIDA
AUGUST 7-11**

AGILE 2017





Rachel Laycock @rachellaycock

Continuous Delivery Explained

ORLANDO
FLORIDA
AUGUST 7-11

AGILE 2017



ThoughtWorks®

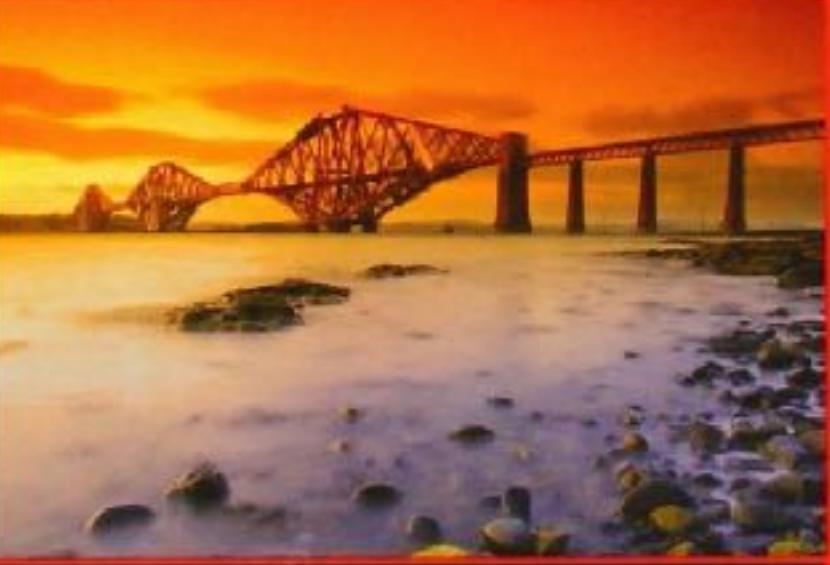
Delivering value
early and often

The Addison-Wesley Signature Series

CONTINUOUS DELIVERY

RELIABLE SOFTWARE RELEASES THROUGH BUILD,
TEST, AND DEPLOYMENT AUTOMATION

JEZ HUMBLE
DAVID FARLEY



Foreword by Martin Fowler

A MARTIN FOWLER SIGNATURE BOOK



Release Cadence

Mary Poppendieck



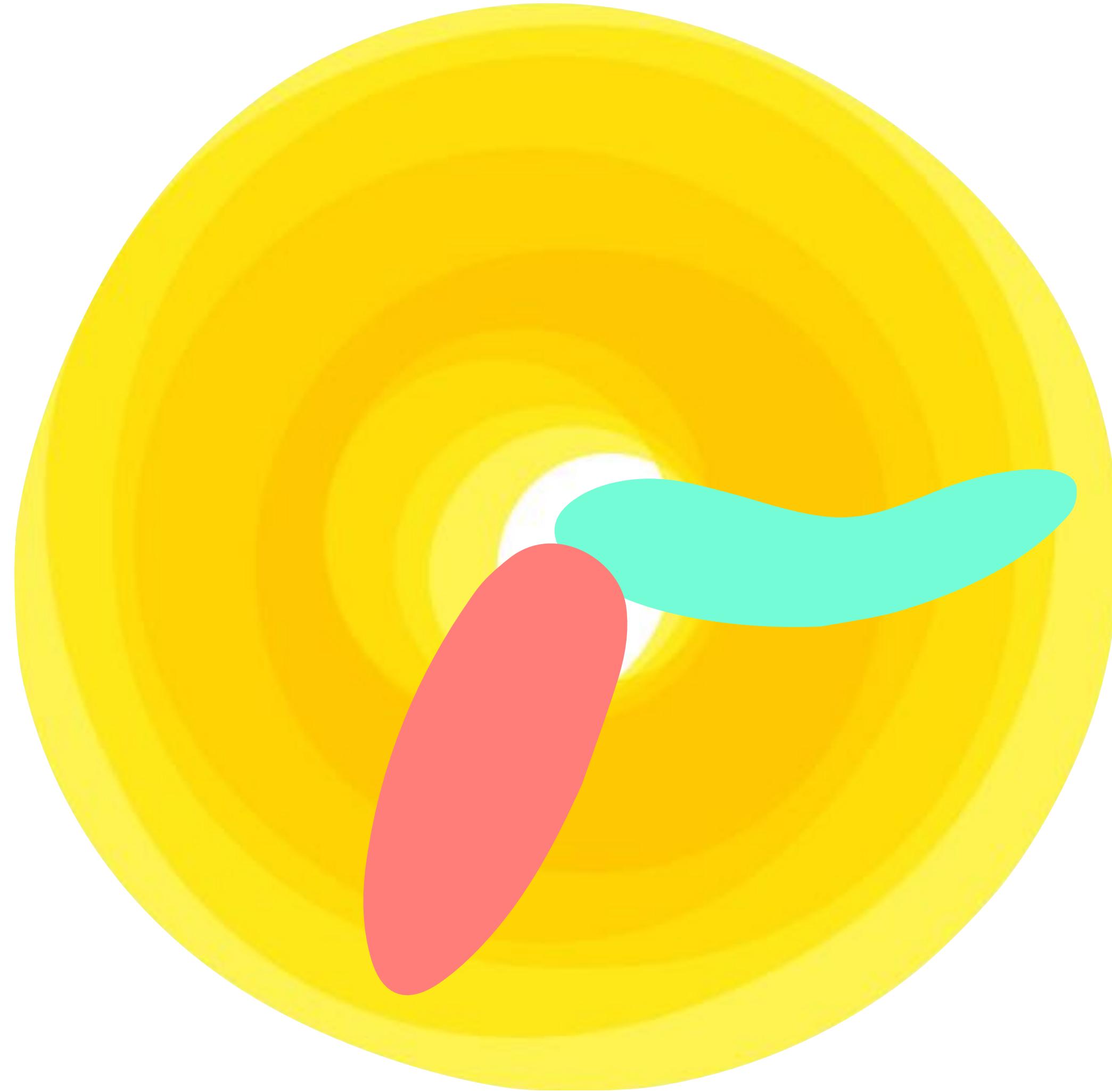
Tom Poppendieck



“How long would it take your organization to deploy a change that involves just one single line of code?”

“Can you do this on a repeatable, reliable basis?”

Release Cadence



Continuous Delivery is **BIG**

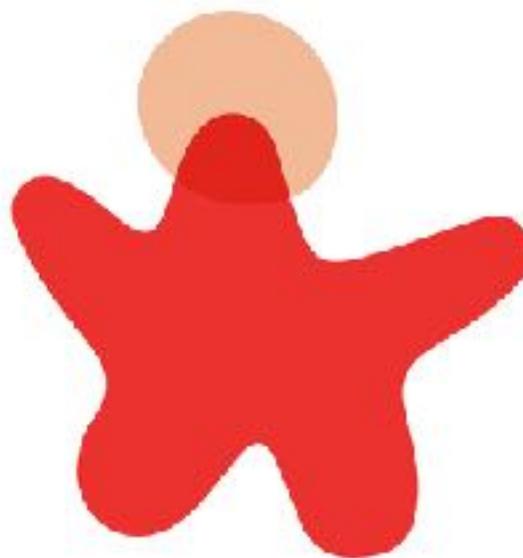
Organisational Alignment

Release Management

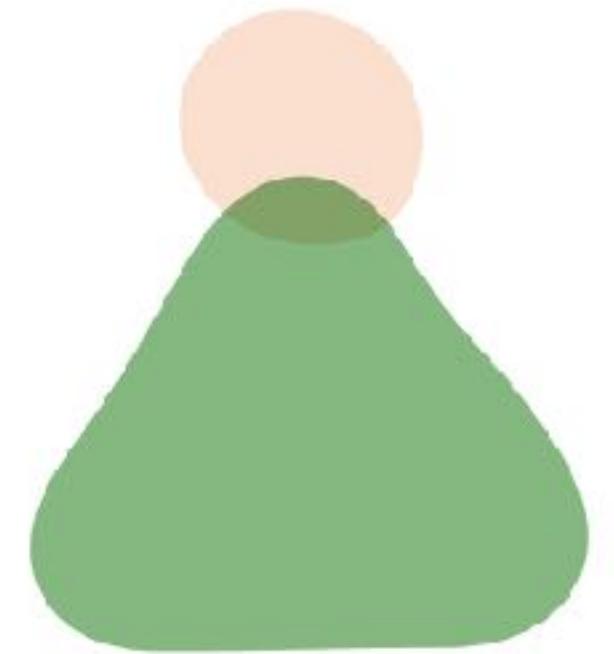
Architecture	Quality Assurance	Continuous Integration	Configuration Management	Data Management	Environments & Deployment
--------------	-------------------	------------------------	--------------------------	-----------------	---------------------------



Manager



DBA



BA



Developer



UX



QA

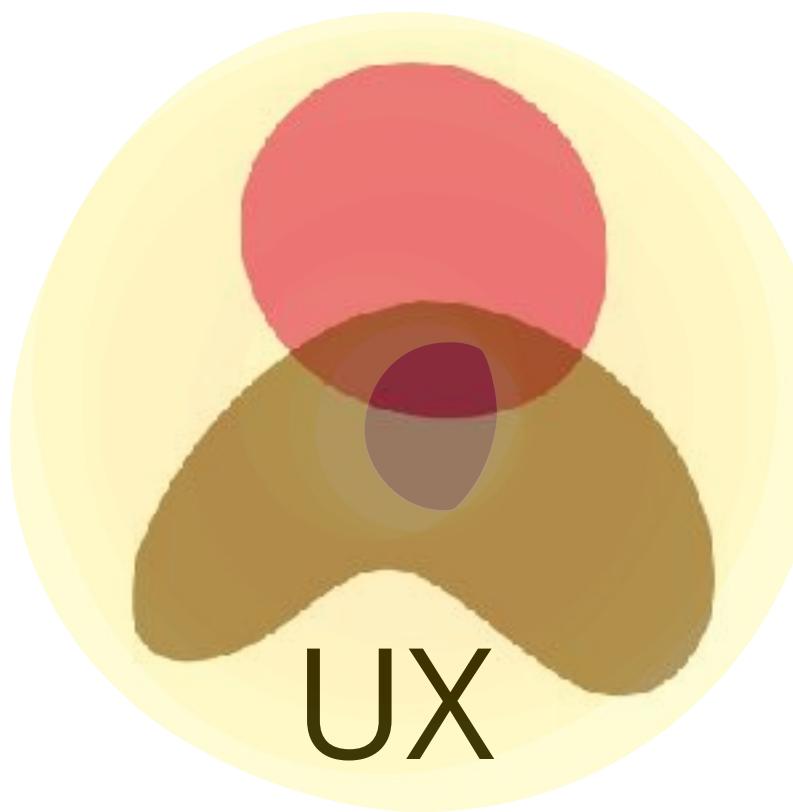
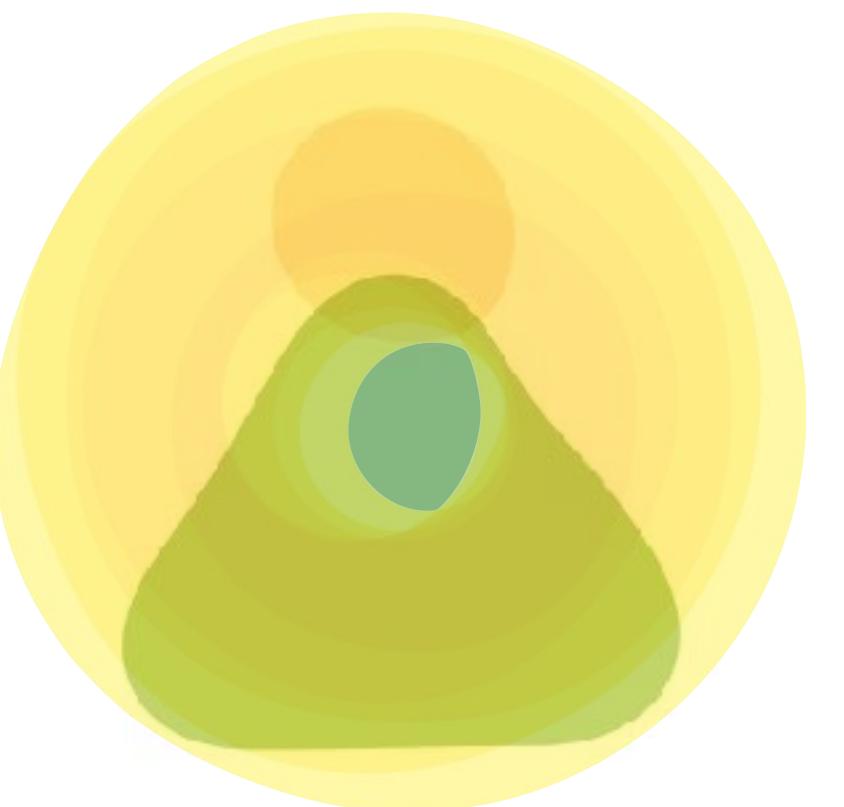


Operations

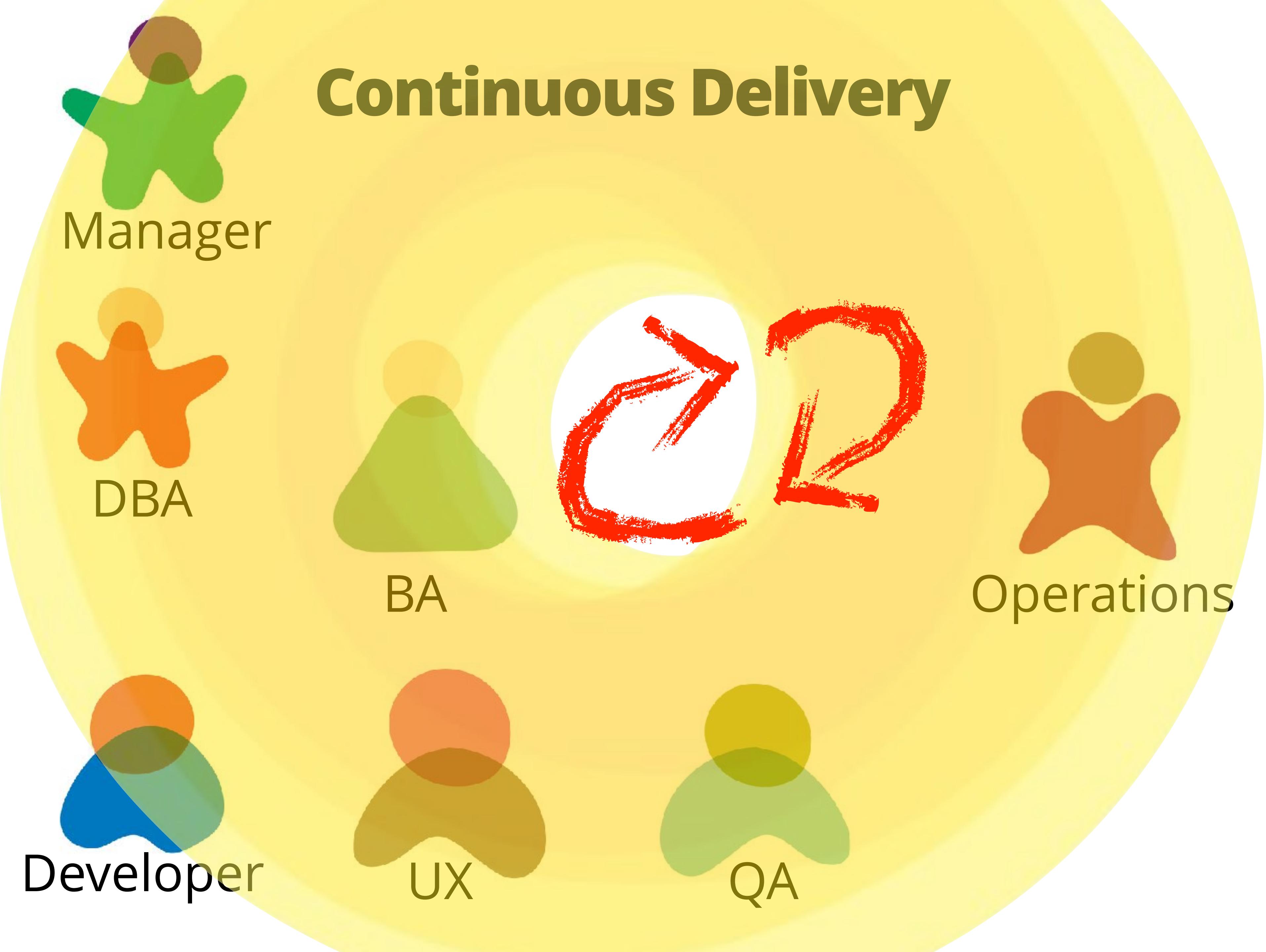
Who are You?



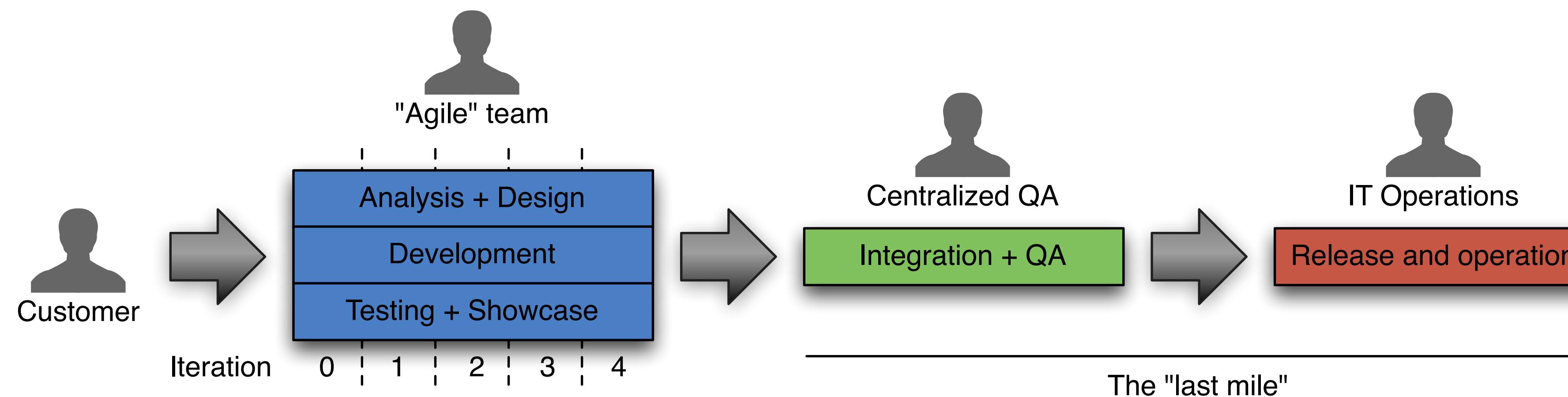
Agile Transformation

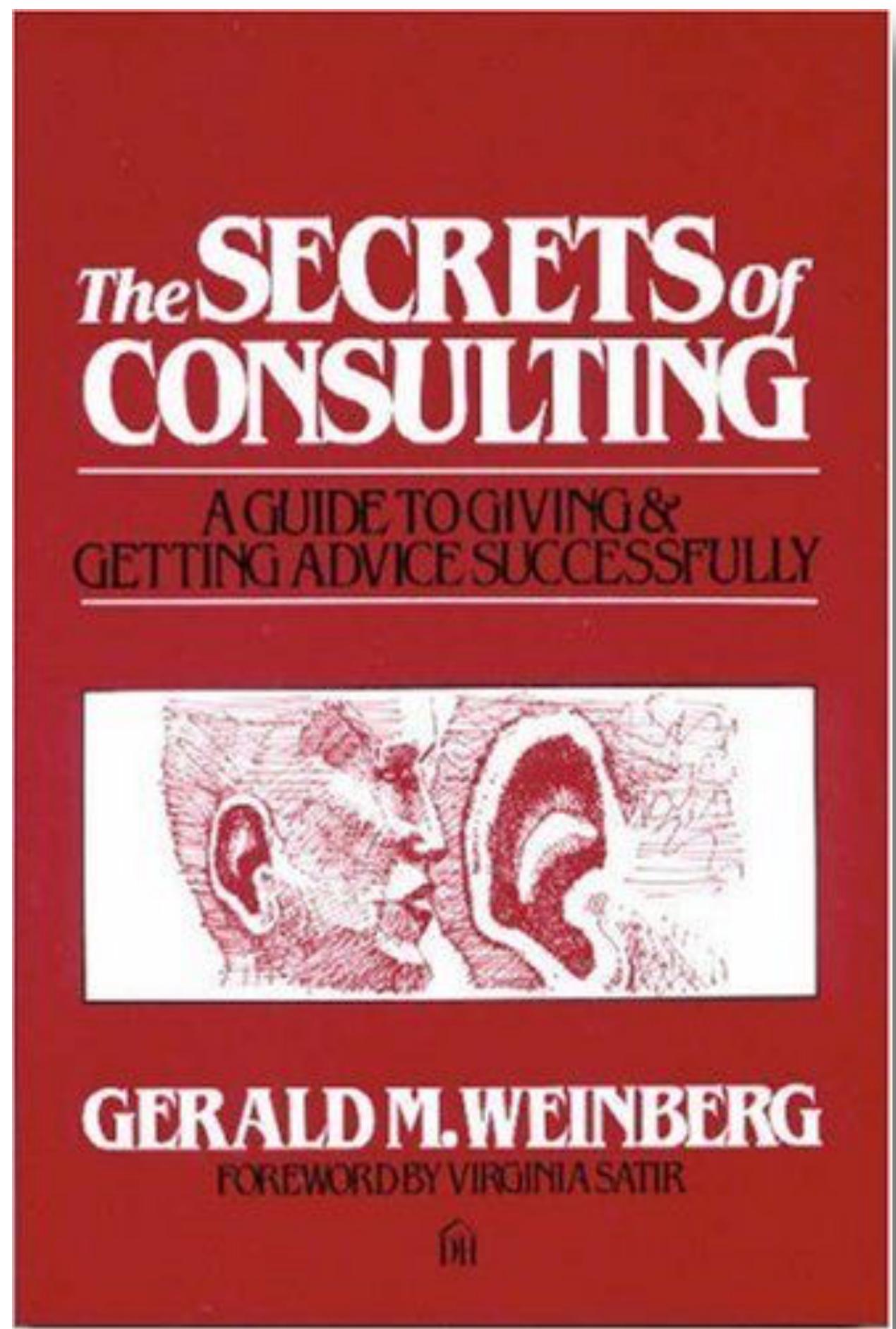


Continuous Delivery



Agile 101





“ *No matter how it looks at first, it's always a people problem.* ”

Continuous Integration

Integration early and often.

Everyone checks into **trunk** at least once a day.

Bring the pain forward.

eager vs. Late

pain



time

Continuous Integration

Fast, automated feedback on
the correctness of your
application every time there
is a change to code

Integration

Integration early and often.

Everyone checks into **trunk** at least once a day.

Continuous Deployment

Deploy as the final stage of continuous integration.

Integration

Integration early and often.

Everyone checks into **trunk** at least once a day.

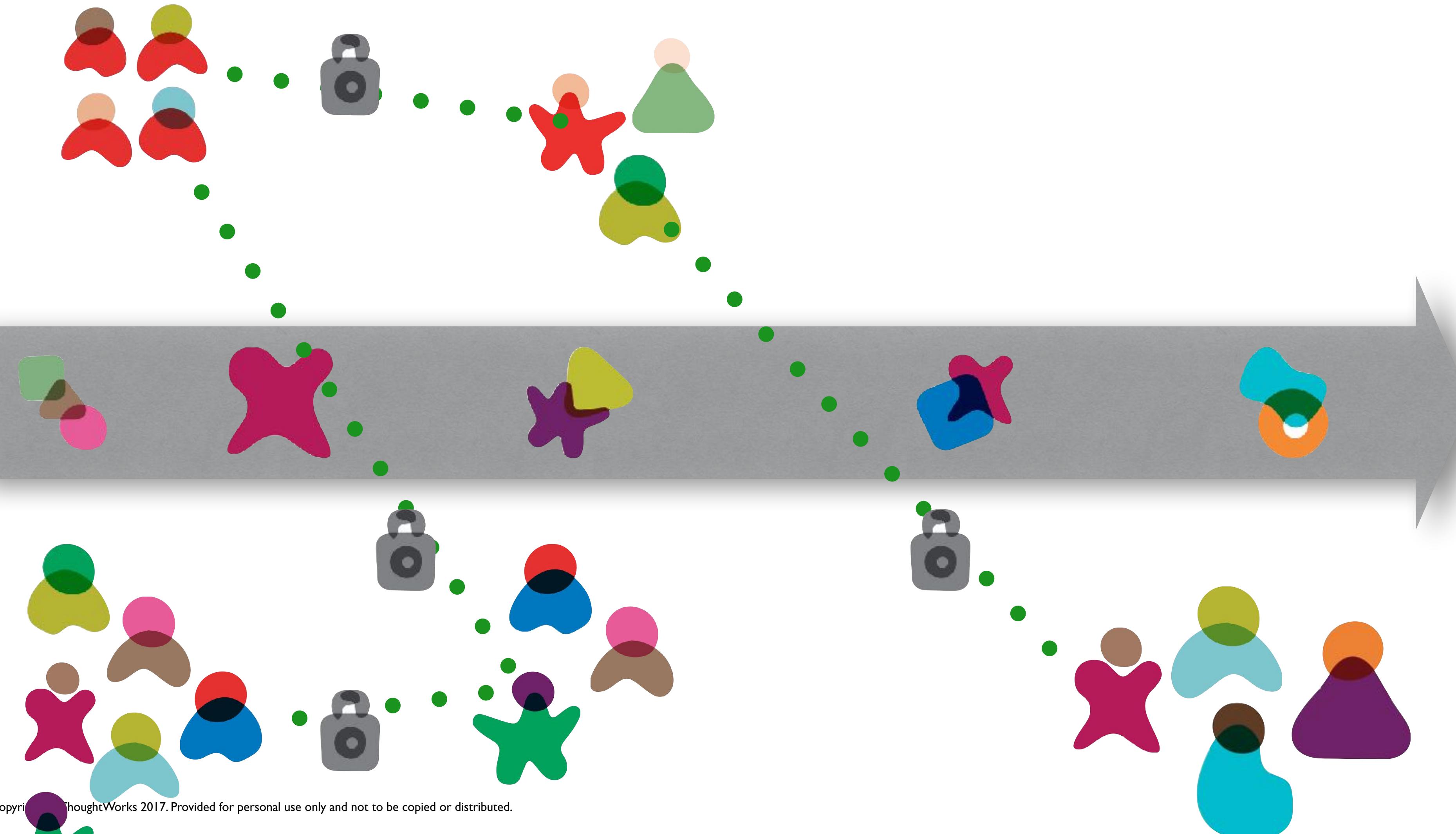
Deployment

Deploy as the final stage of continuous integration.

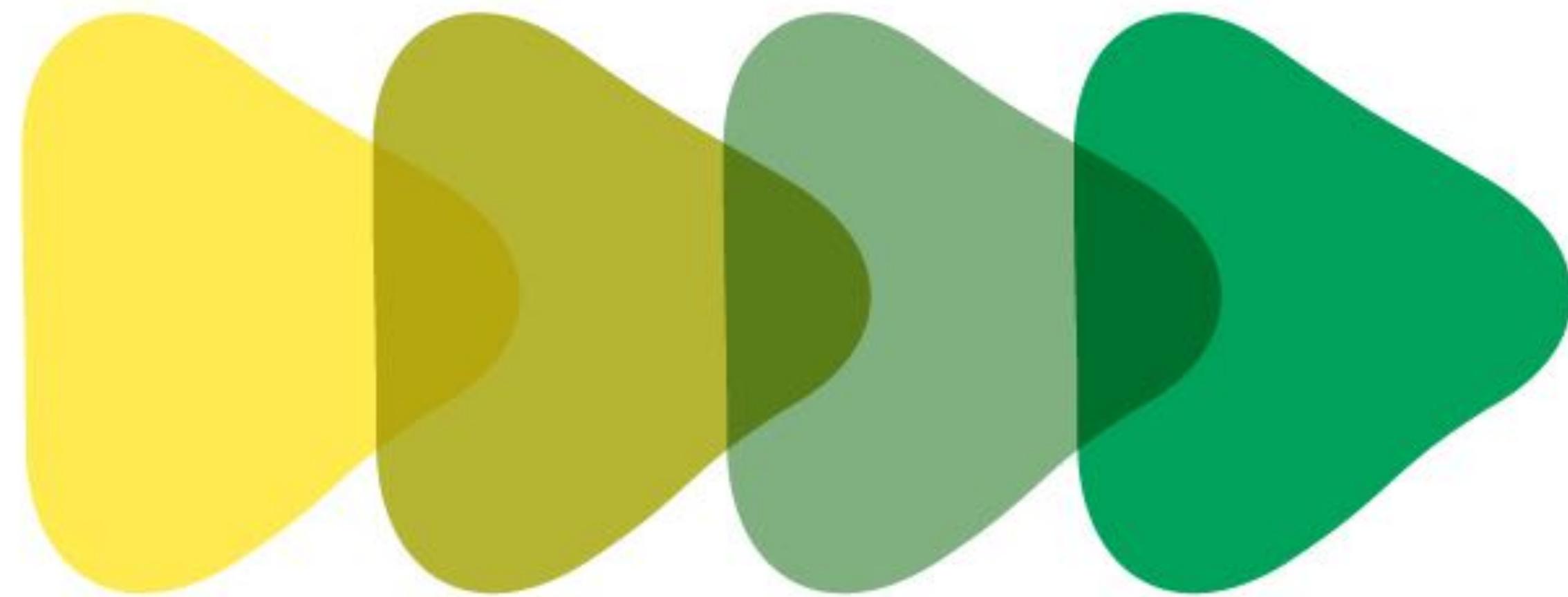
Continuous Delivery

Software is always in a deployable state.

So what's your plan?



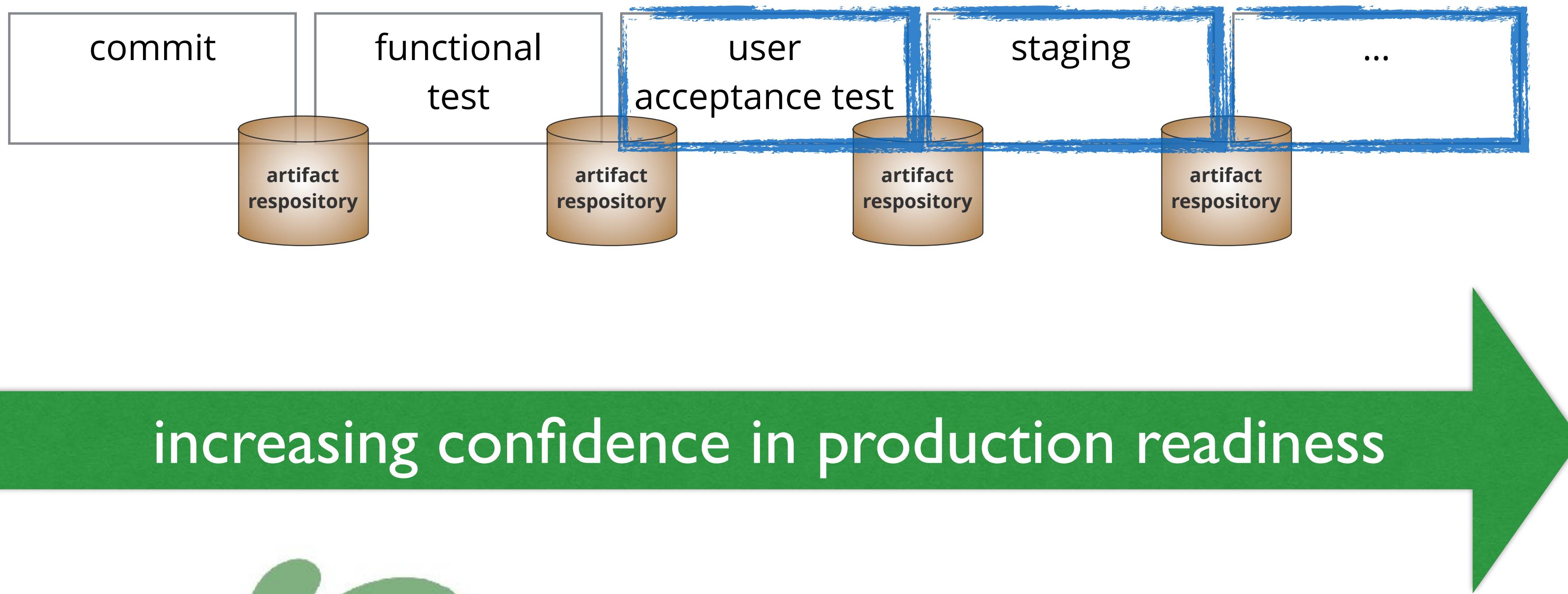
Deployment Pipelines



Deployment Pipeline

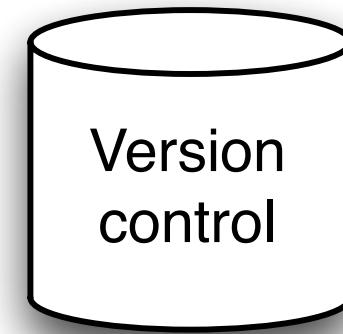
Fast, automated feedback
on the **production readiness**
of your application every
time there is a change — to
code, infrastructure or
configuration

Pipeline Construction

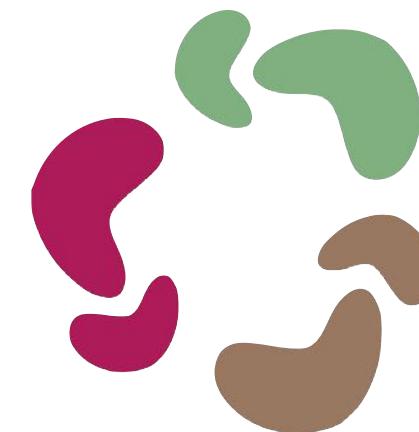


Pipeline stages = feedback opportunities

commit Stage



source code
commit tests
build scripts



Run against each check-in

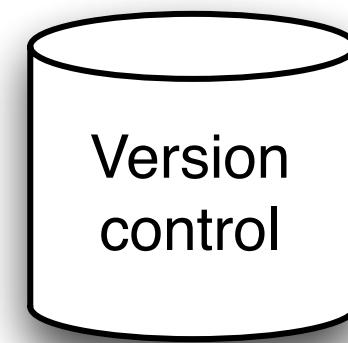


Starts building a release candidate

If it fails, fix it immediately

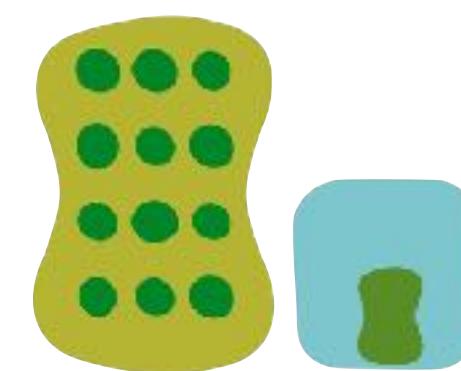
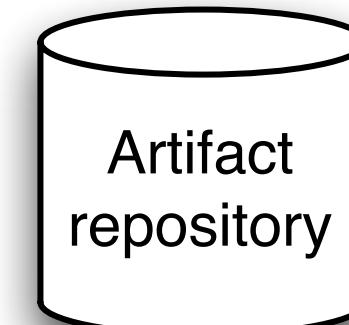


UAT Stage

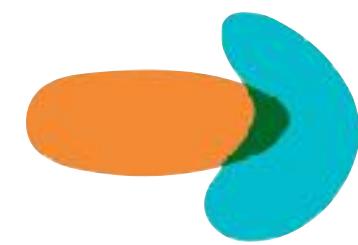


acceptance tests
deployment scripts
configuration data

test reports
metadata



End-to-end tests in production-like environment

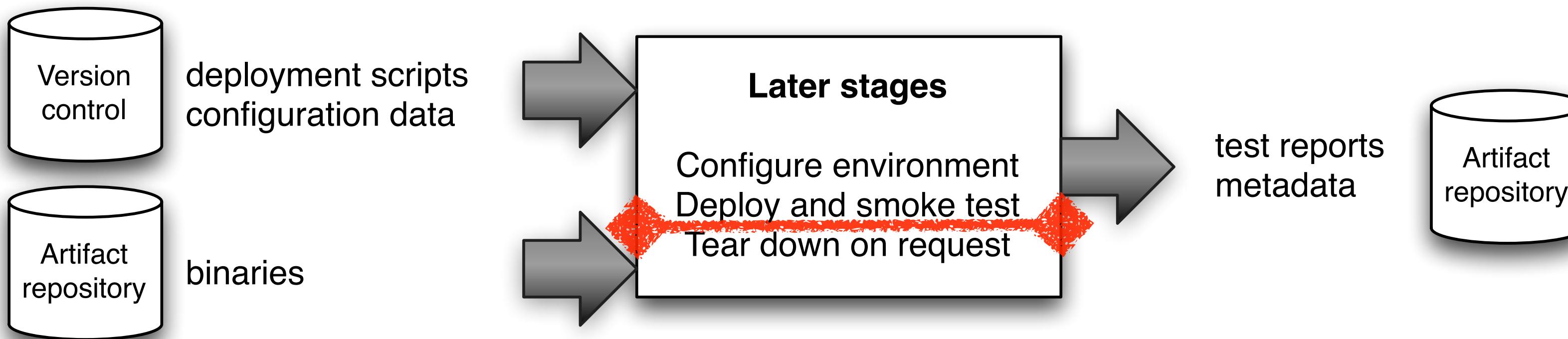


Triggered when upstream stage passes

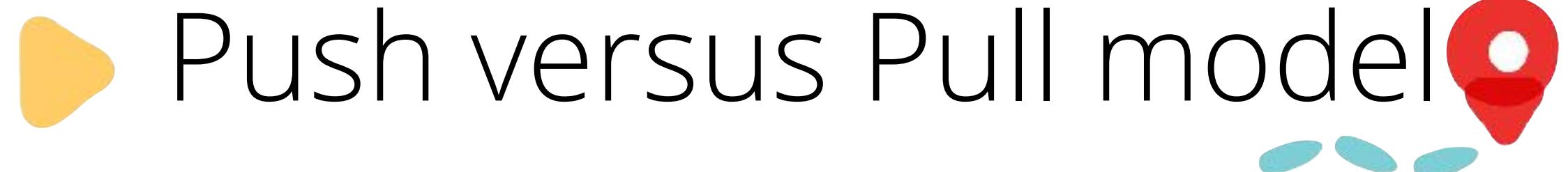


First DevOps-centric build

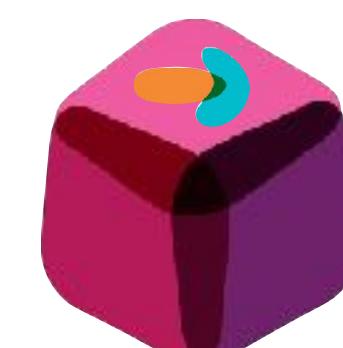
Manual Stage

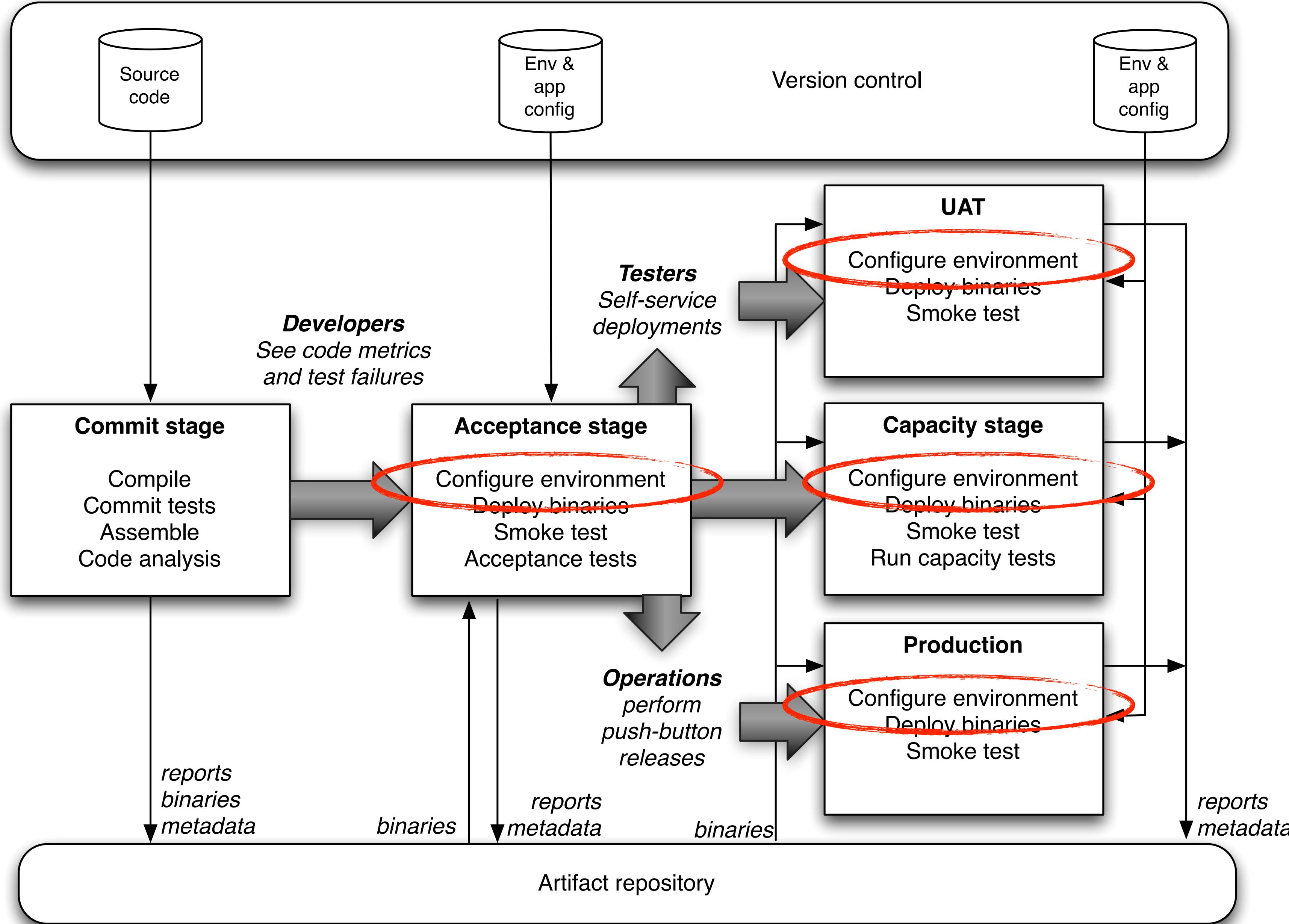


UAT, staging, integration, production, ...



Deployments self-serviced through
push-button process





Machinery

continuous integration ++



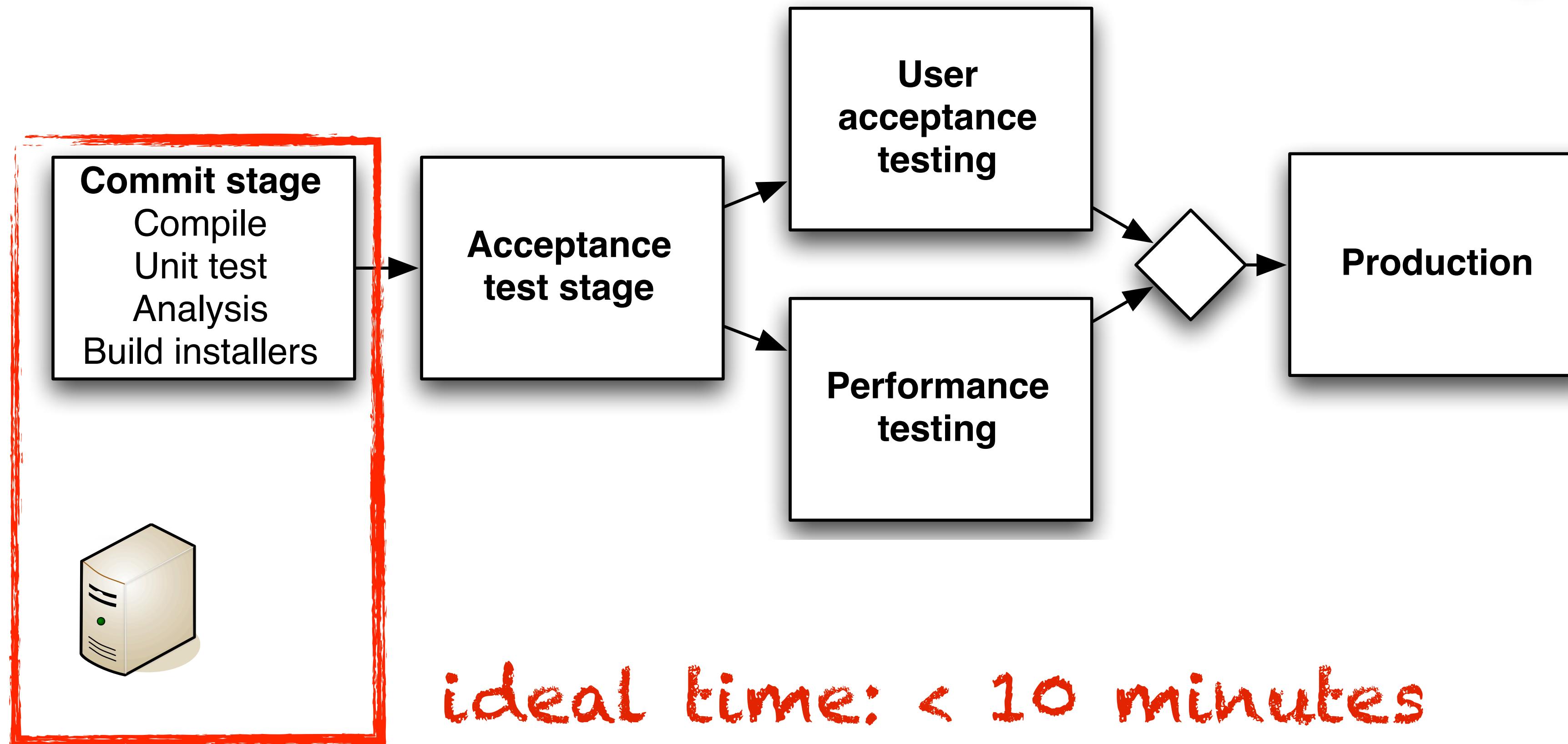
Jenkins



www.thoughtworks.com/products/go-continuous-delivery

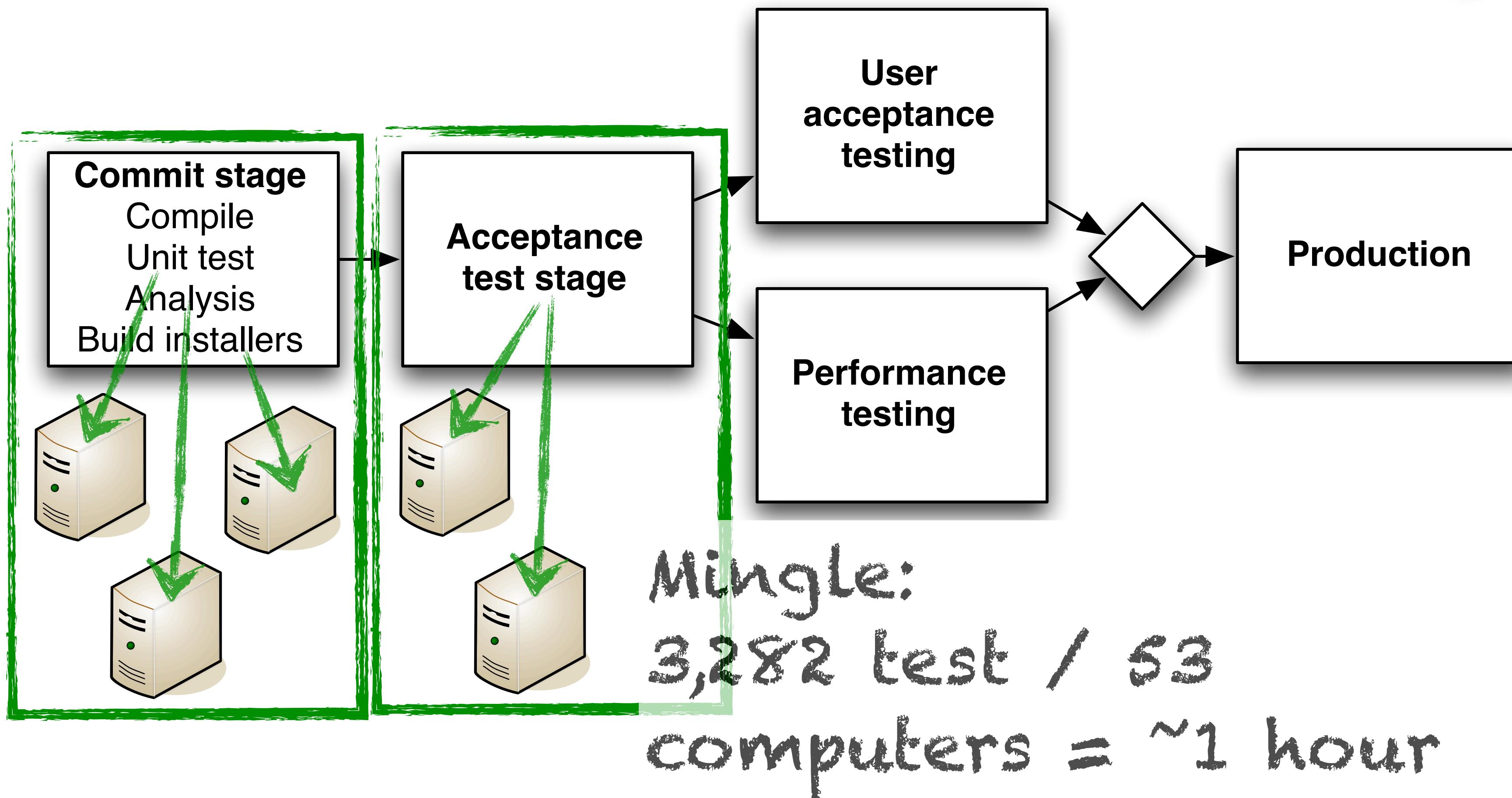
Pipeline Anti-patterns

insufficient parallelization



Pipeline Anti-patterns

insufficient parallelization



Insufficient Parallelization Heuristic:

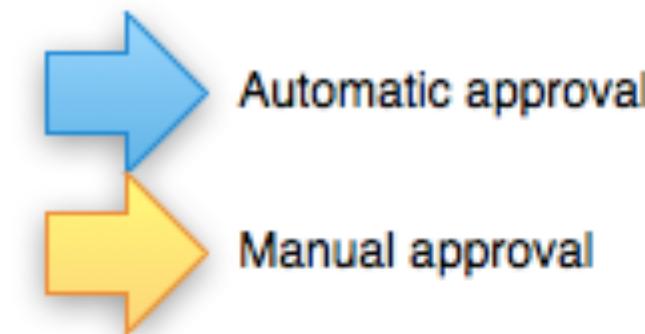
make your pipeline *wide*, not *long*

parallelize each stage as much as you can

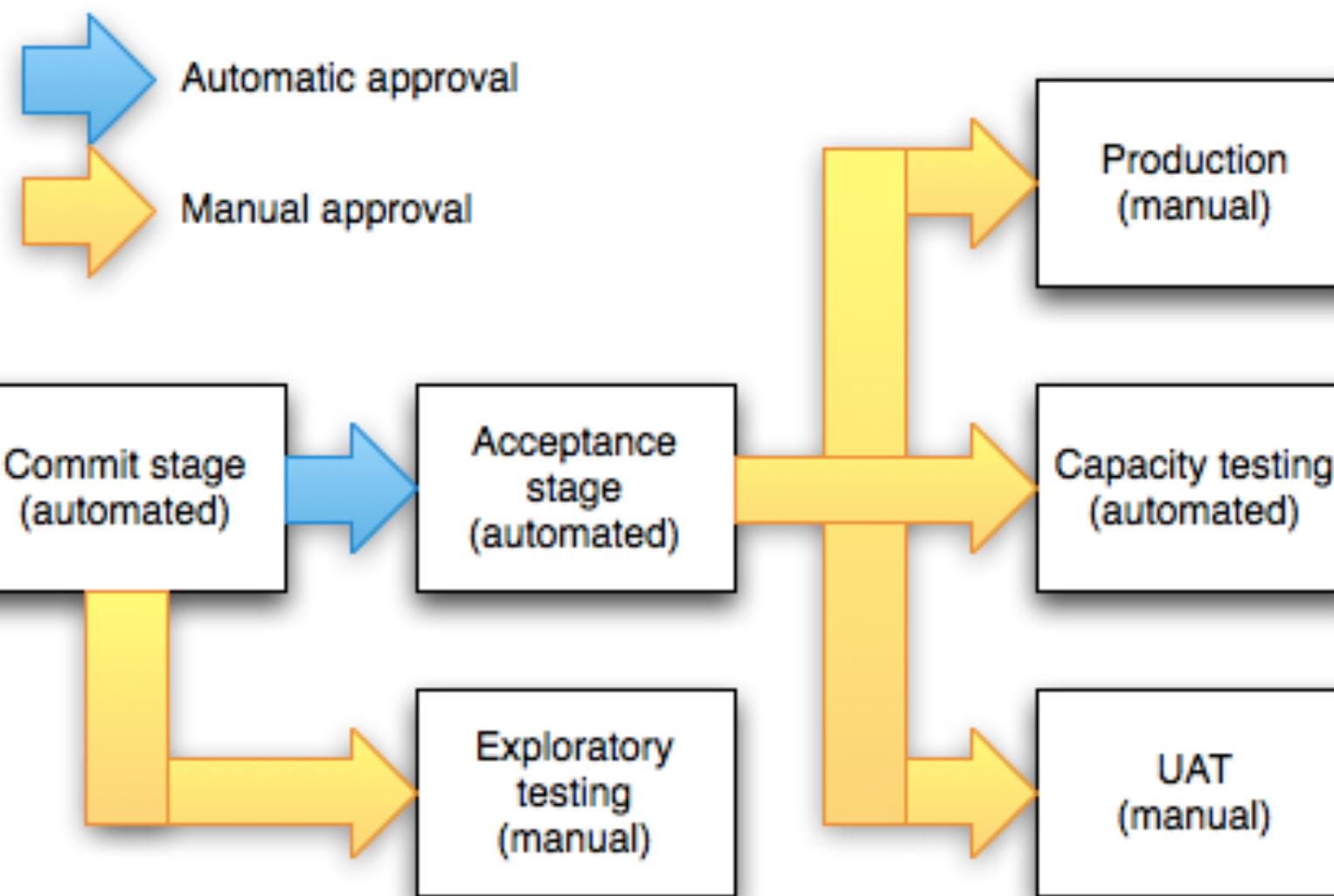
create more stages if necessary to optimize feedback



Pipeline Anti-patterns



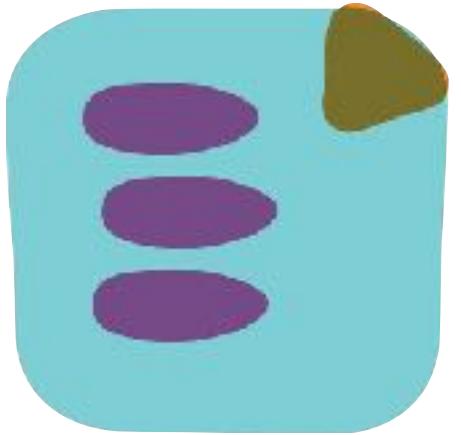
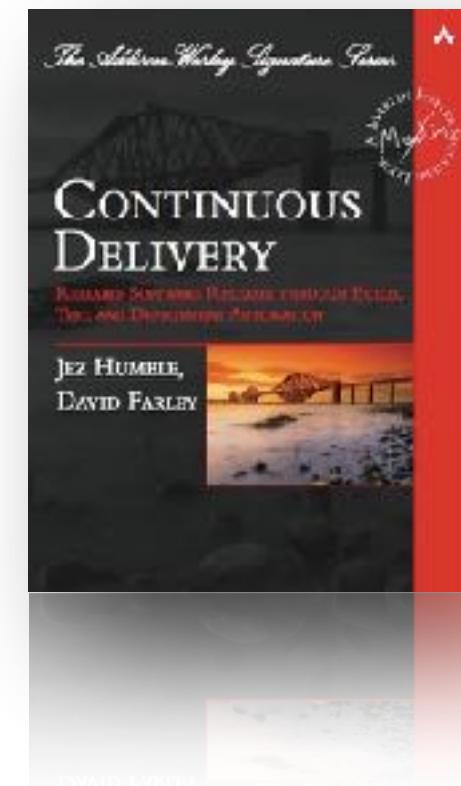
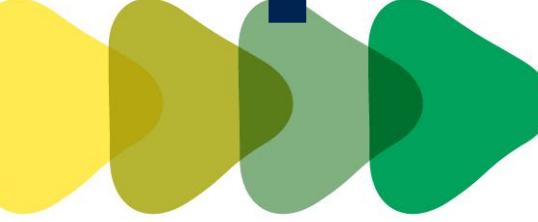
inflexible workflow



pipeline fans out
as soon as it
makes sense to do
so

automate almost everything

Principles



keep everything you need to build, deploy,
test, & release in version control

- requirements documents
- test scripts
- automated test cases
- network configuration scripts
- technical documentation
- database creation, upgrade, downgrade, and initialization scripts
- application stack configuration scripts
- libraries
- deployment scripts
- tool chains

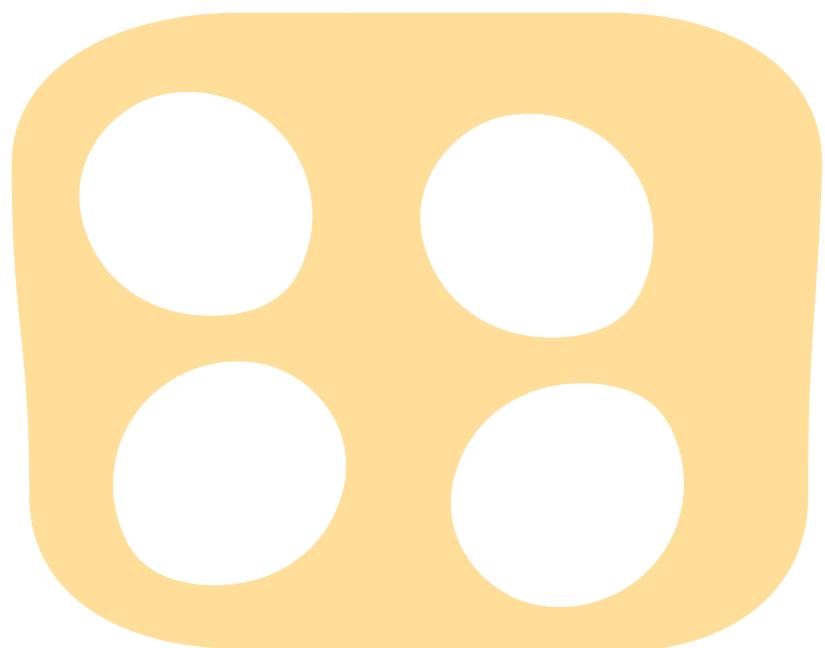
Infrastructure Consistency



Chef

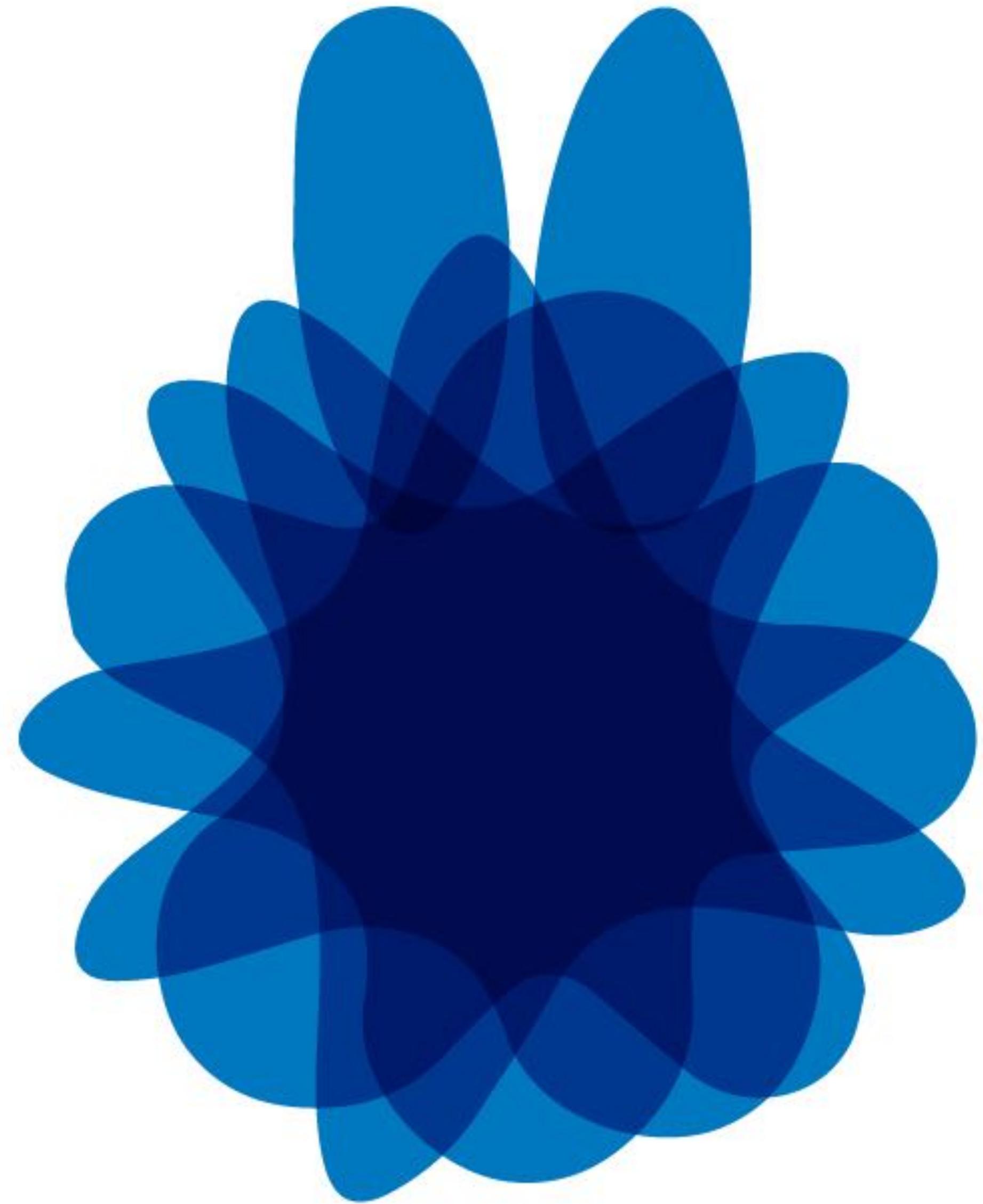


ANSIBLE



boxen.github.com

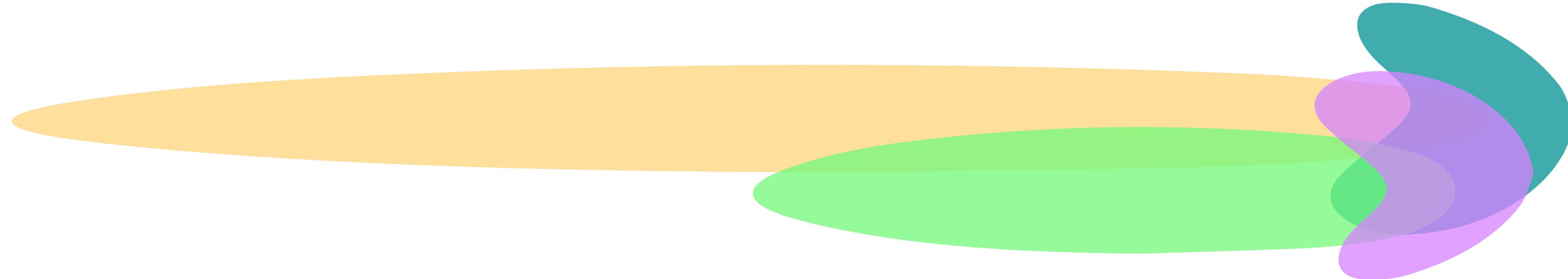
Identify & remove friction



Continuous Delivery Metrics

lead time

the time between the initiation and completion of a production process.



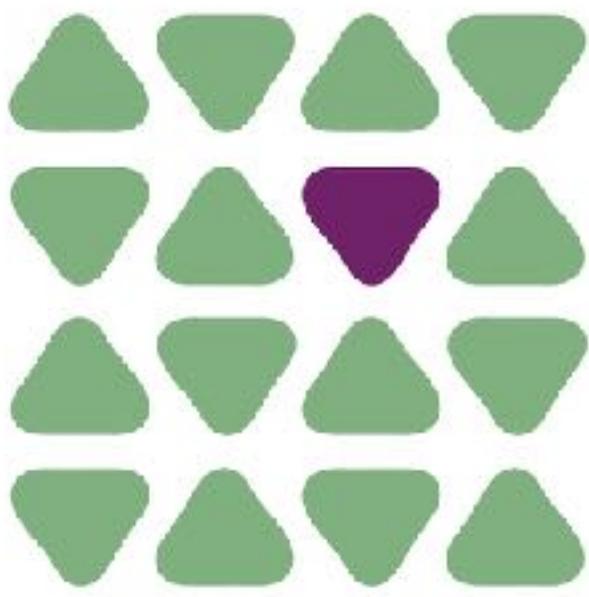
cycle time

the total elapsed time to move a unit of work from the beginning to the end of a physical process

Potential Hindrances

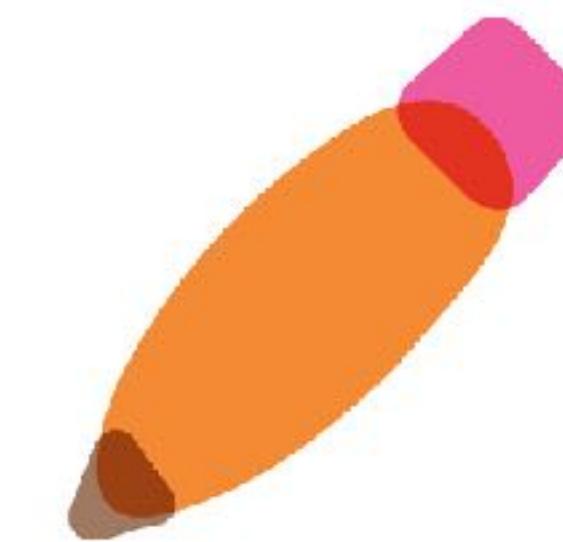
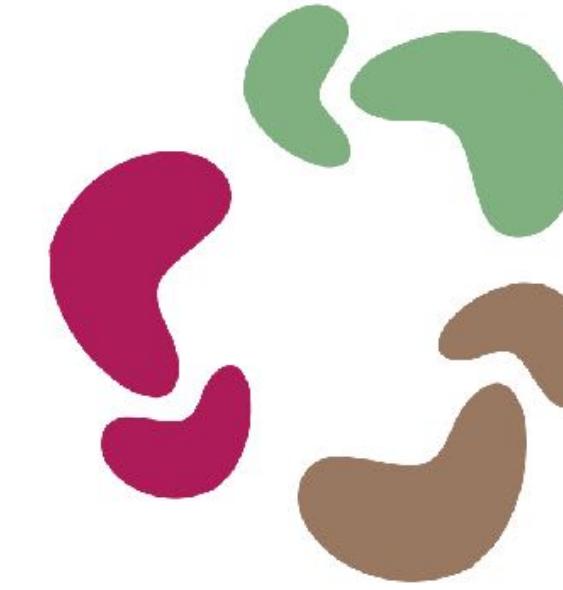


Prerequisites



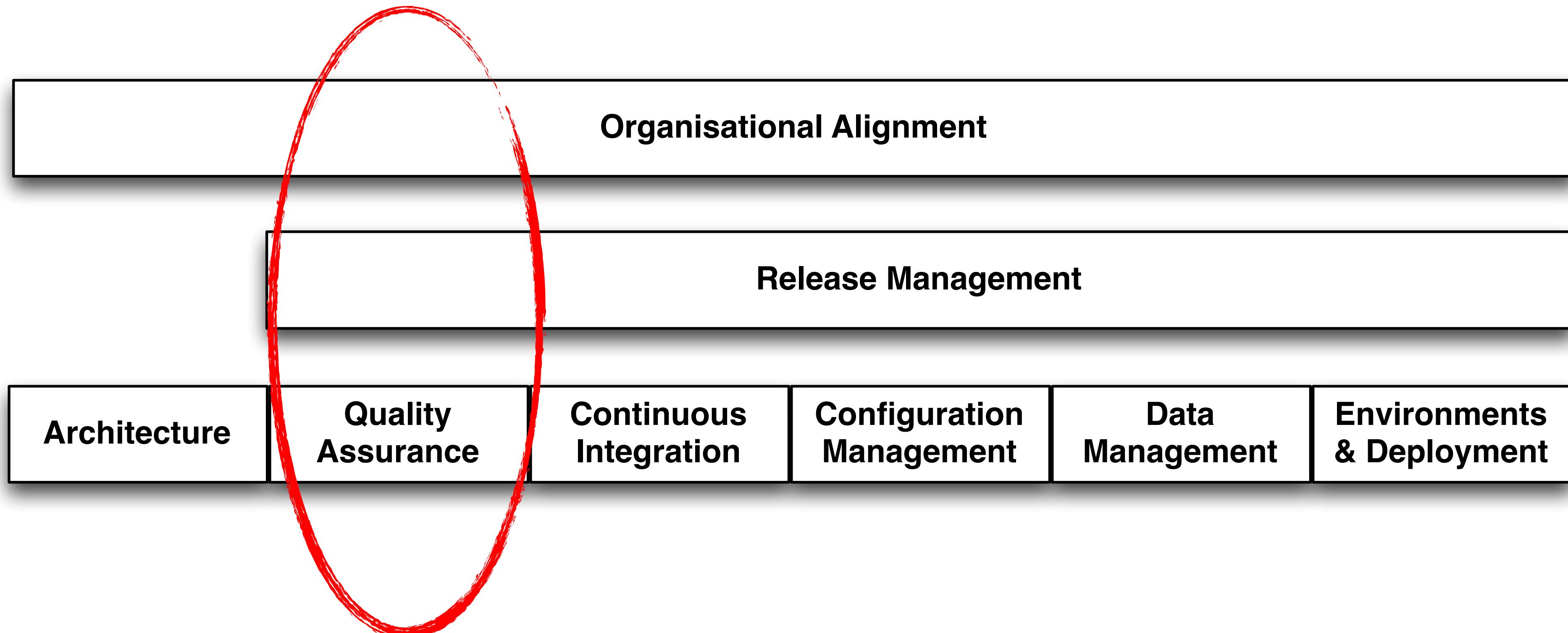
comprehensive
configuration management

continuous integration

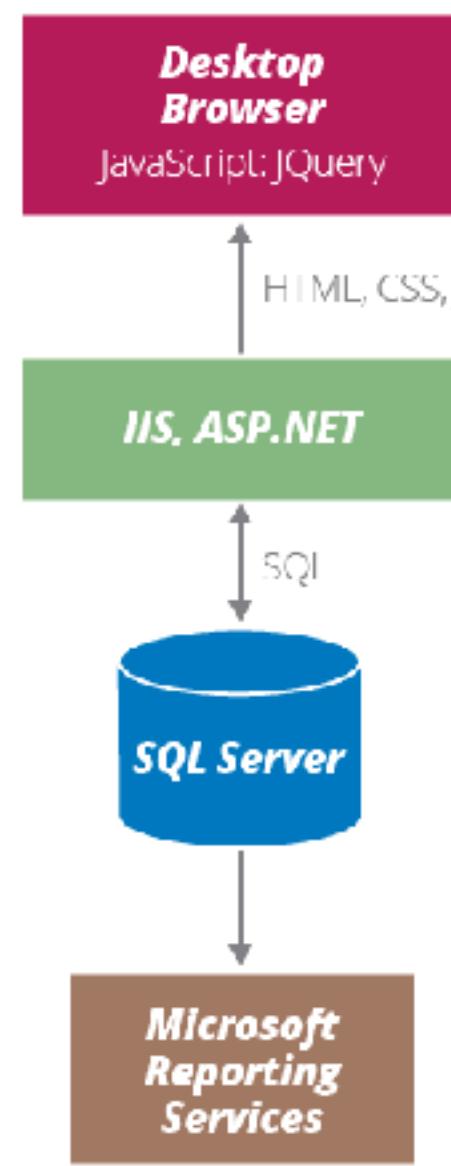


excellent automated testing at all levels

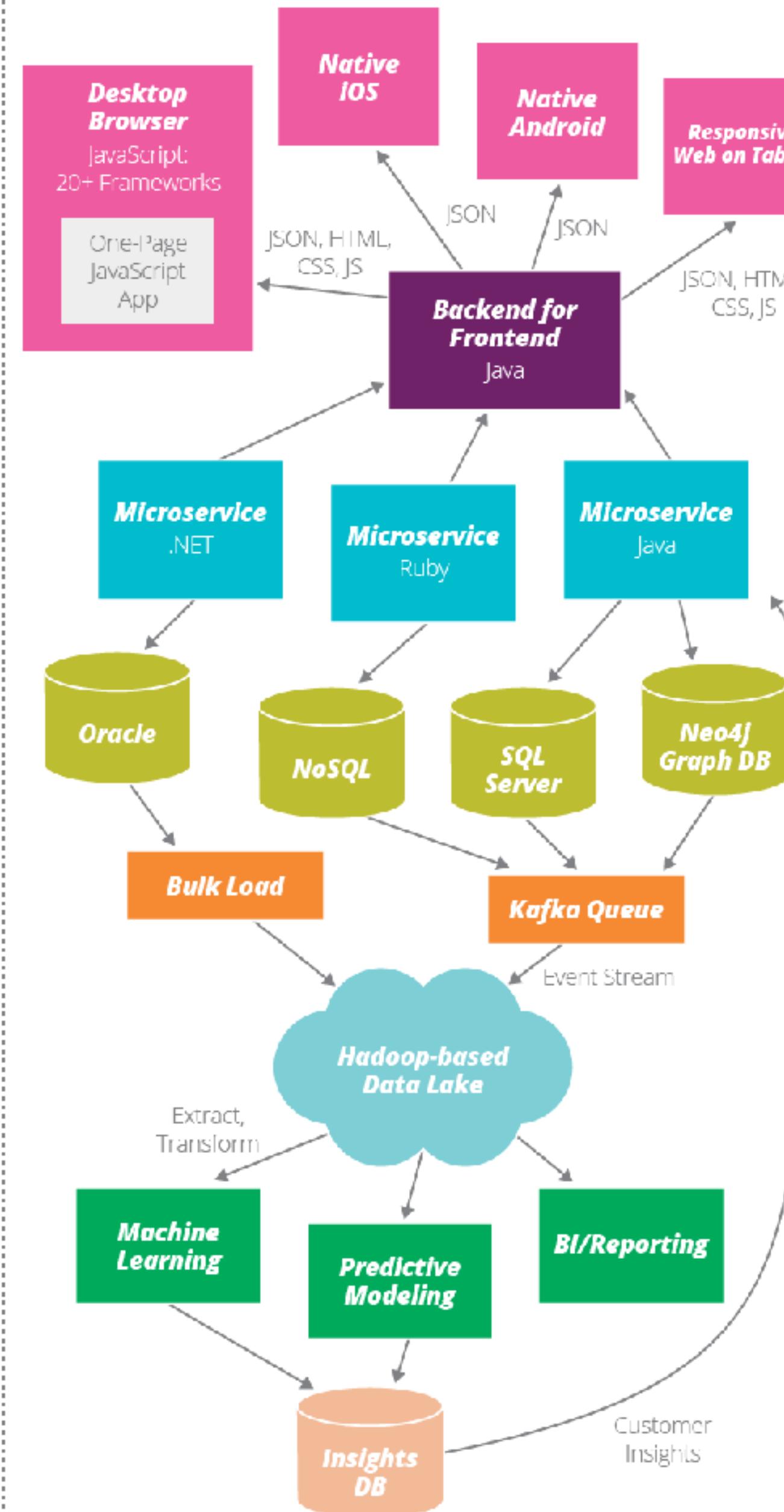
Continuous Delivery



2005



2016



**Modern
software
is complex!**

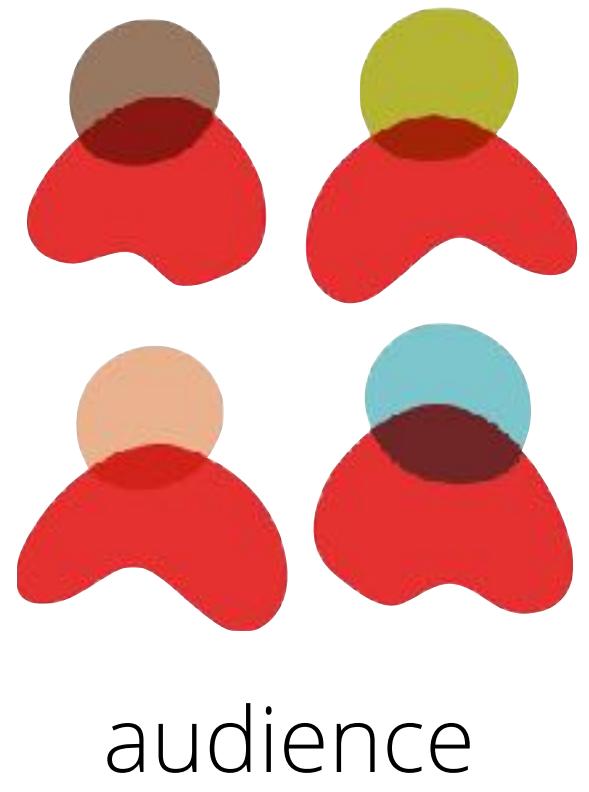


feedback



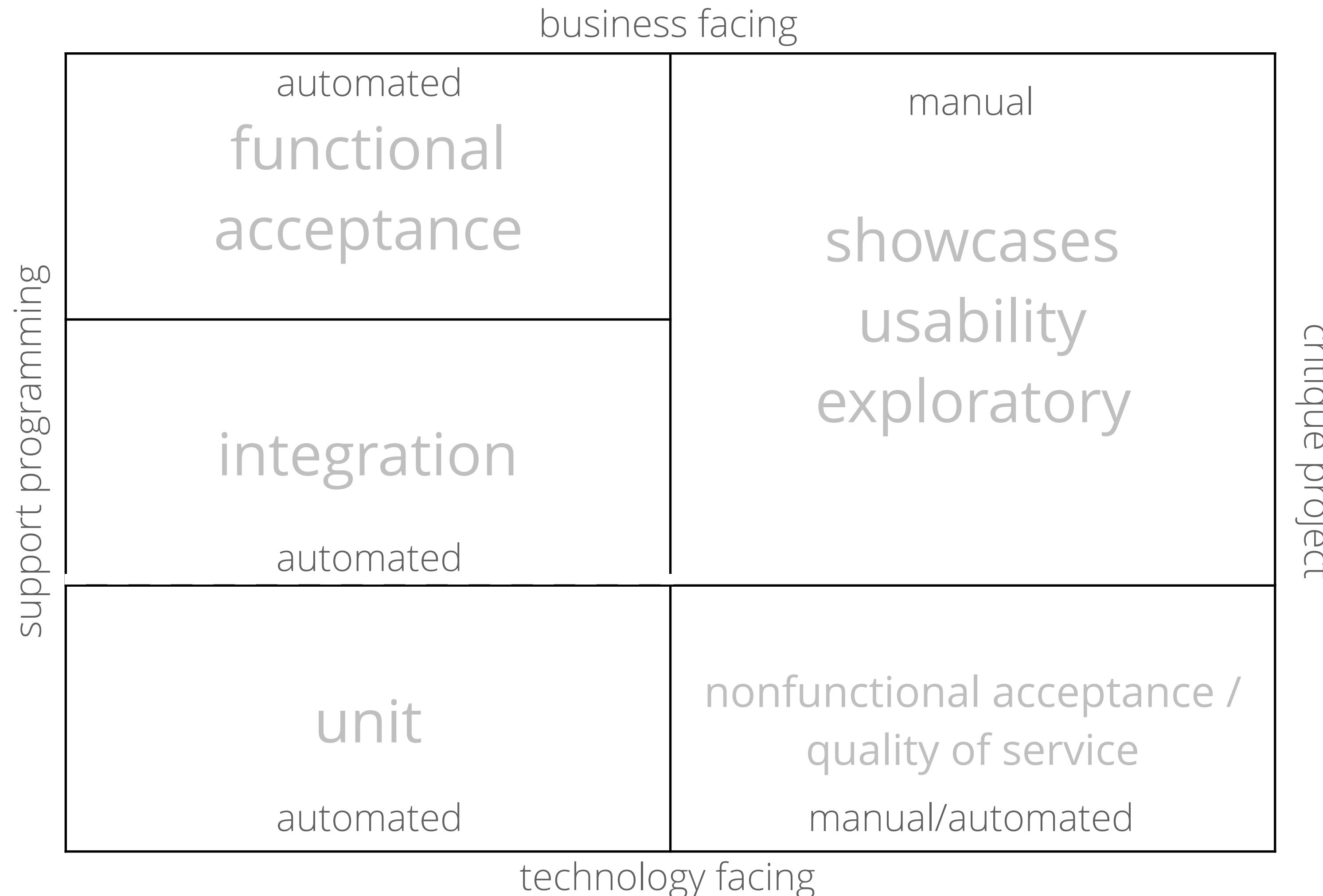
effort

Testing



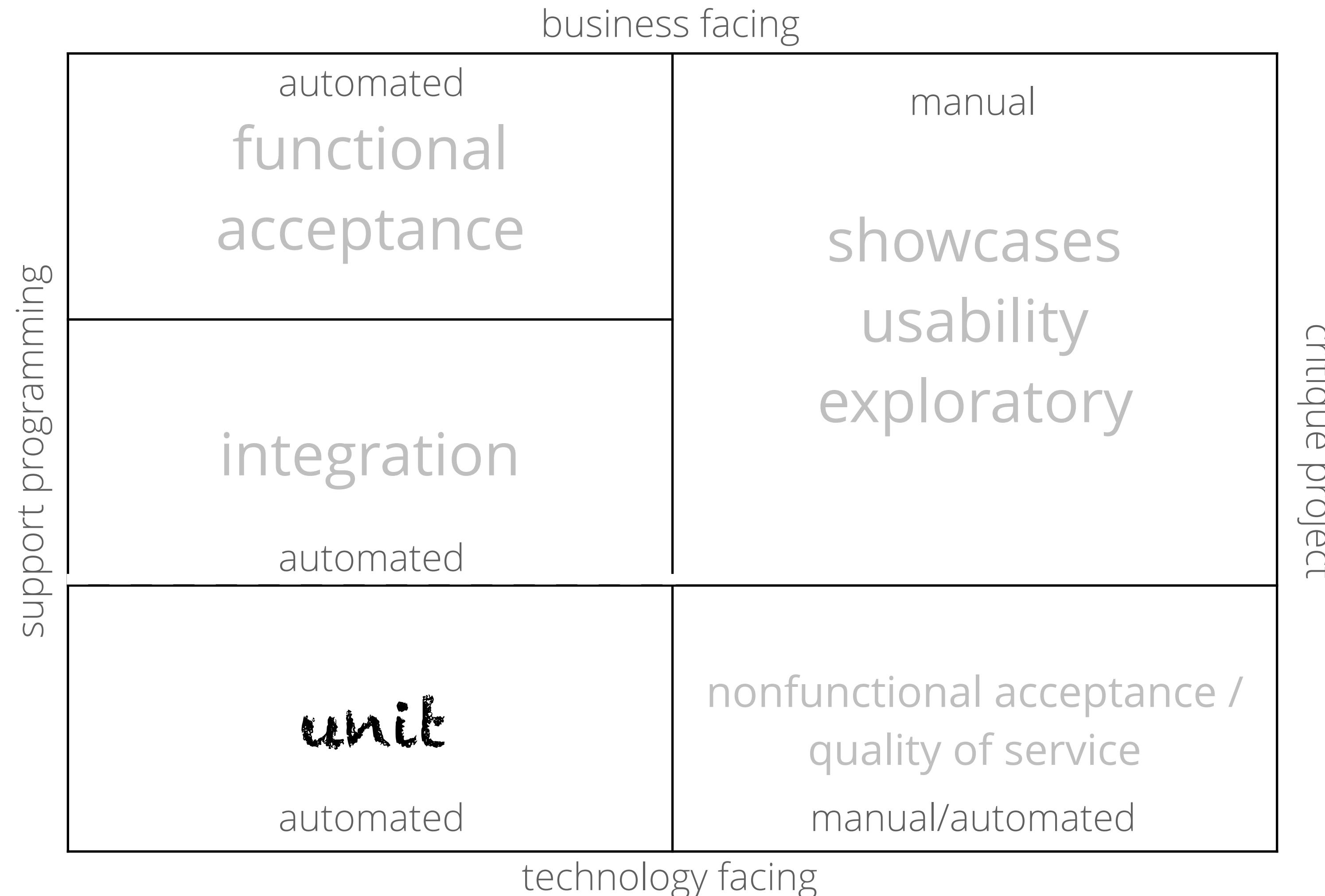
audience

Testing Quadrants



source: Brian Marrick, *Continuous Delivery* (Humble/Farley), with modifications

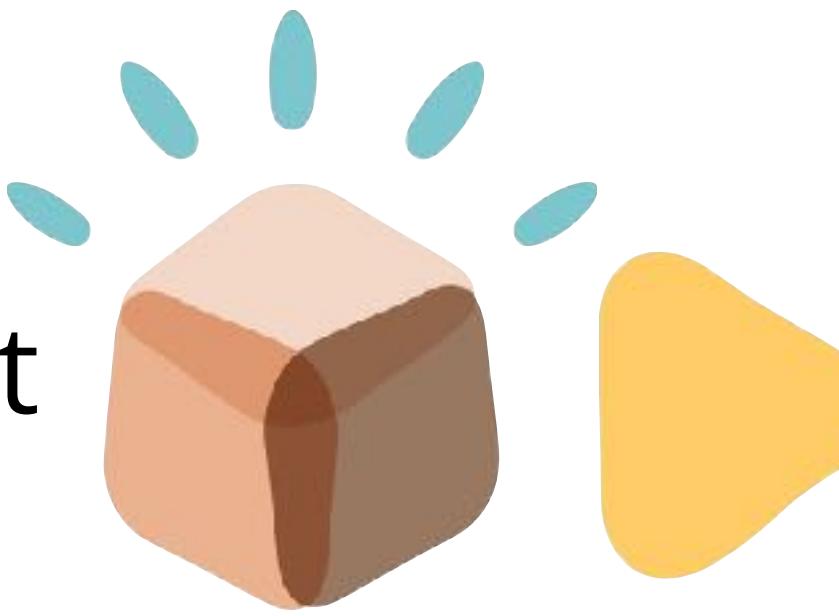
Testing Quadrants



source: Brian Marrick, *Continuous Delivery* (Humble/Farley), with modifications

~~unit~~ Testing

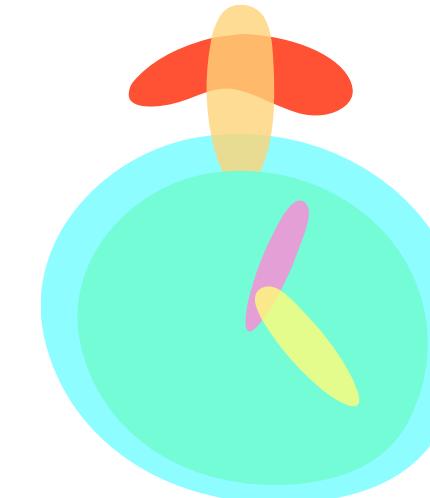
prefer test-driven to test-after development



prefer pragmatism over dogmatic metrics



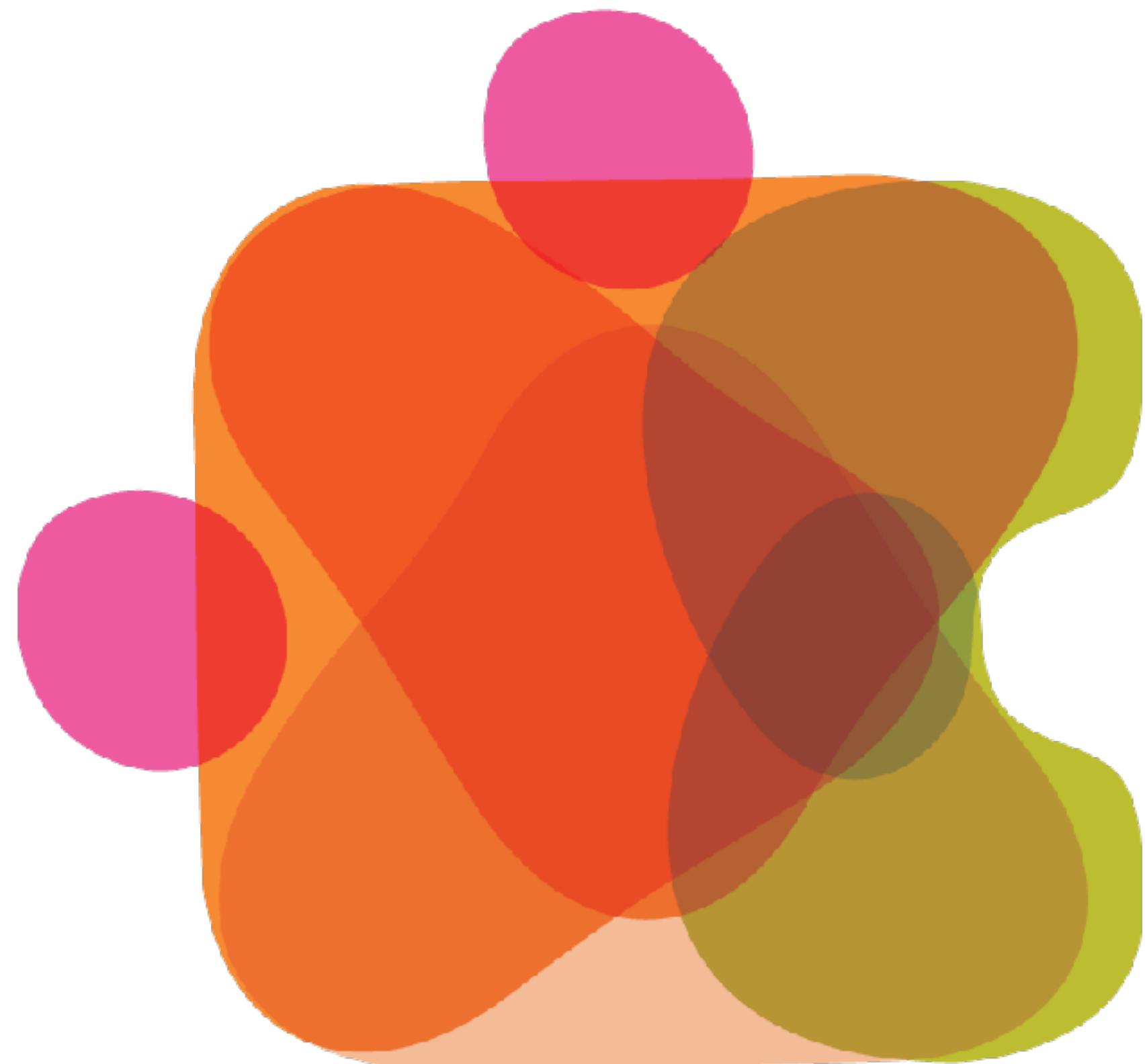
optimize for the target audience



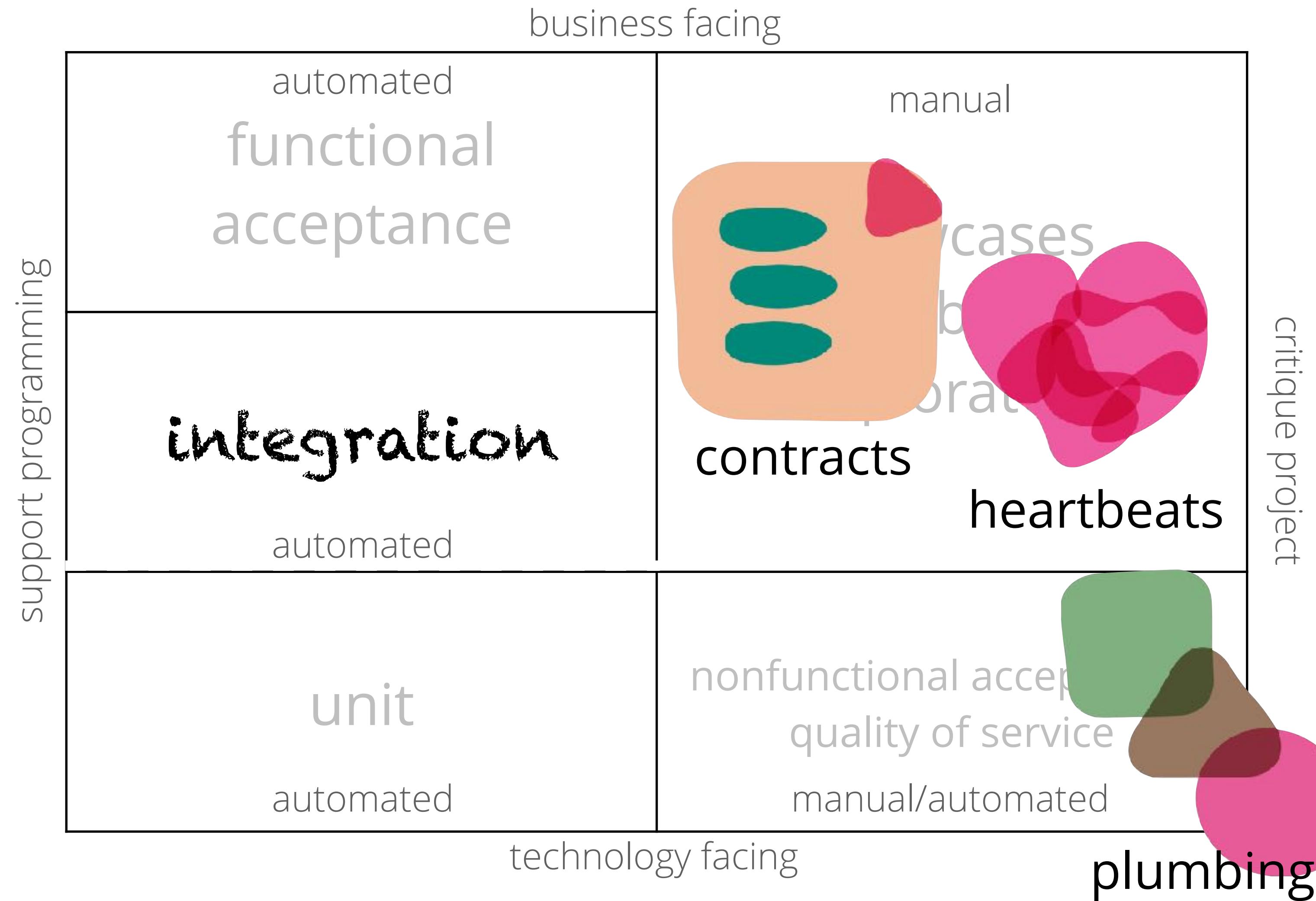
how much time?

Common Anti-pattern

mixed unit/
functional tests

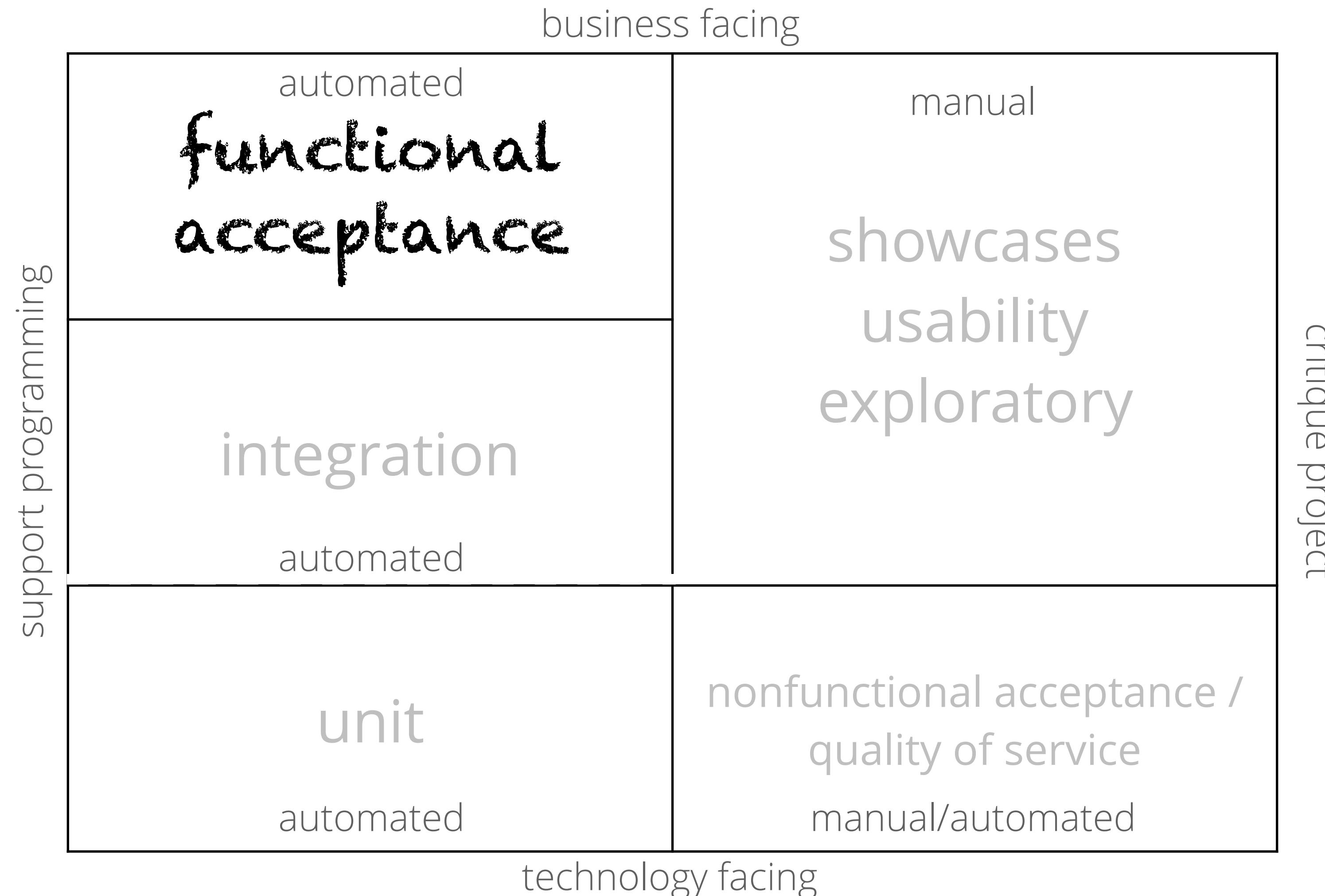


Testing Quadrants

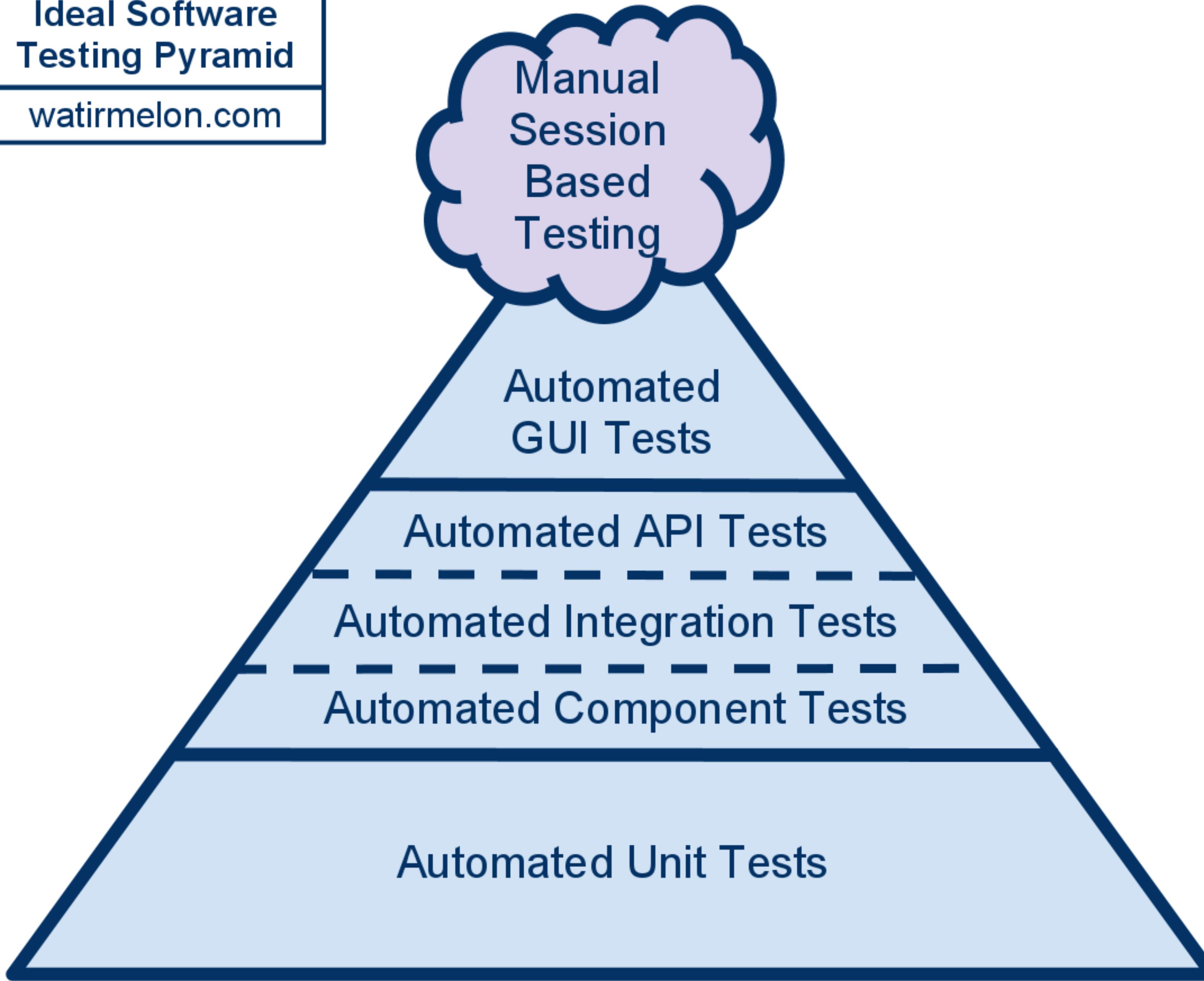


source: Brian Marrick, *Continuous Delivery* (Humble/Farley), with modifications

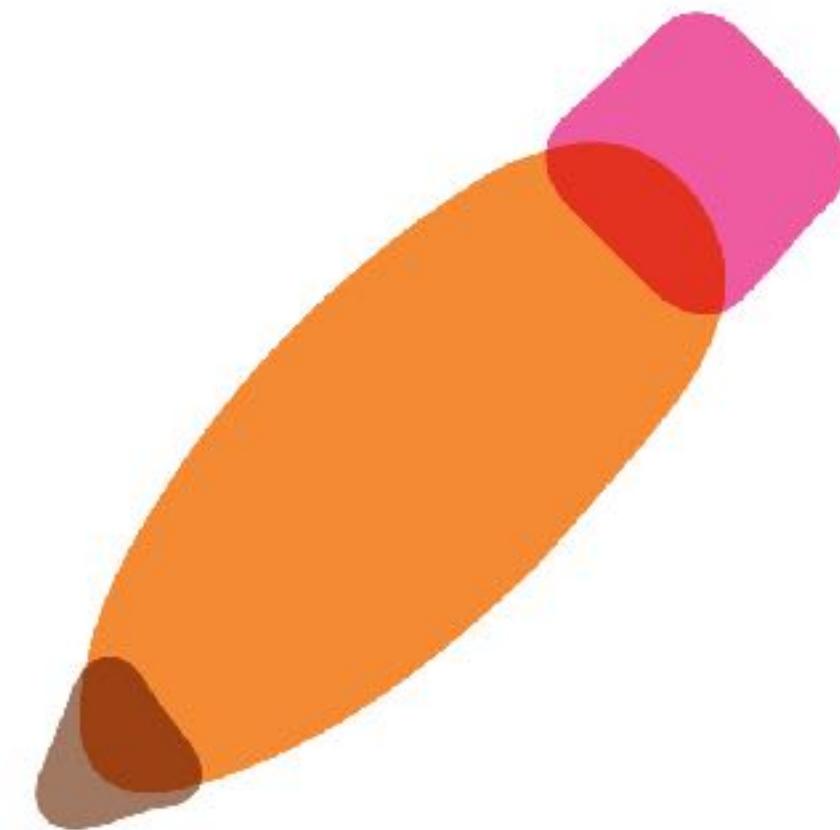
Testing Quadrants



source: Brian Marrick, *Continuous Delivery* (Humble/Farley), with modifications



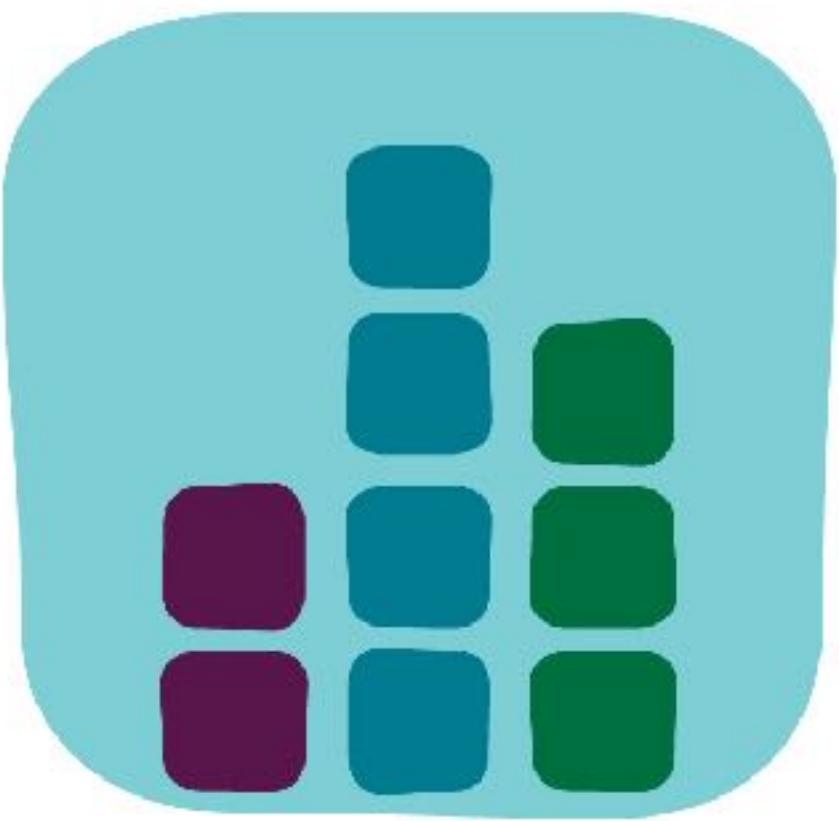
To Cuke or not to Cuke...



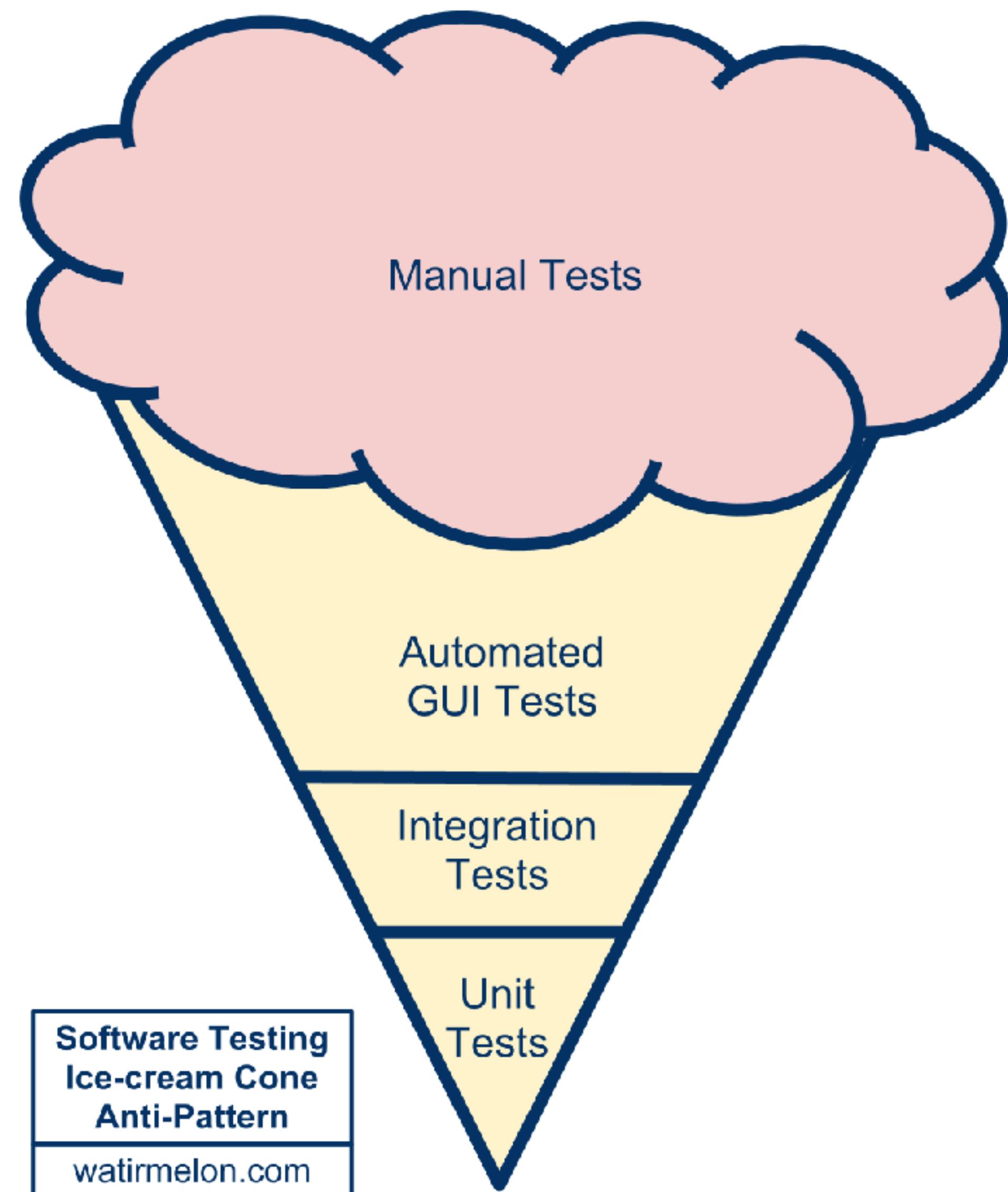
BDD ≠



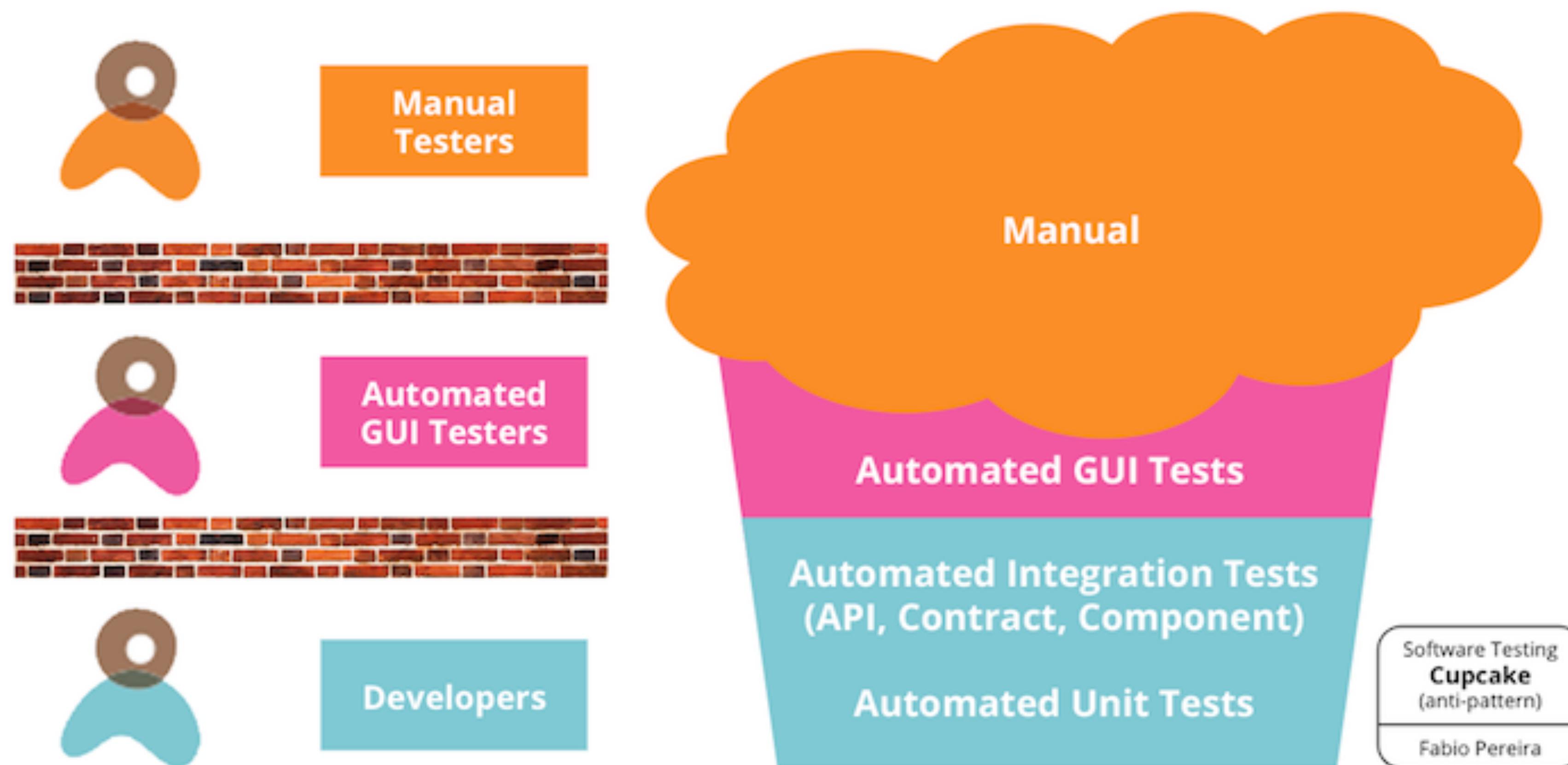
*use when it provides useful
feedback to the target audience*



Anti-pattern: Ice-cream Cone



Cupcake Anti-pattern



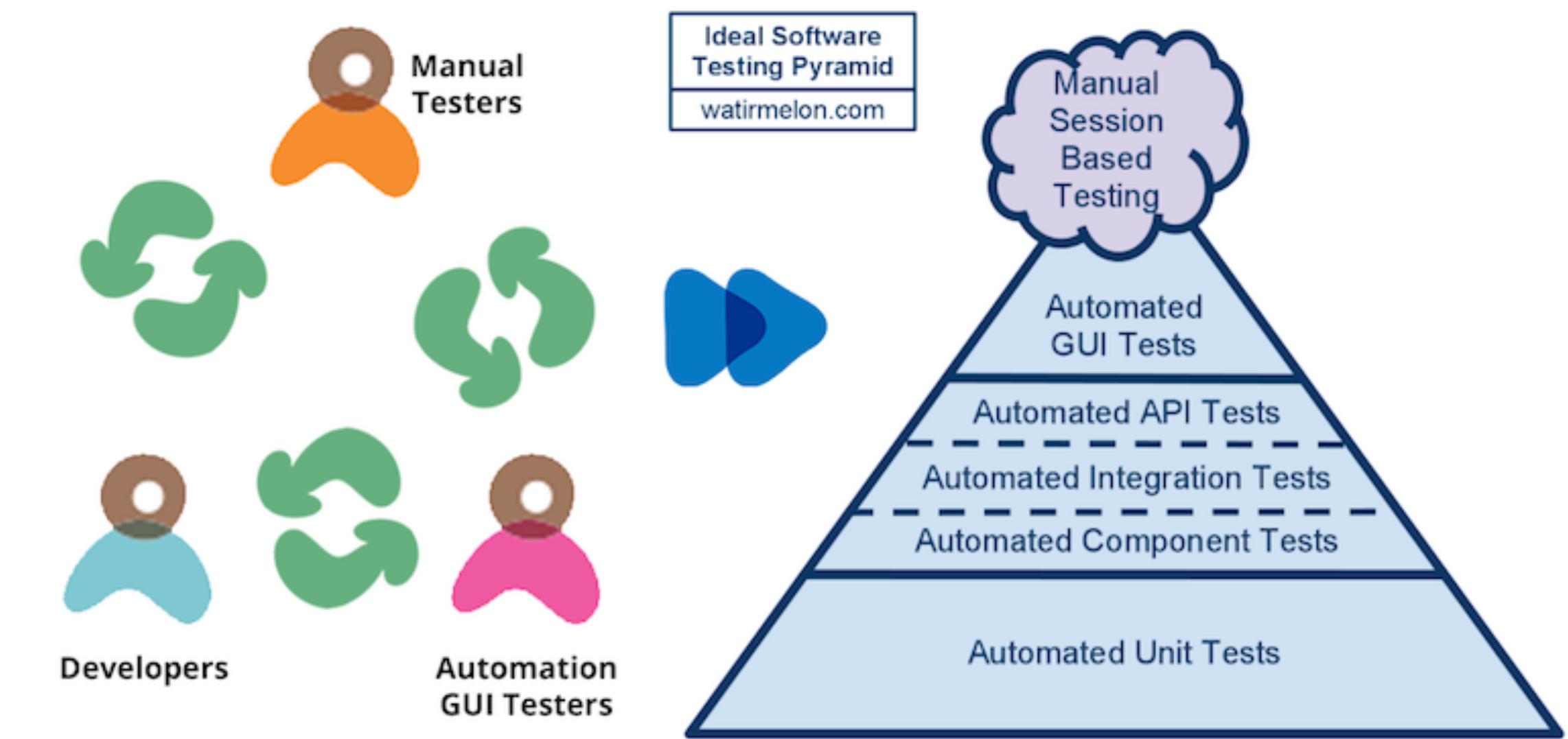
Avoiding Cupcakes

collaborate
work in sync
cross-role pair programming
story kickoff

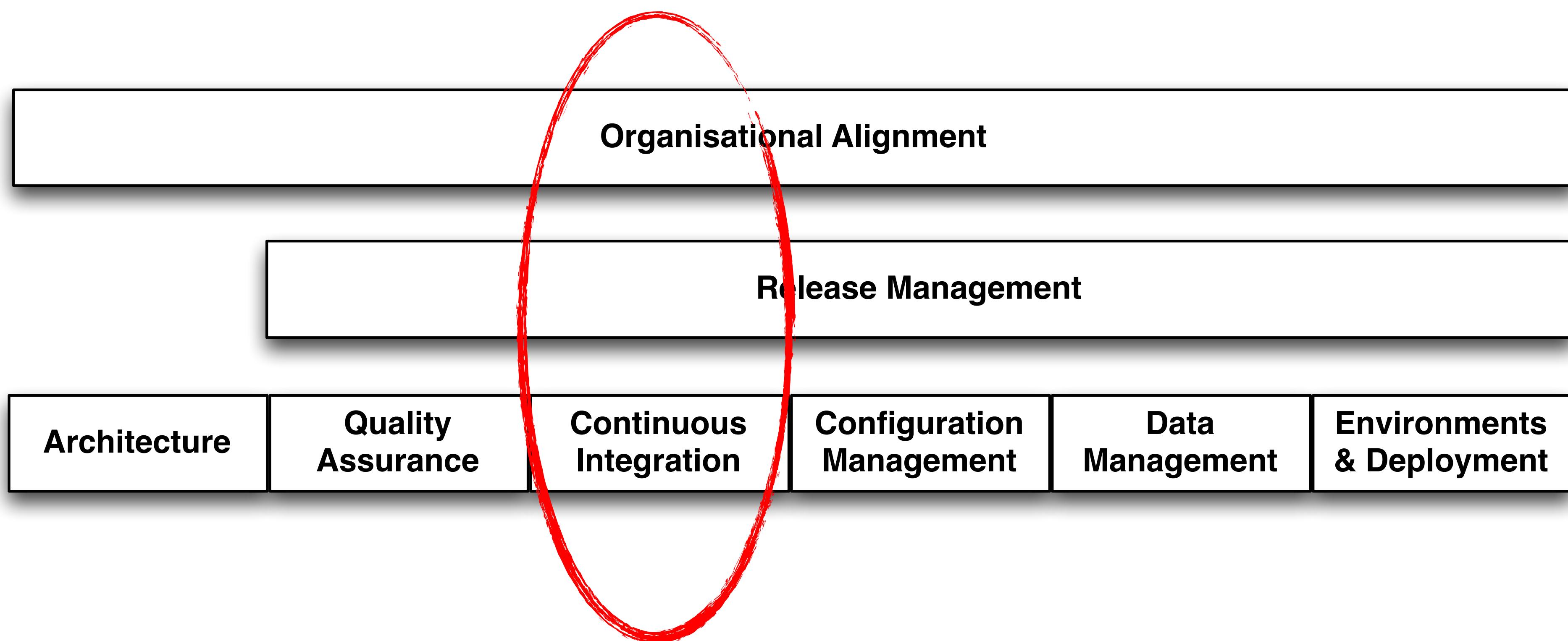
test at the lowest level

merge teams
when possible

agree on goals
and metrics



Continuous Delivery

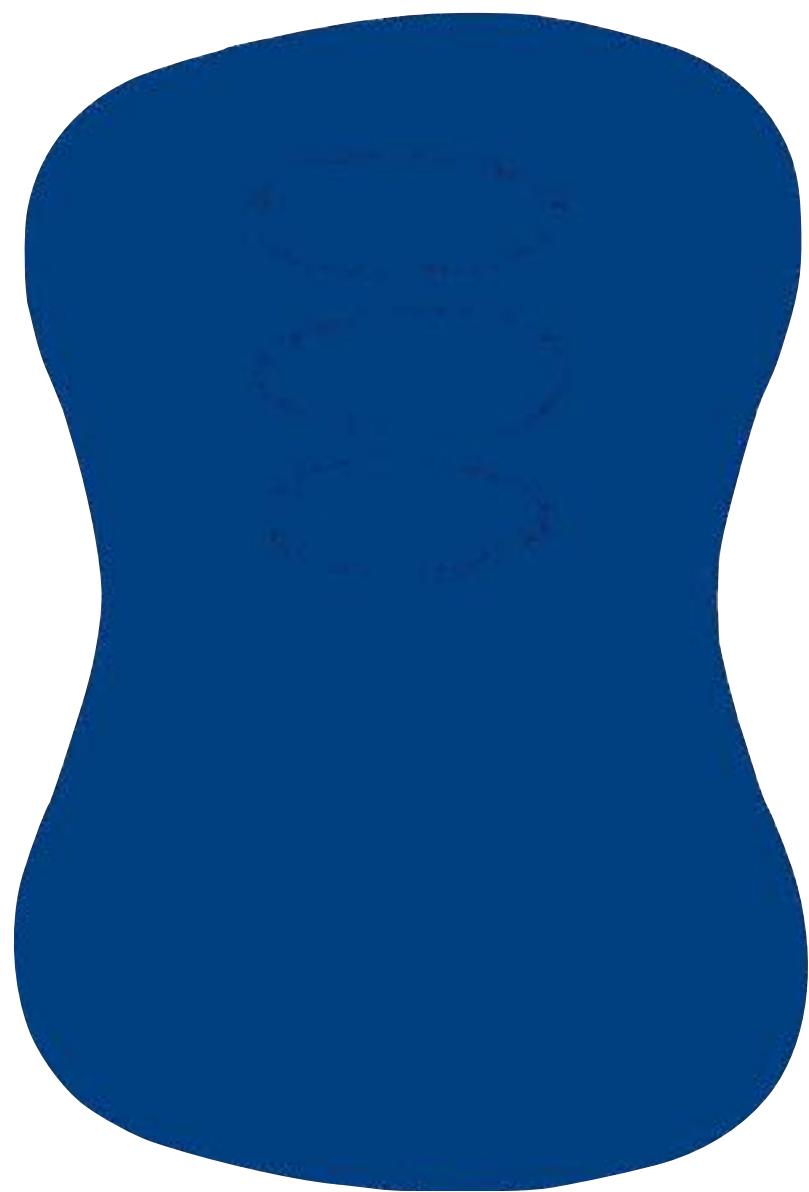




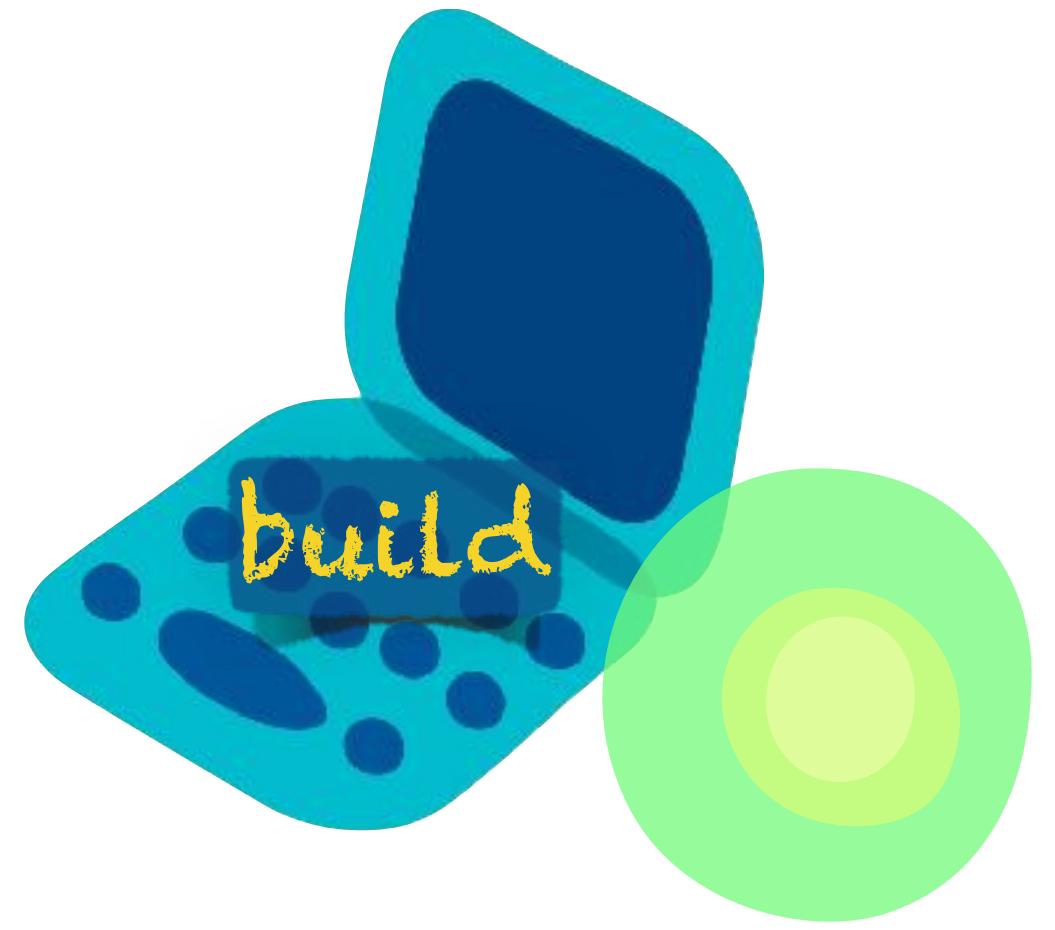
local
workstation



version
control

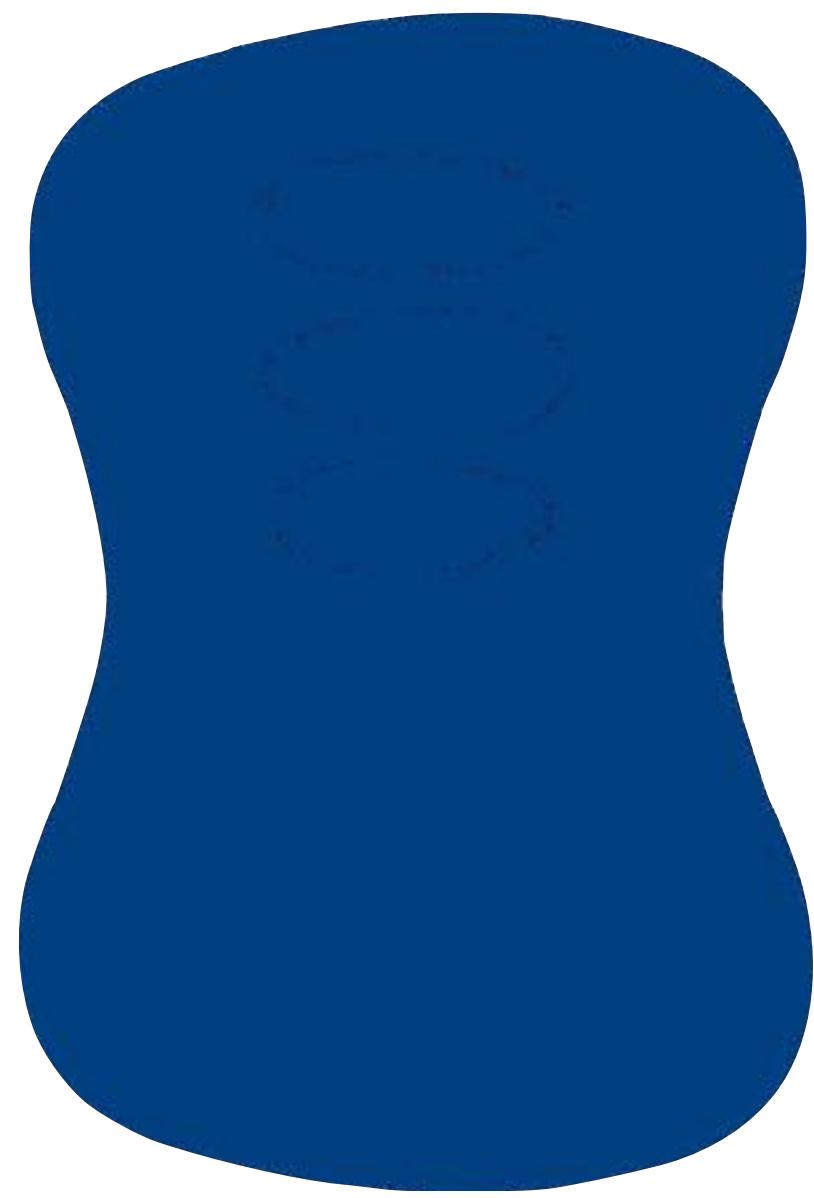


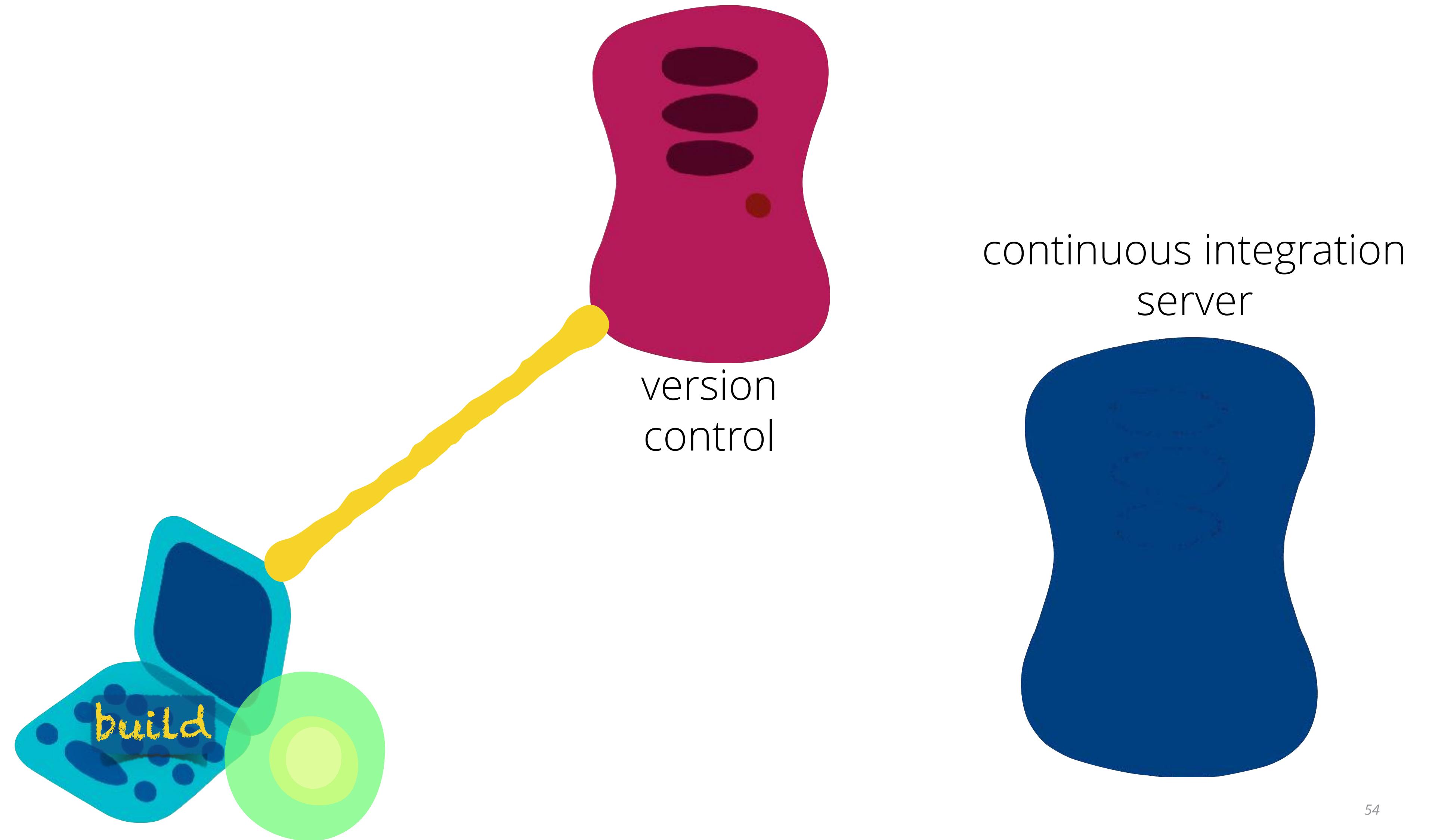
continuous integration
server

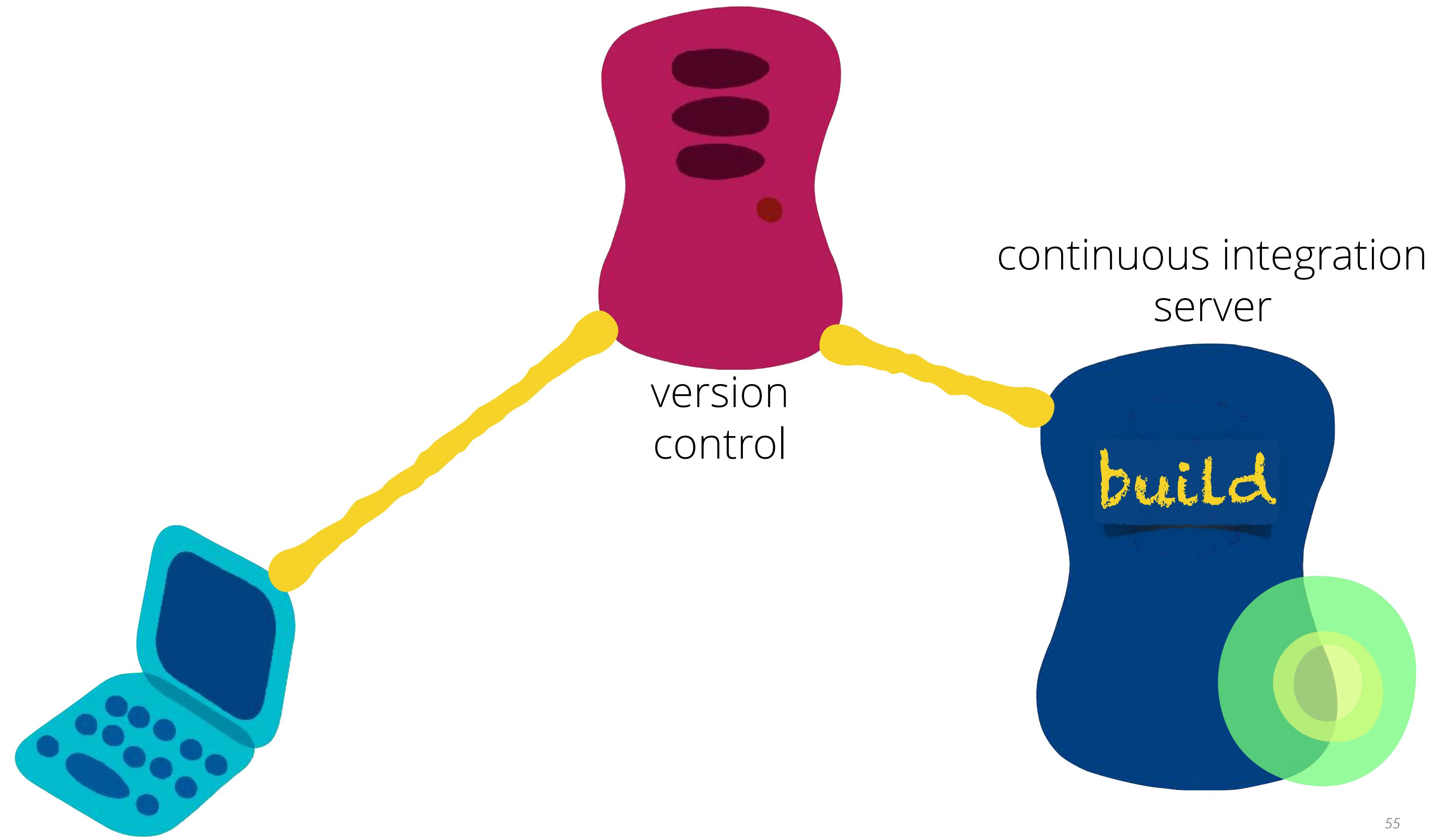


version
control

continuous integration
server







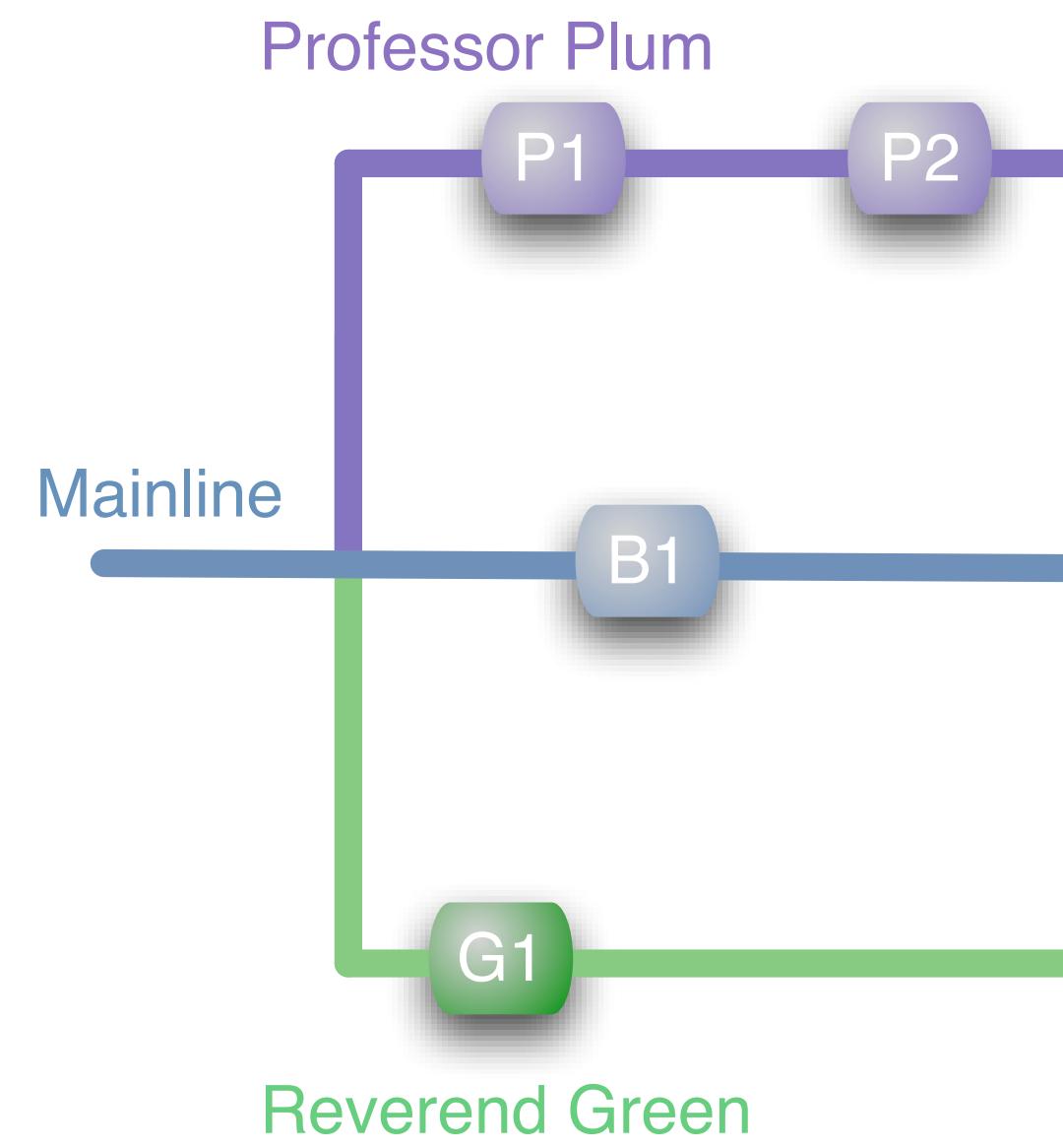
everyone commits

to trunk at least

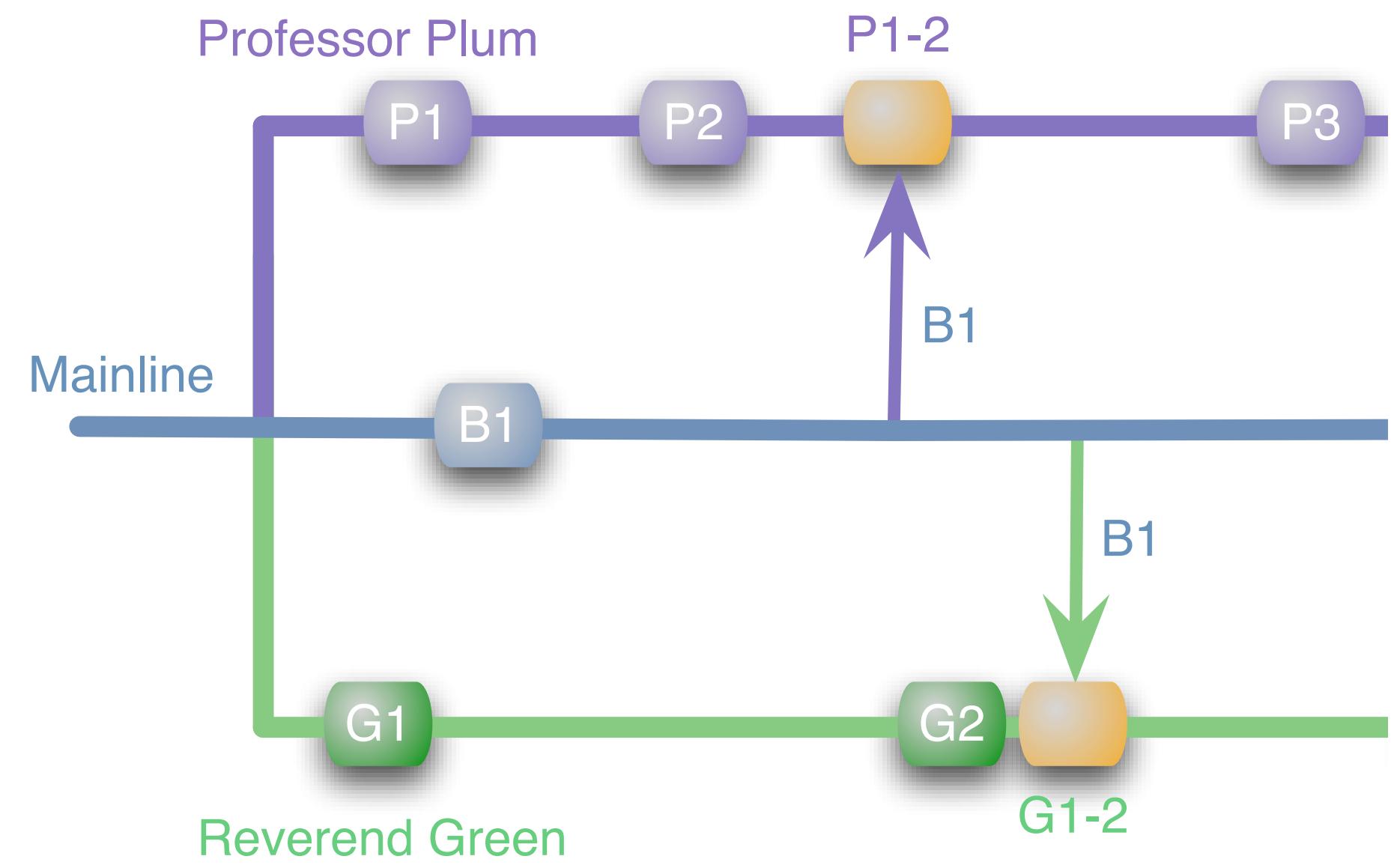
once a day

version control

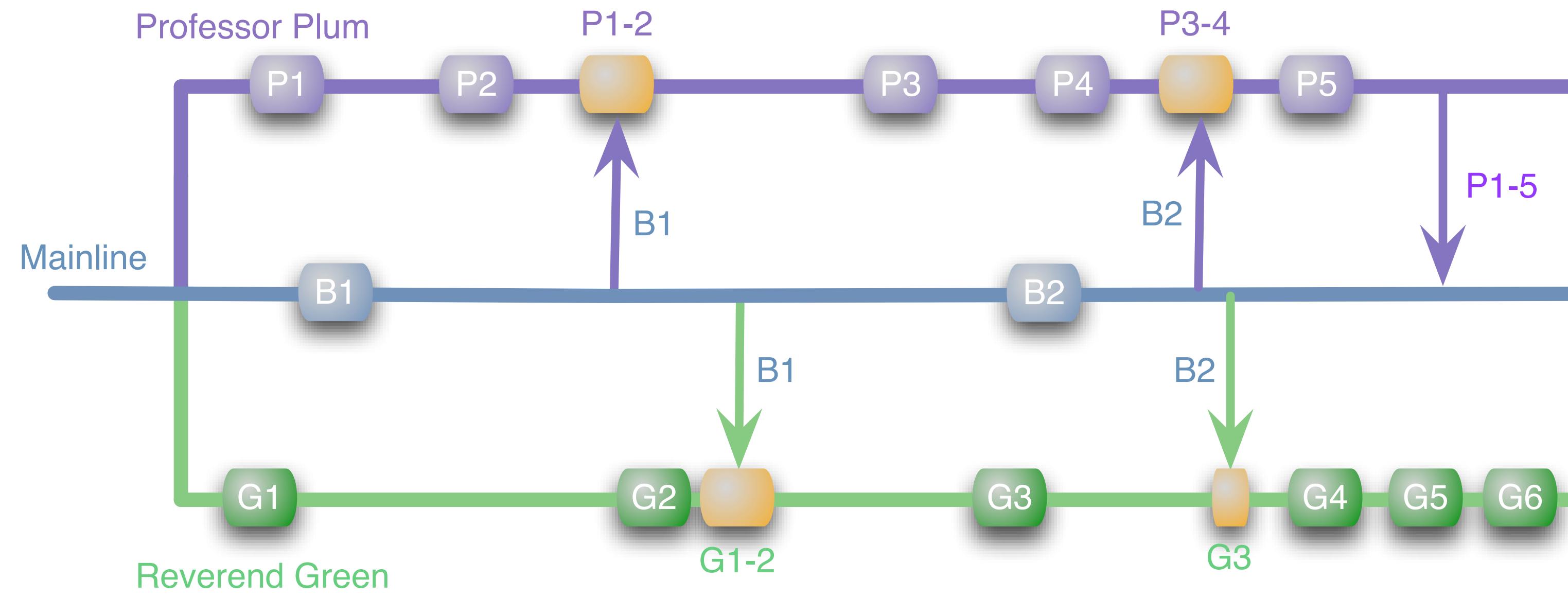
build



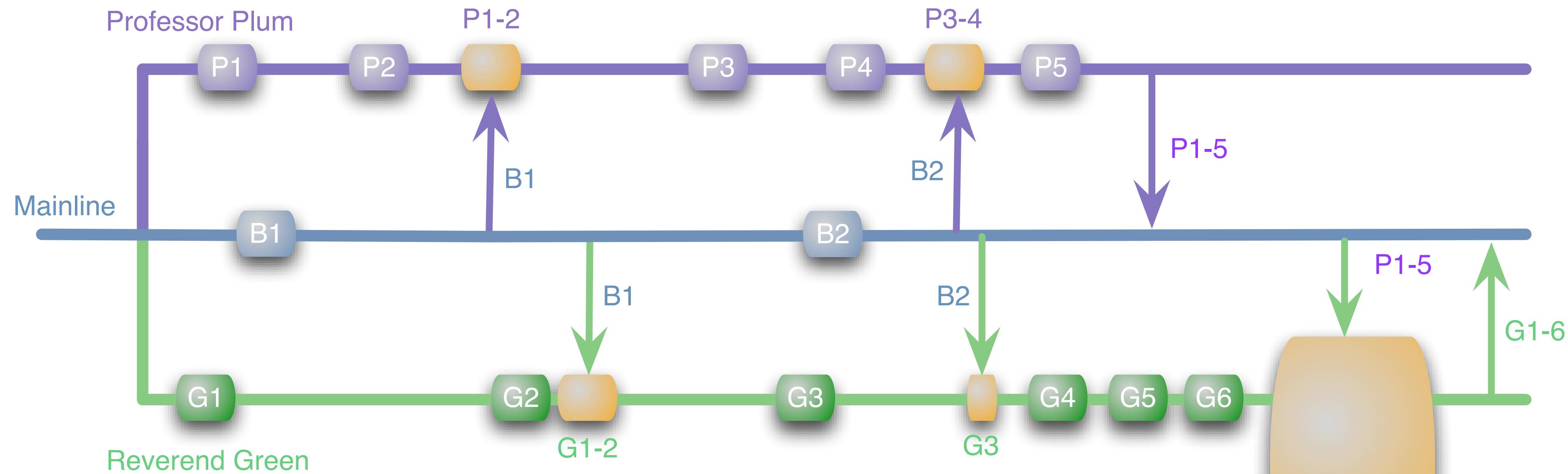
Feature Branching



Feature Branching

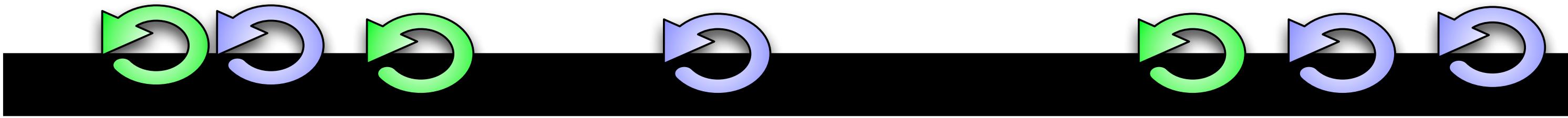


Feature Branching

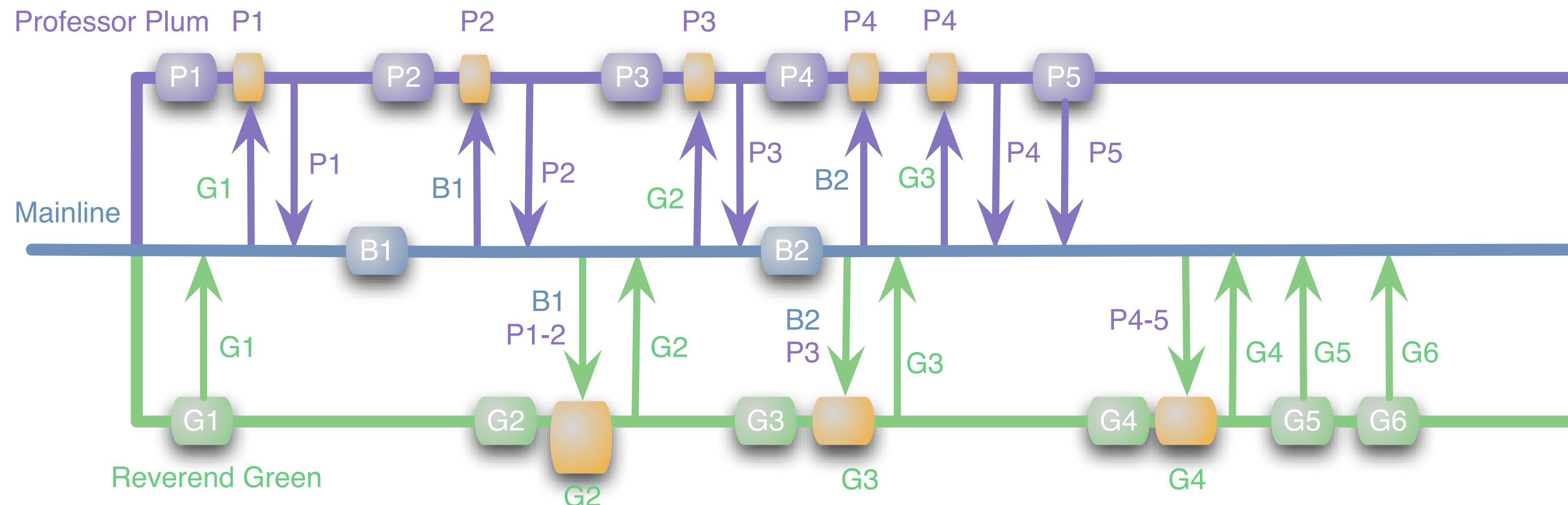


merge
ambush!

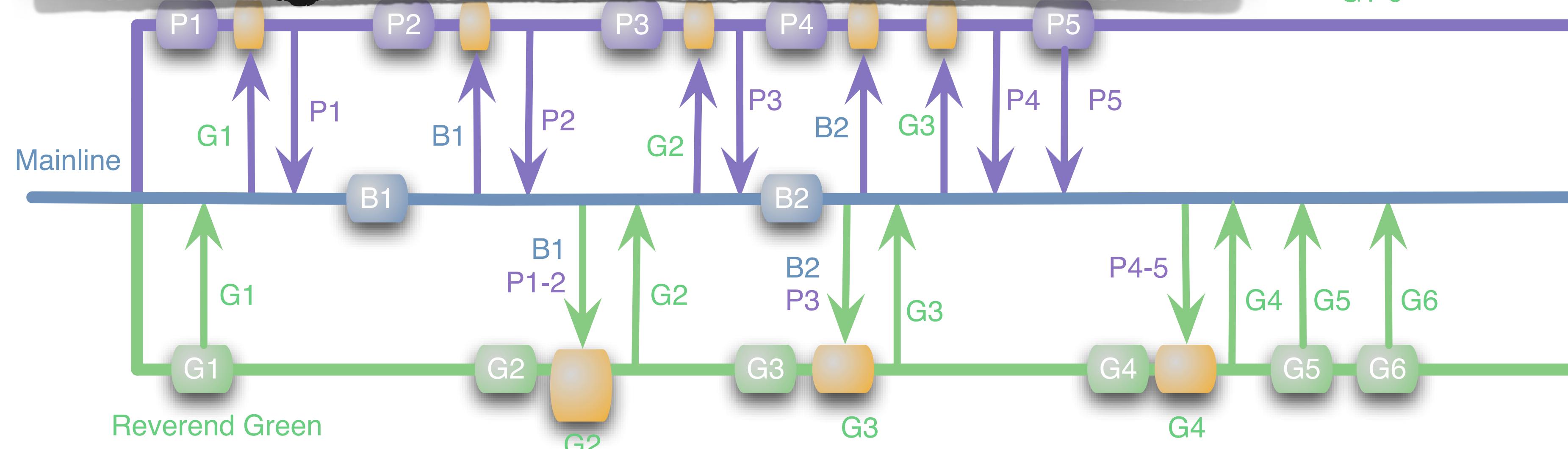
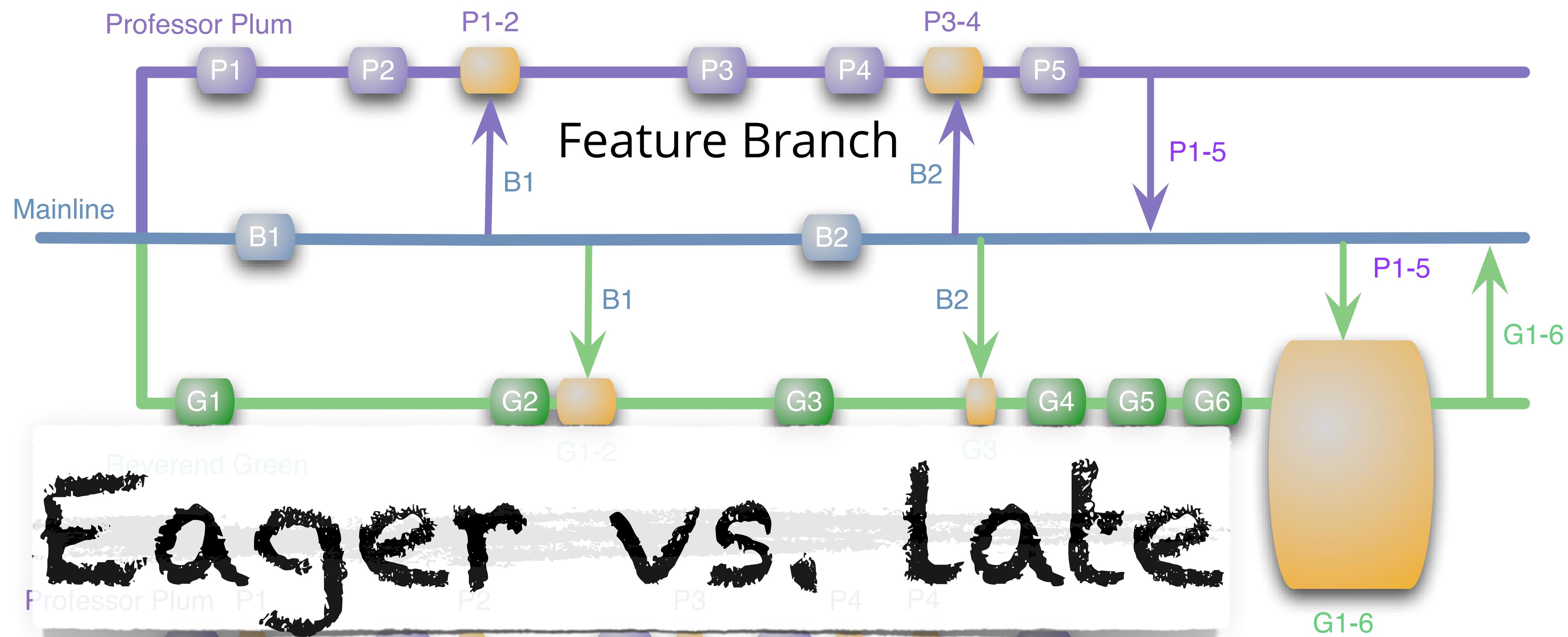
Feature Branching



trunk-based development



Continuous Integration removes the pain...



Config File

```
[featureToggles]
wobblyFoobars: true
flightyForkHandles: false
```

feature toggles

some.jsp

```
<toggle name=wobblyFoobars>
    . various UI elements
</toggle>
```

other.java

```
forkHandle = (featureConfig.isOn('flightyForkHandles)) ?
    new FlightyForkHandler(aCandle) :
    new ForkHandler(aCandle)
```

Togglz - Features flag for Java

www.togglz.org

Reader



Feature Flags for the Java platform

MAIN

- [Home](#)
- [Downloads](#)
- [Source Code](#)
- [Forums](#)
- [Issue Tracker](#)
- [stackoverflow.com](#)
- [Continuous Integration](#)
- [License](#)

REFERENCE

- [What's new?](#)
- [Getting Started](#)
- [Javadocs 2.0.0.Final](#)
- [Javadocs 1.1.0.Final](#)
- [Javadocs 1.0.0.Final](#)
- [Updating Notes](#)

DOCUMENTATION

Togglz

What is it about?

Togglz is an implementation of the [Feature Toggles](#) pattern for Java. Feature Toggles are a very common agile development practices in the context of continuous deployment and delivery. The basic idea is to associate a toggle with each new feature you are working on. This allows you to enable or disable these features at application runtime, even for individual users.

Want to learn more? Have a look at an [usage example](#) or check the [quickstart guide](#).

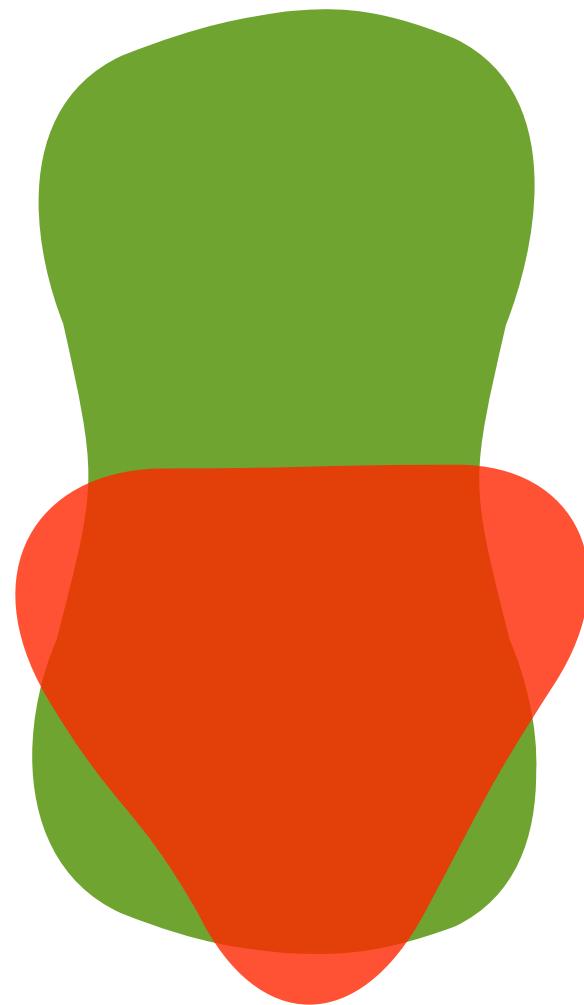
News

01-Jul-2013

Togglz 2.0.0.Final released

I'm very happy to announce the release of Togglz 2.0.0.Final. This new version is the result of many months of hard work. Many core concepts of Togglz have been revised to provide much more flexibility.

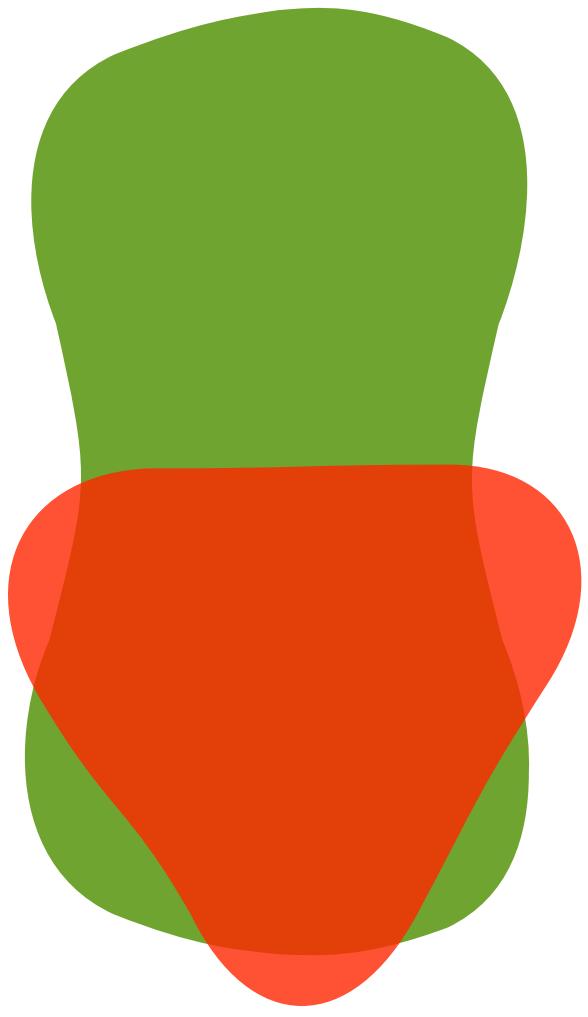
The most noteworthy change in Togglz 2.0.0.Final is the new extendible feature activation mechanism that allows to implement custom strategies for activating features. Beside that there are many other updates.



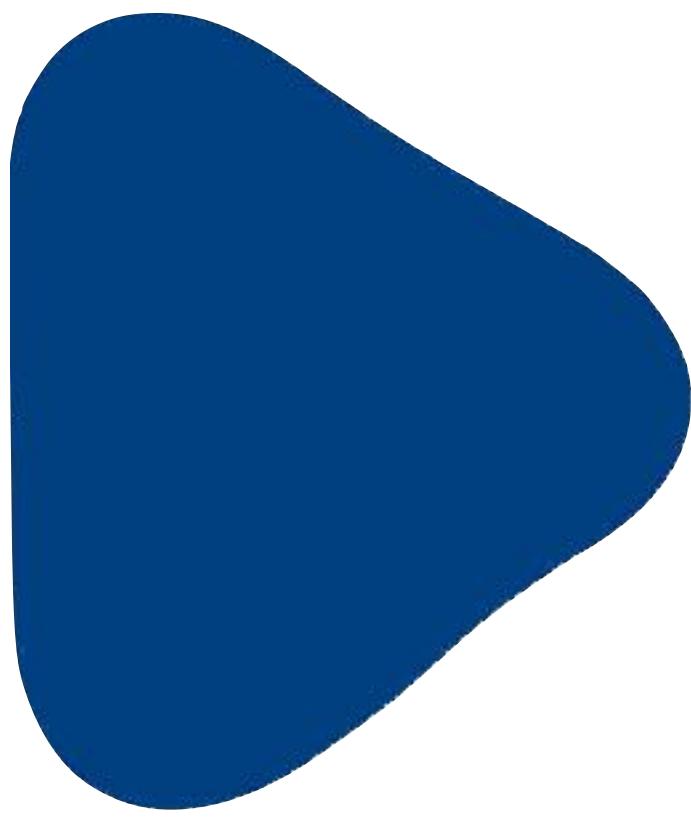
removed as soon as feature
decision is resolved

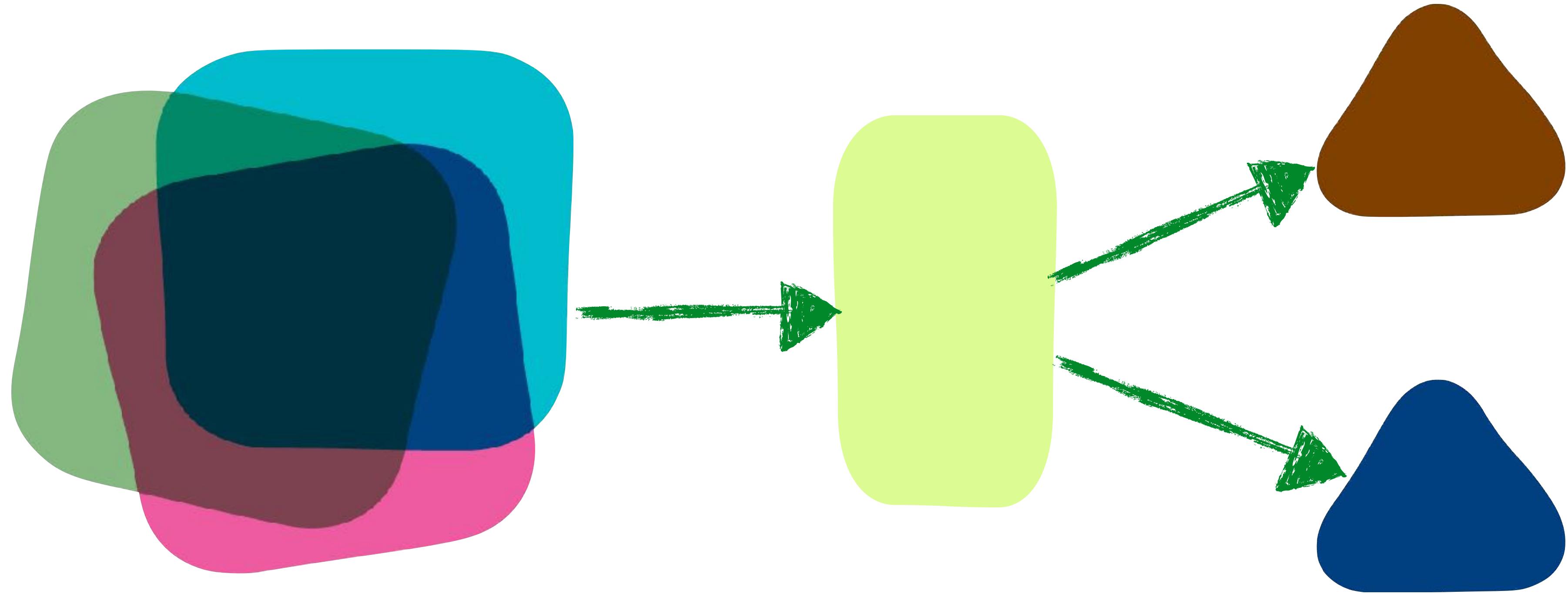
Feature toggles are purposeful
technical debt added to support
engineering practices like
Continuous Delivery.



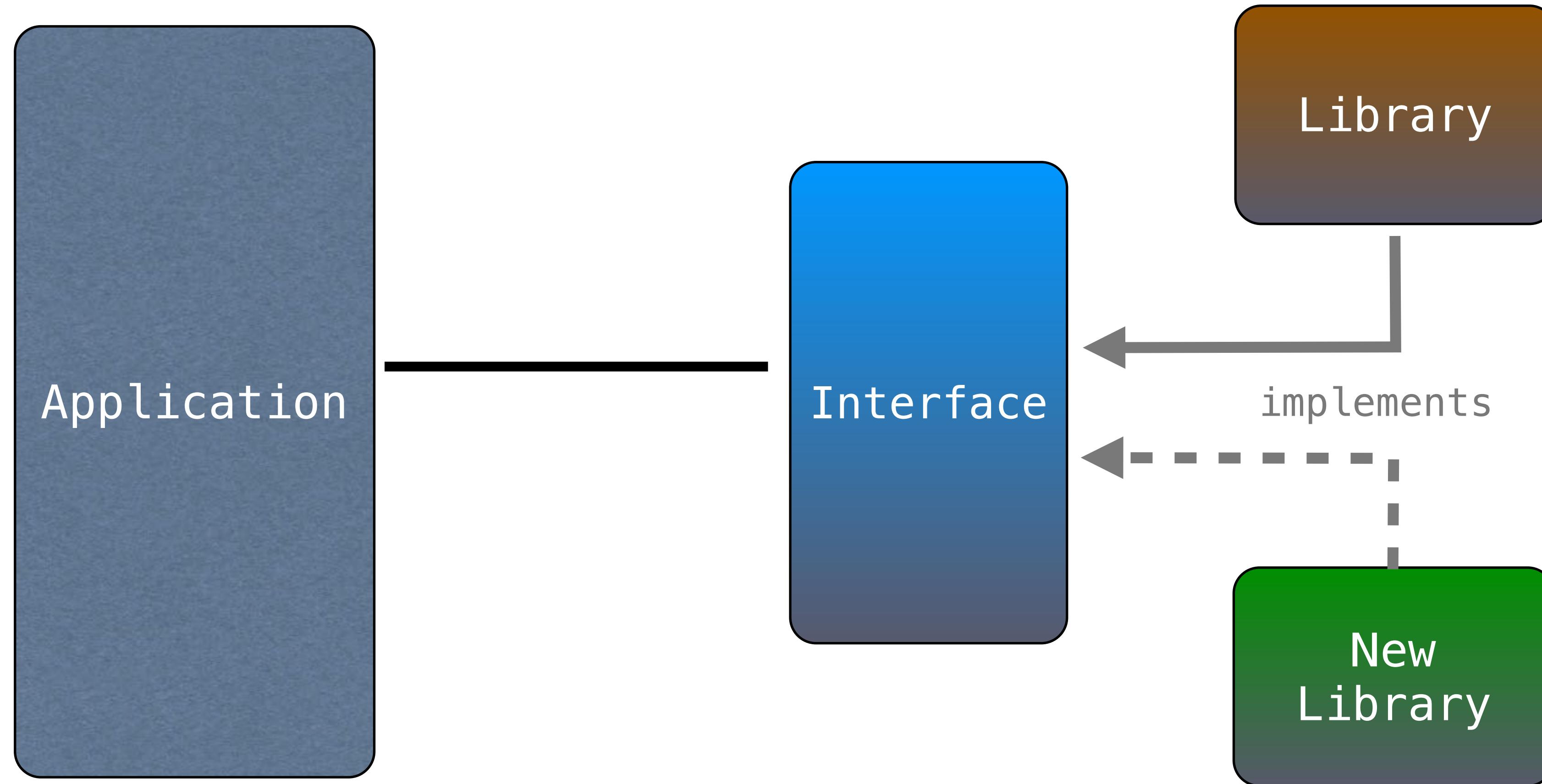


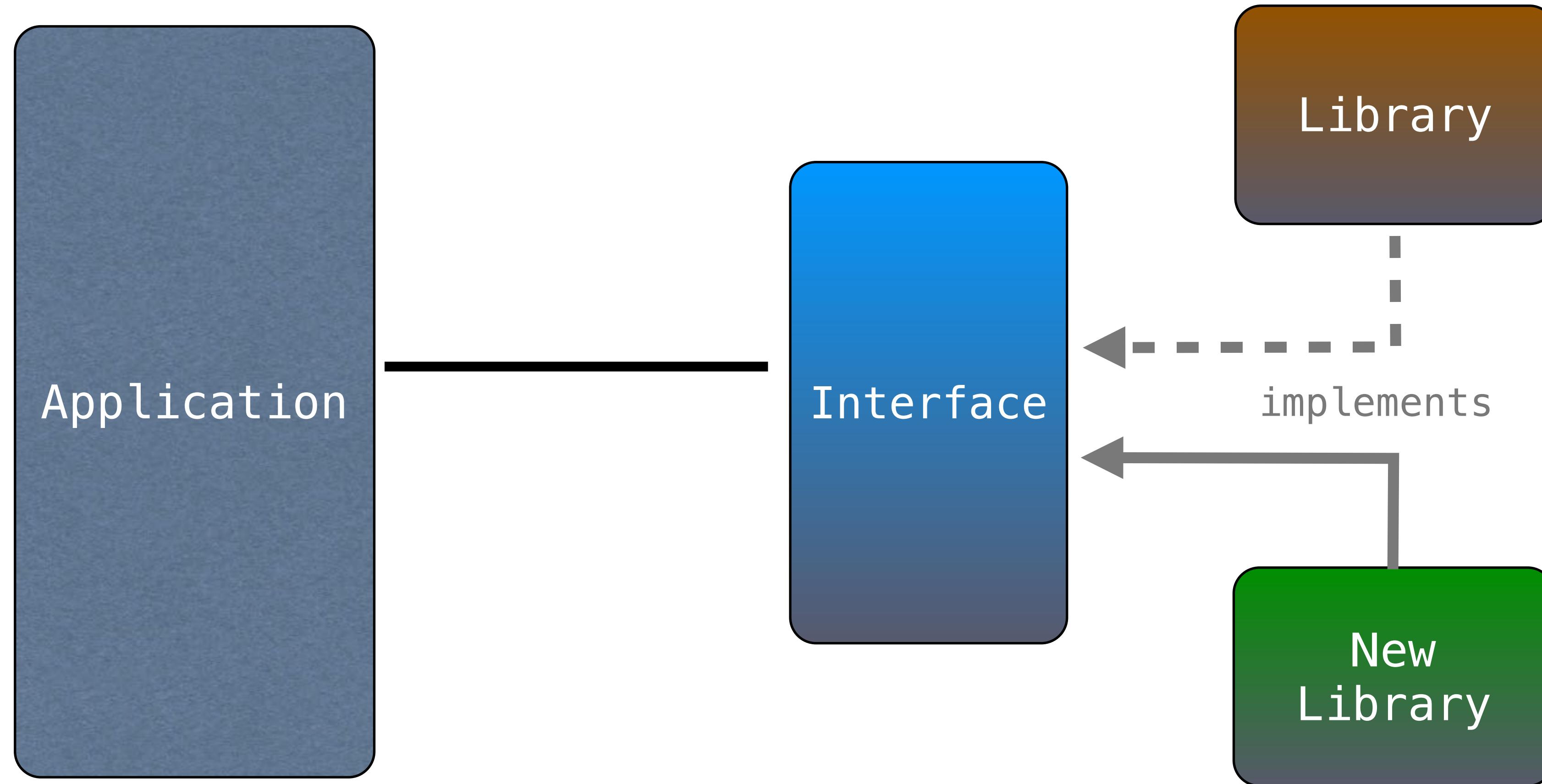
build-time vs. run-time

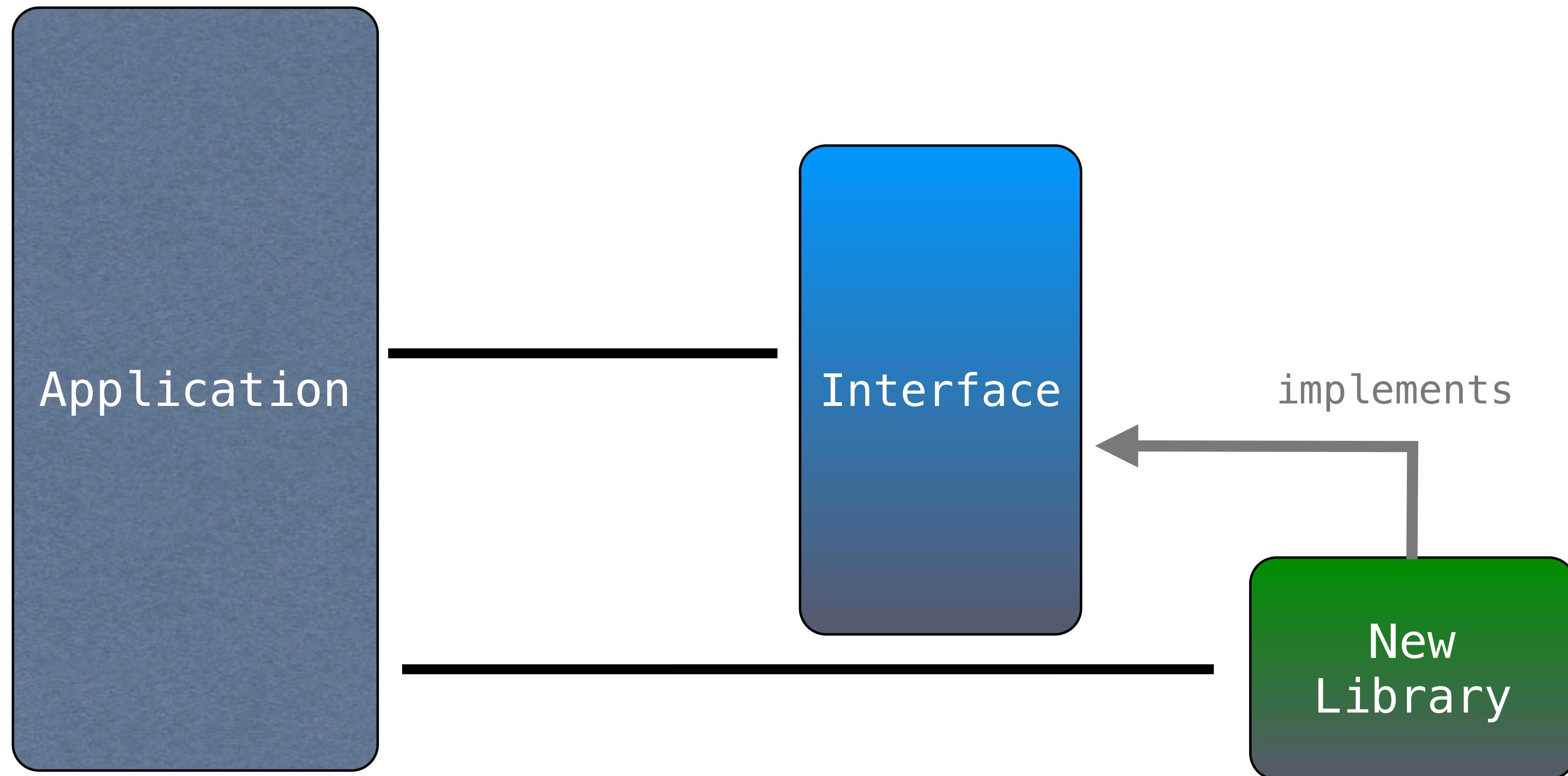




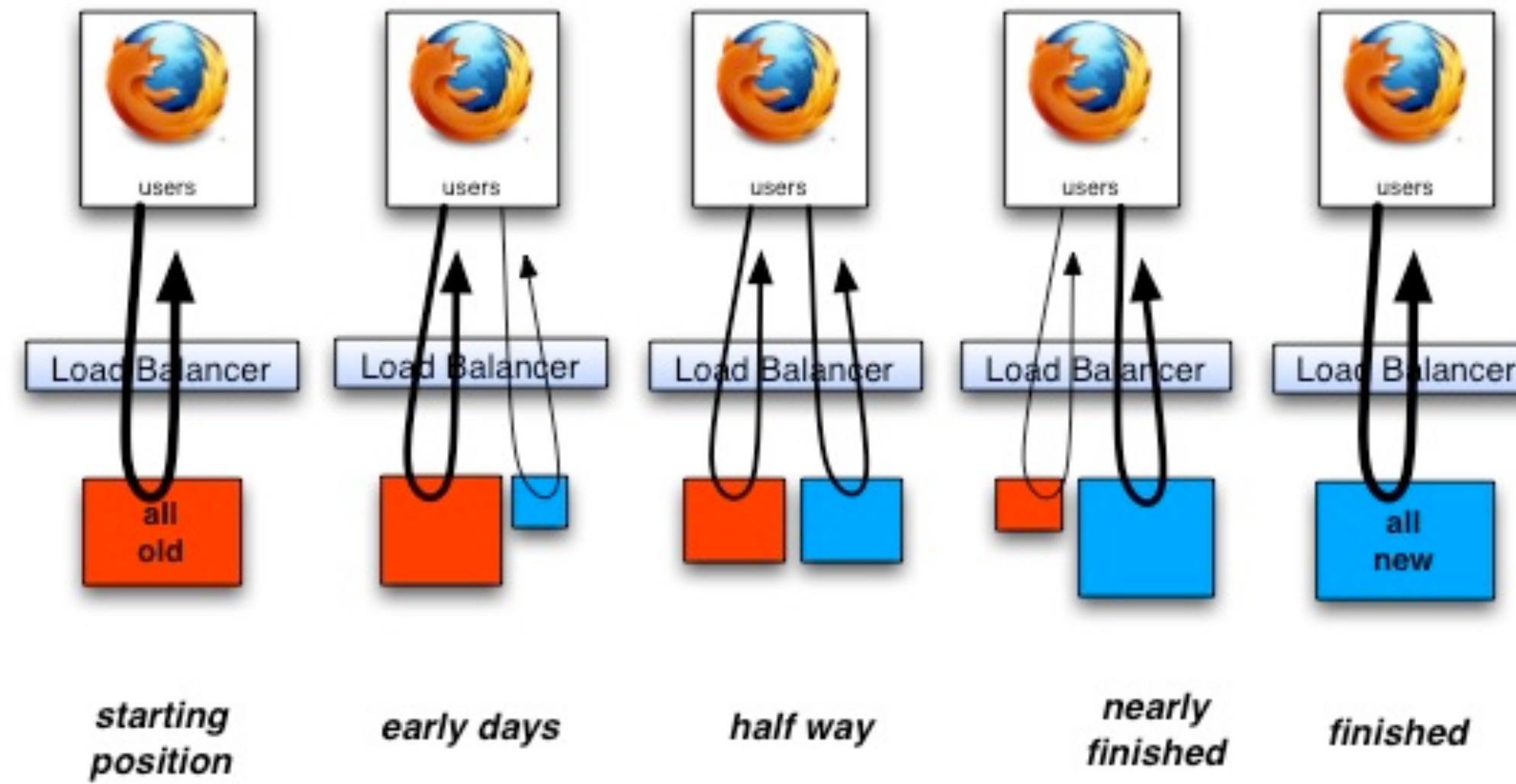
Branch by
Abstraction







“Strangler” Pattern

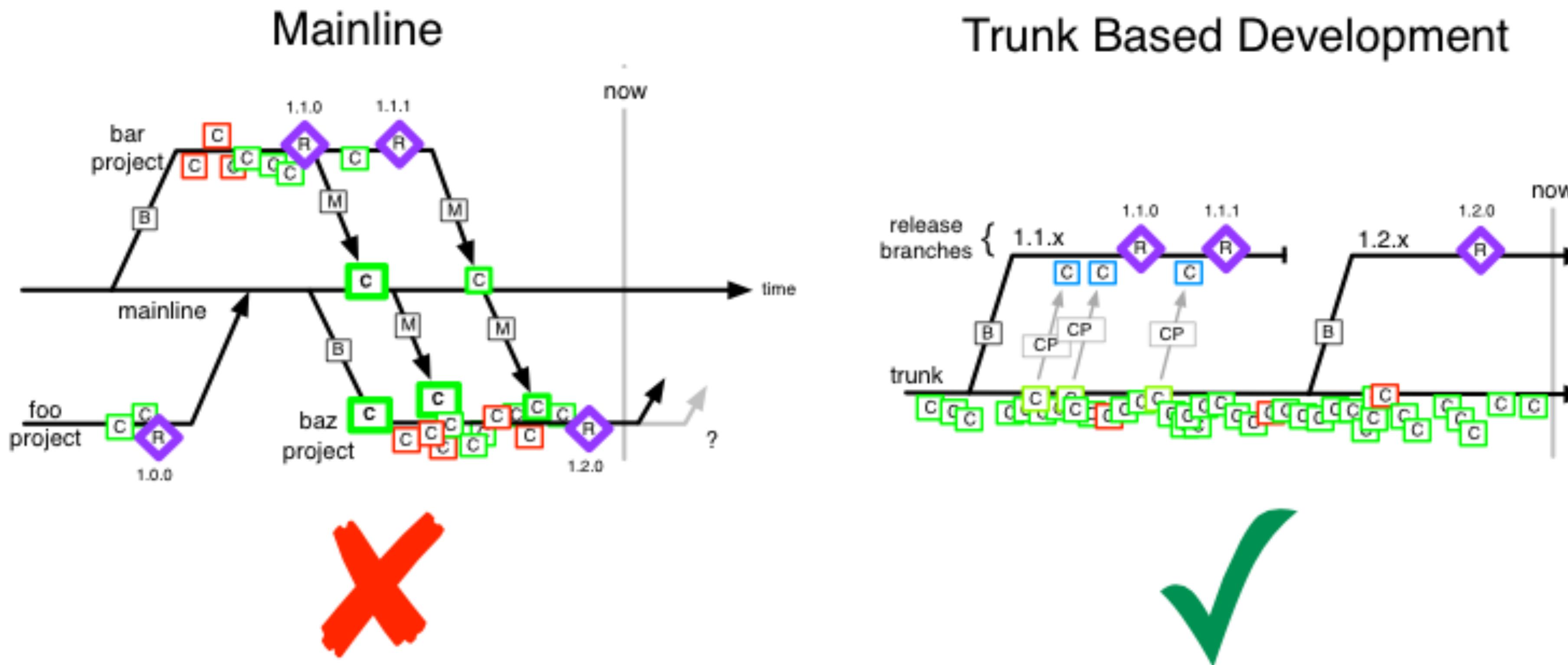


make something new that obsoletes a
small percentage of something old

put them live together

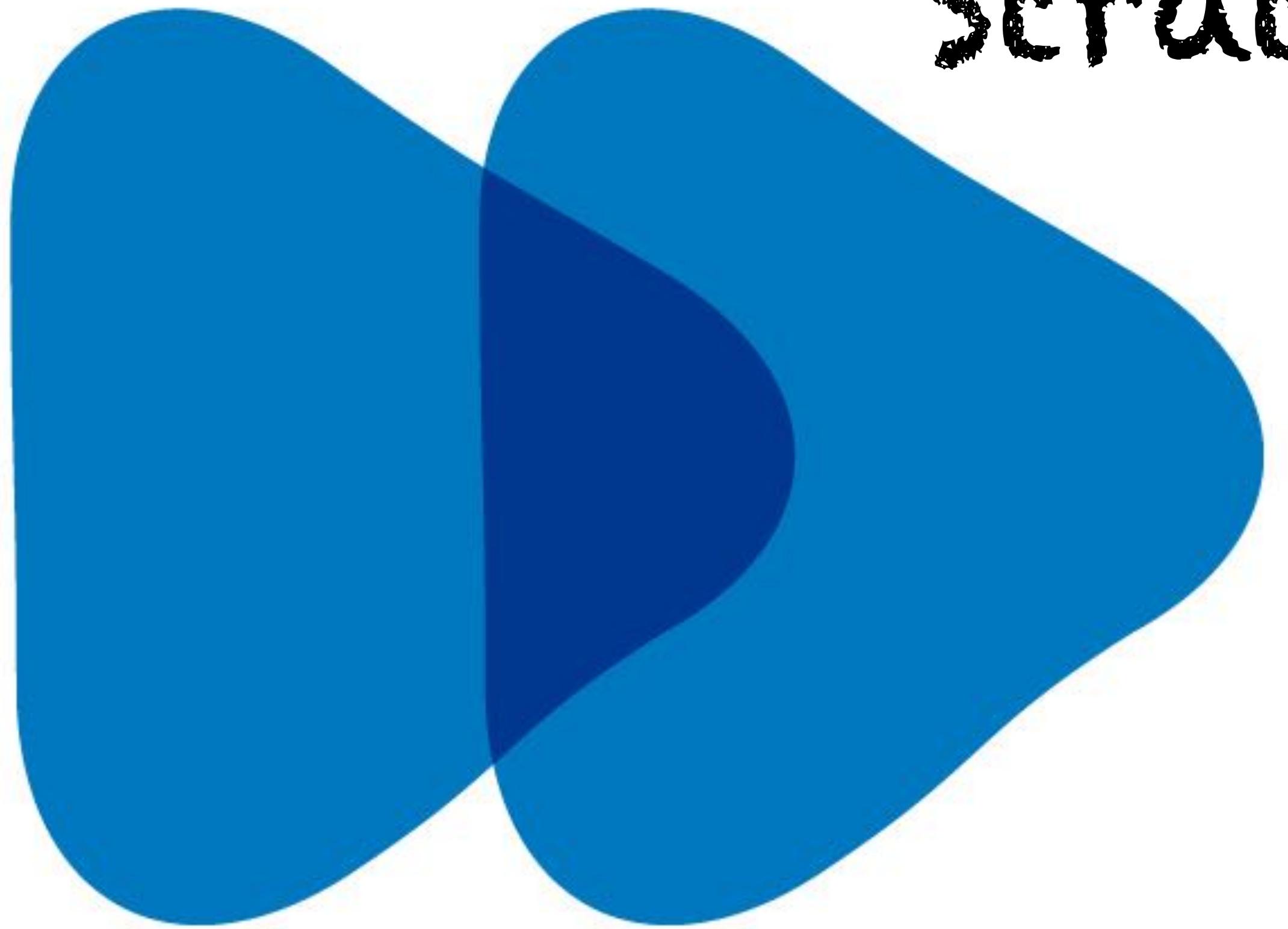
rinse, repeat

Release branches are OK...



Long-lived branches damage
continuous integration.

Release Strategies

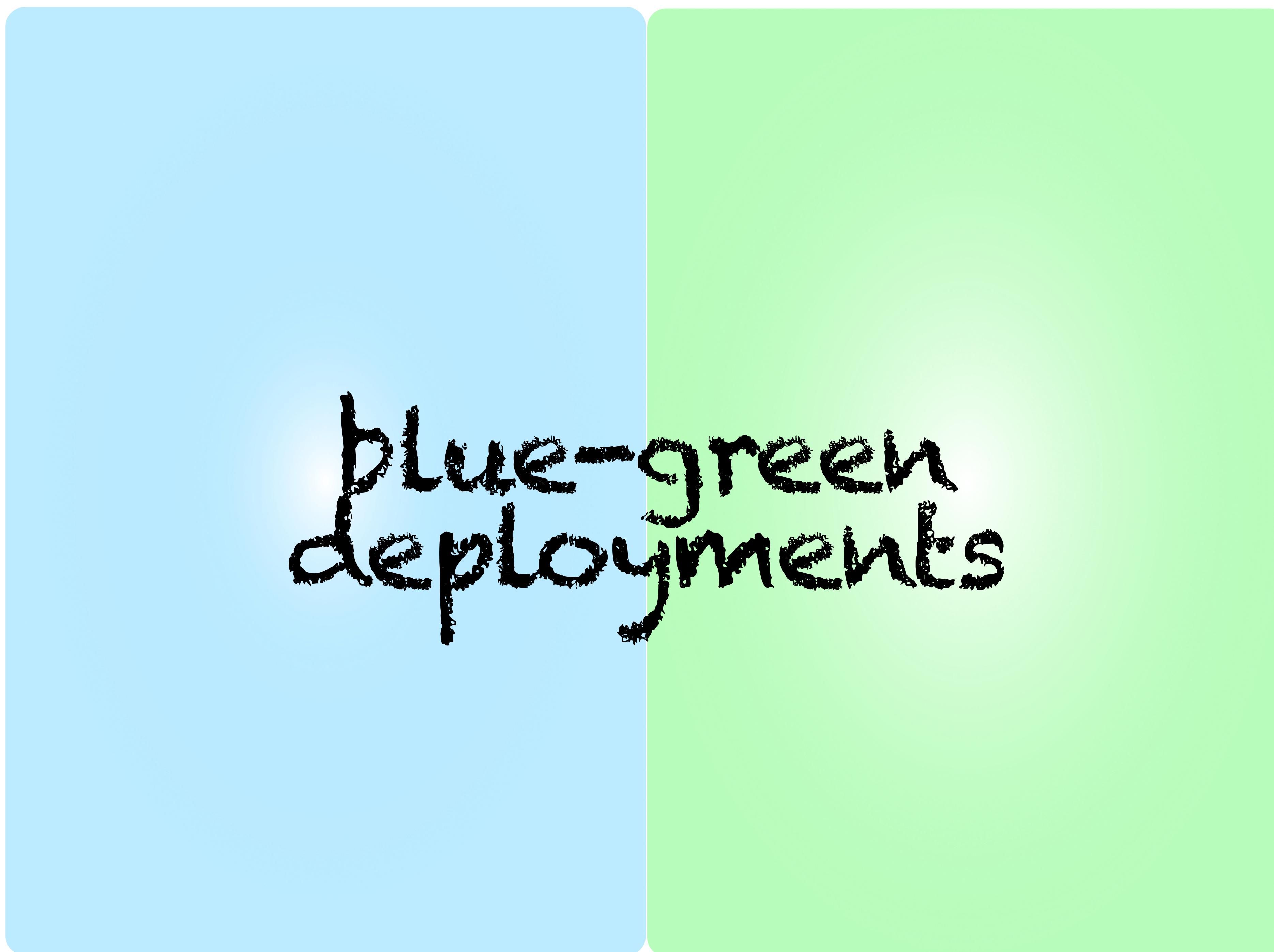


blue-green deployments

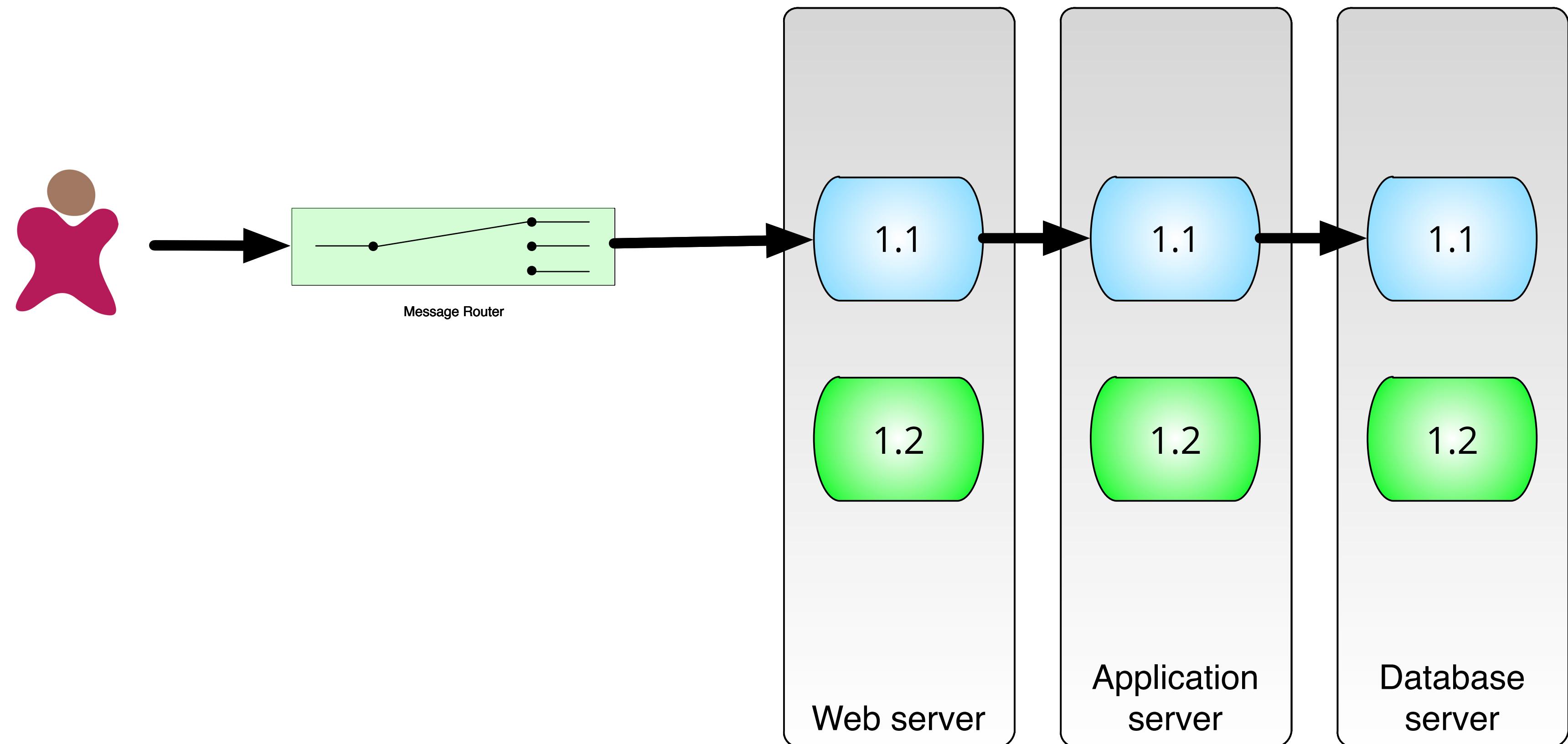
canary releases

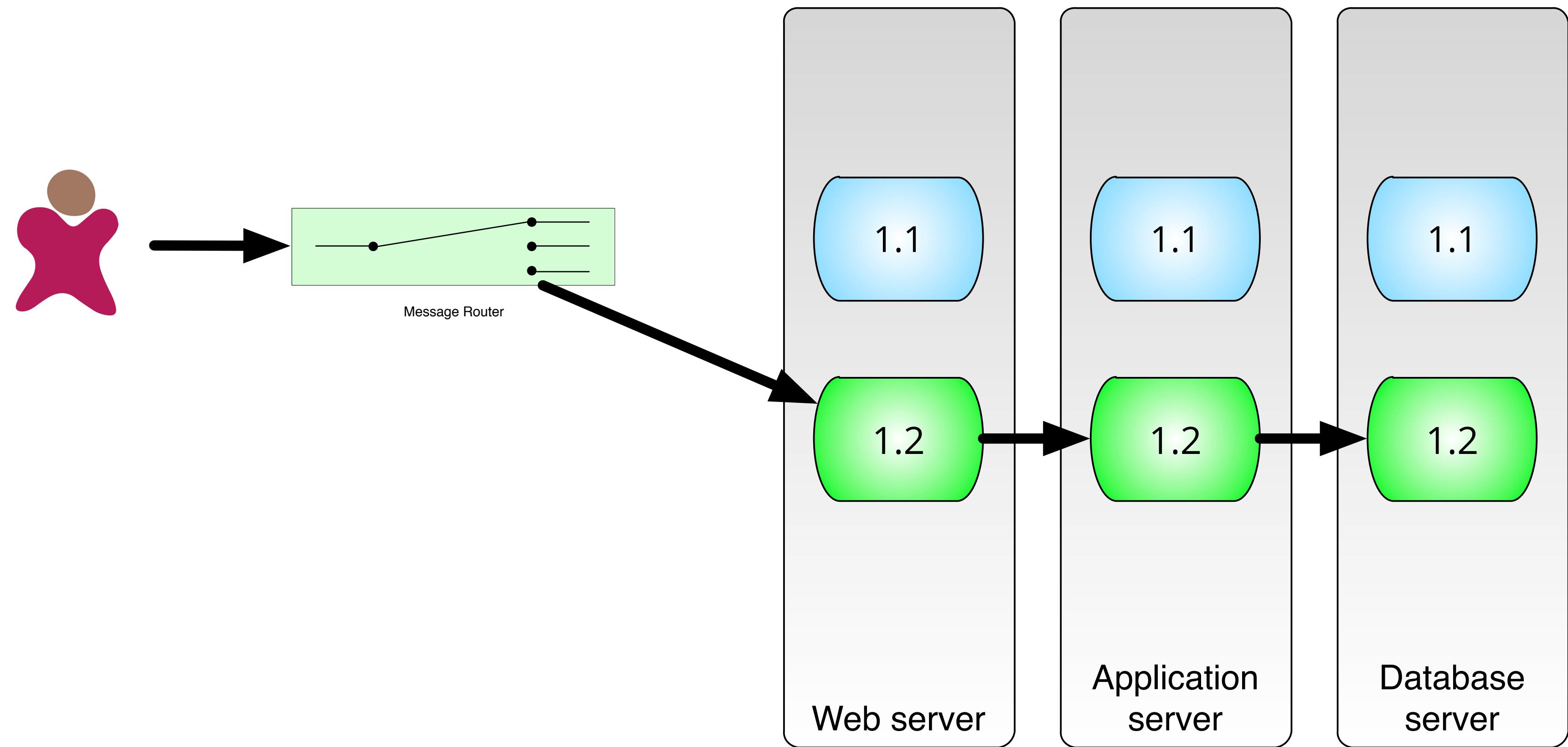
Incremental Release Strategies

dark launching

The background of the slide is divided into two vertical sections: a light blue section on the left and a light green section on the right. In the center, there is a graphic consisting of two overlapping circles. The top circle is blue and the bottom circle is green, representing the 'blue' and 'green' environments used in a blue-green deployment strategy.

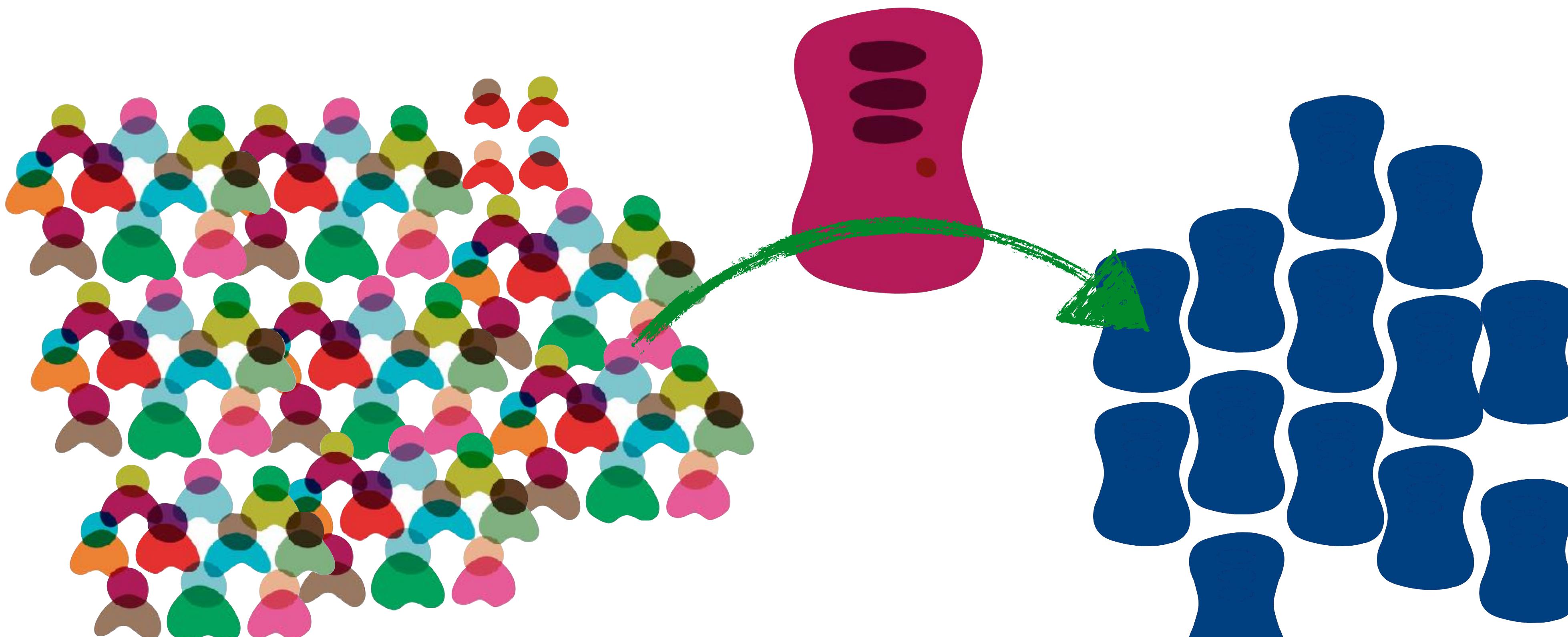
blue-green
deployments



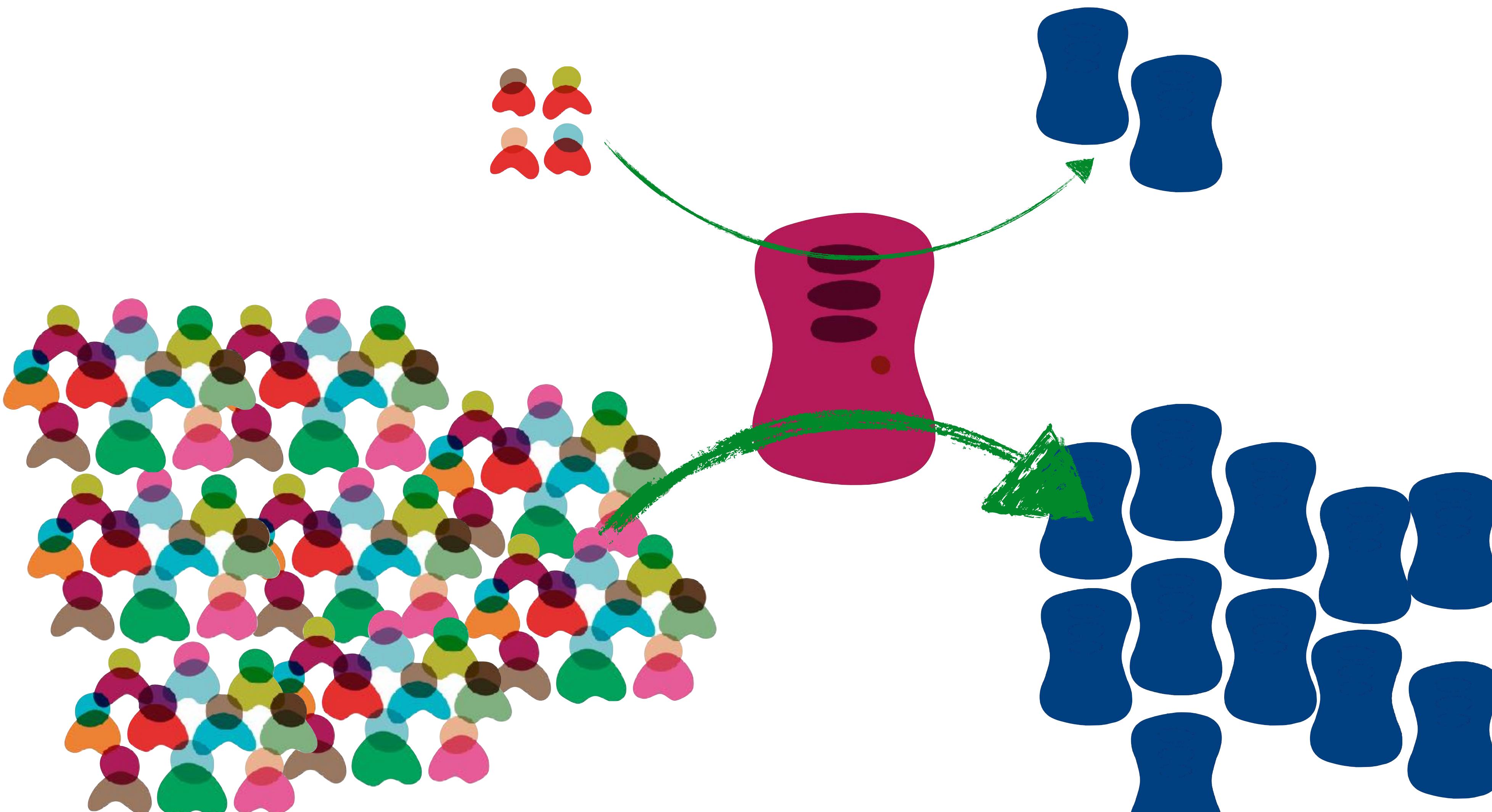


Canary Releasing

Canary Releasing

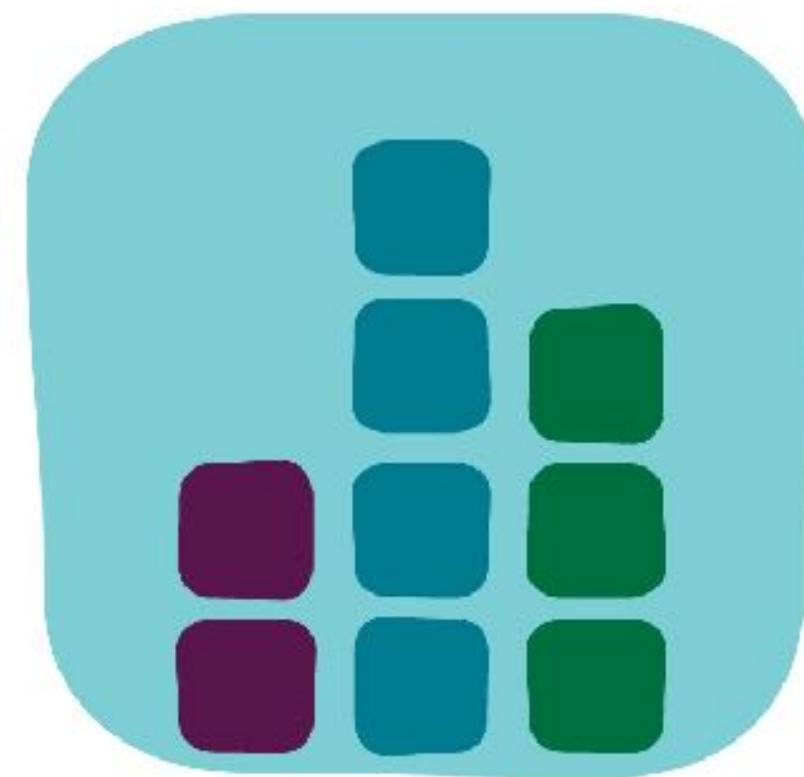


Canary Releasing

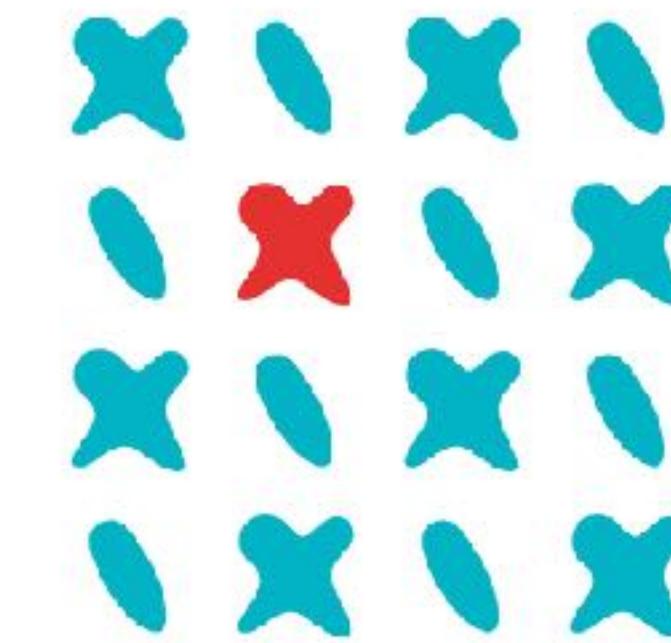


Canary Releasing

reduce risk of release

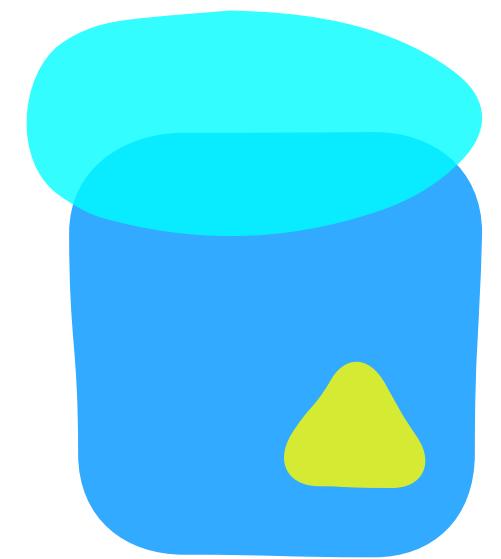
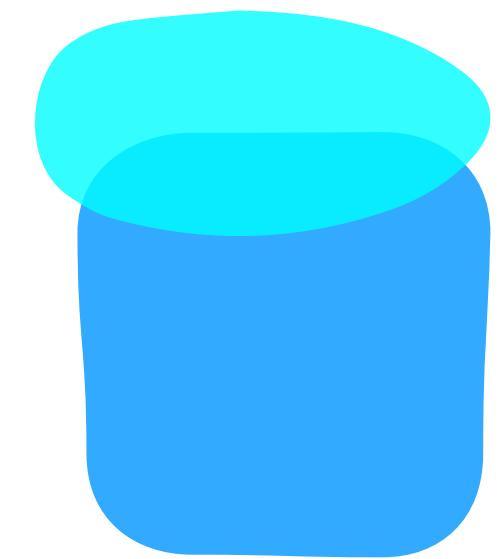
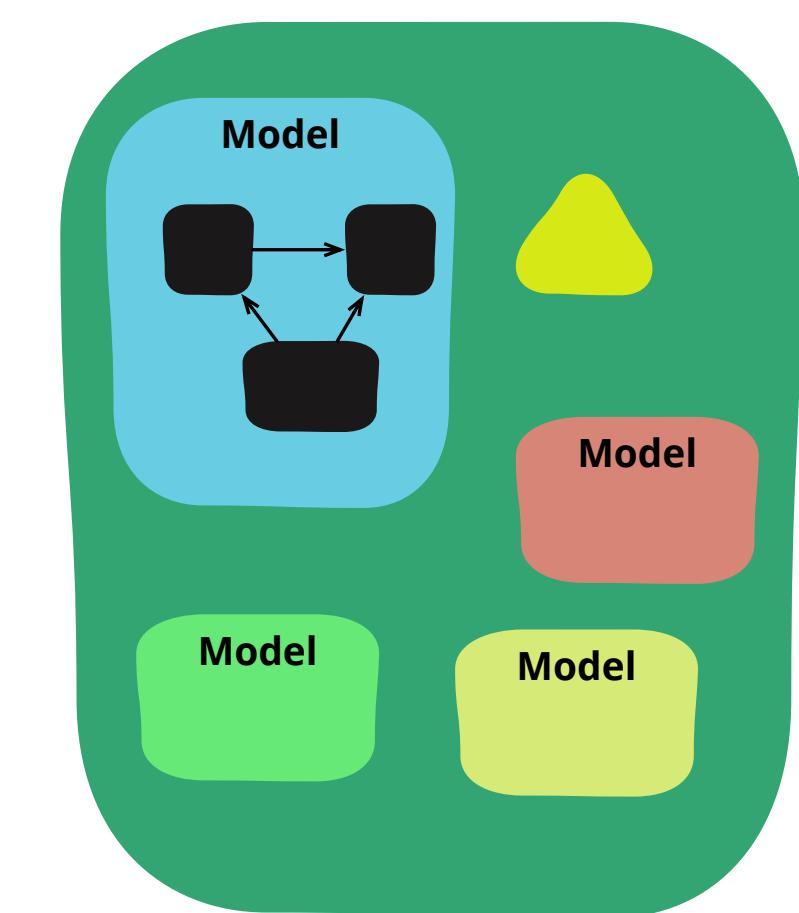
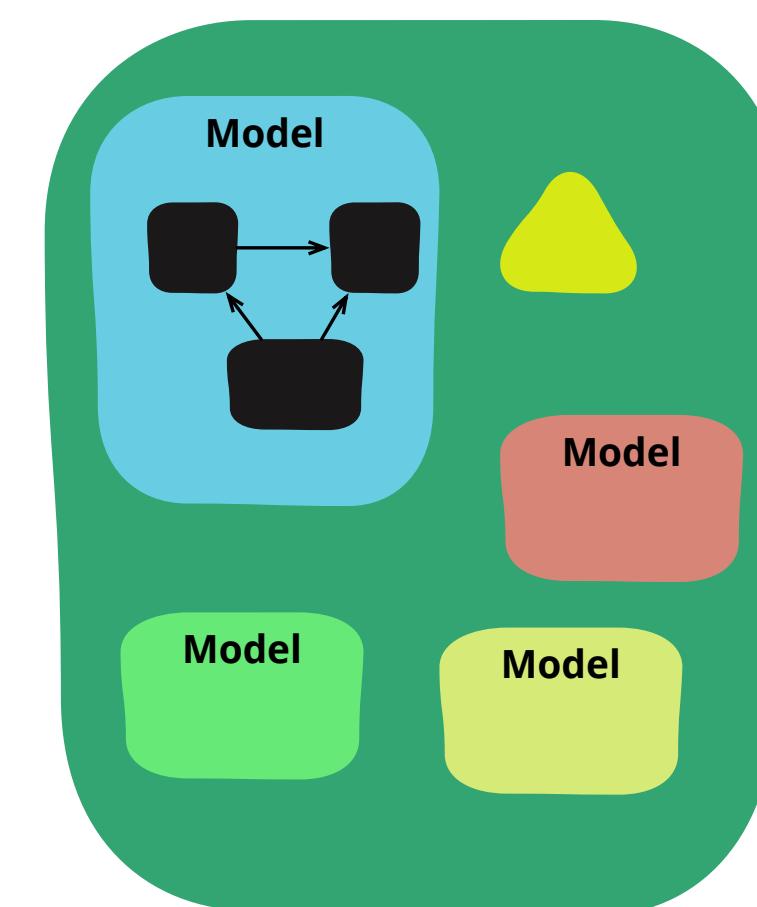
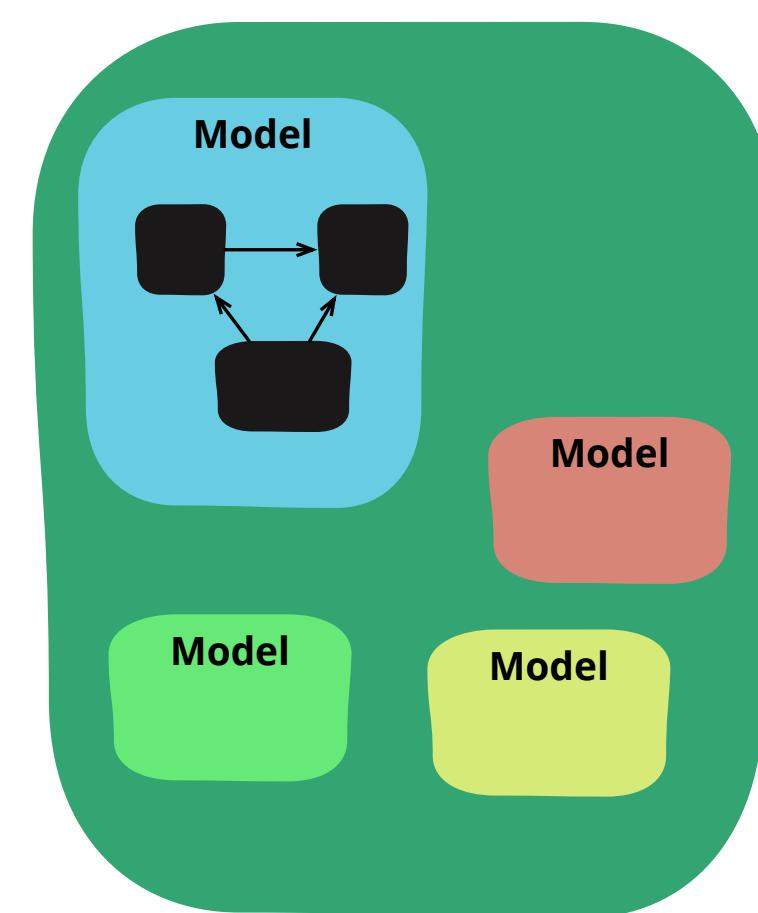
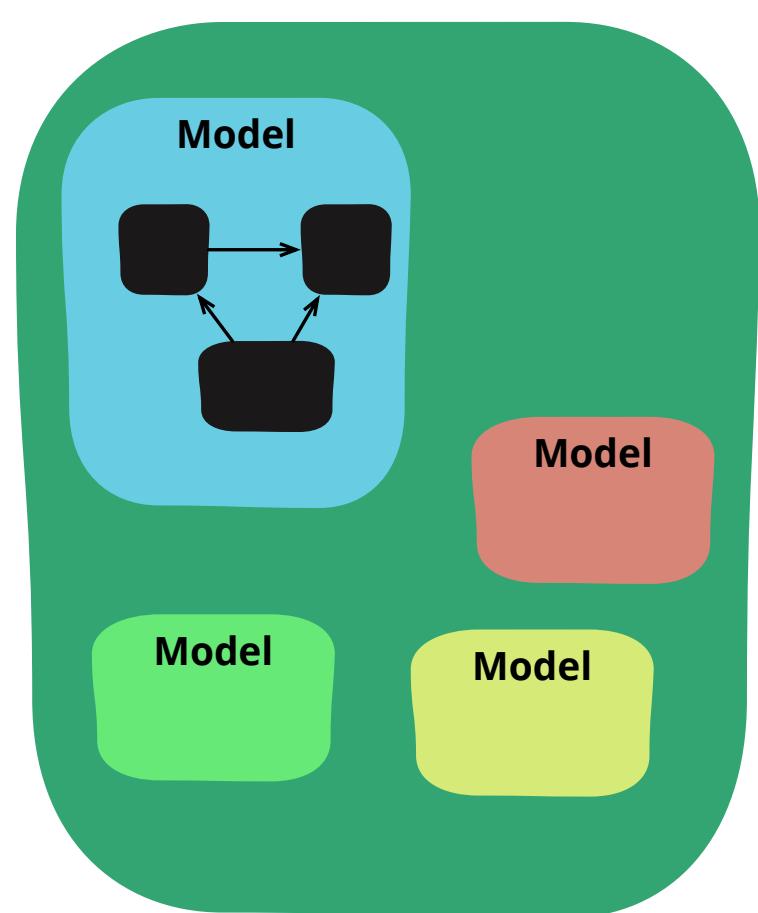
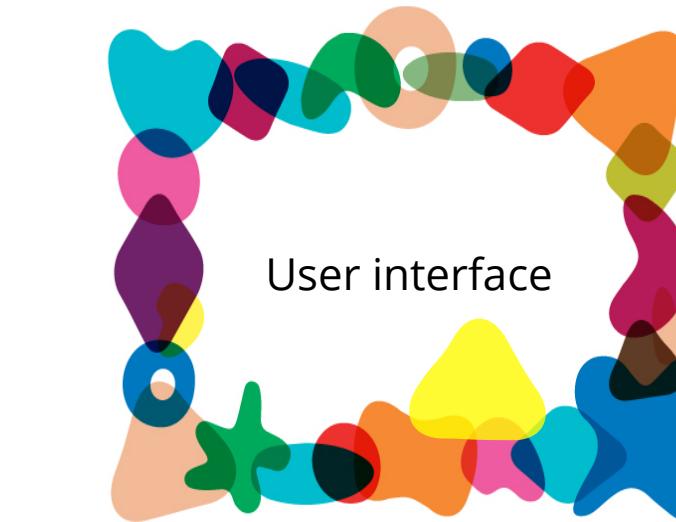
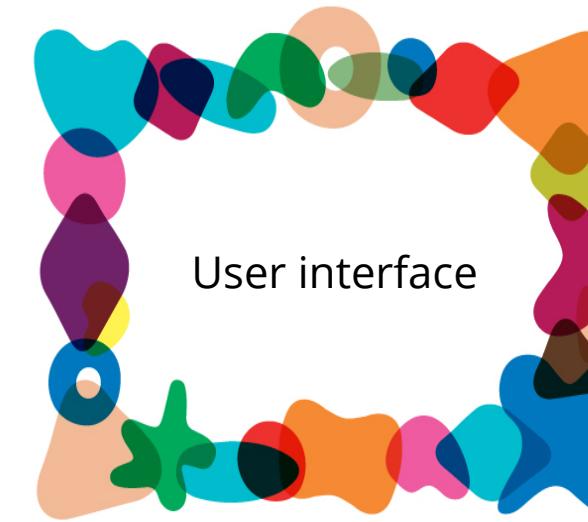
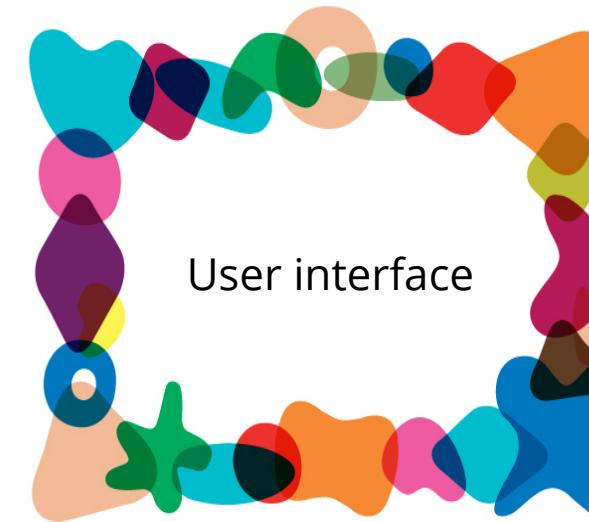
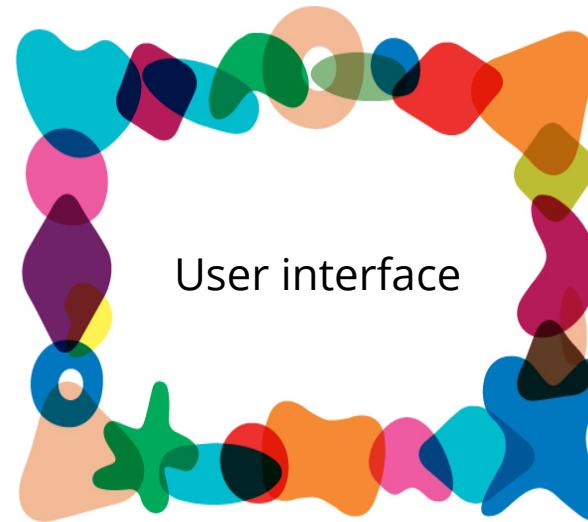


performance testing



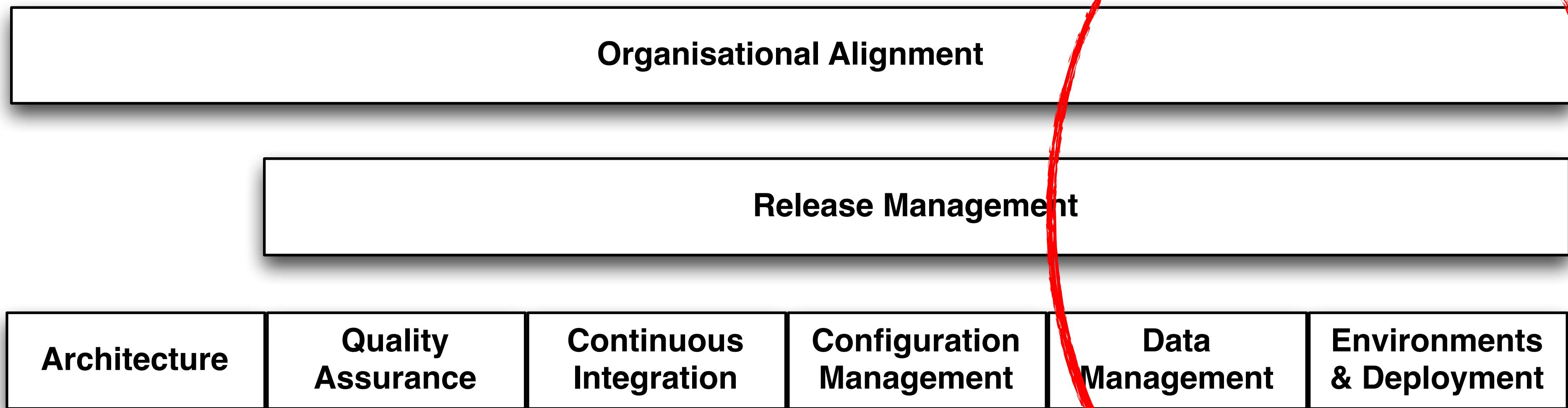
multi-variant testing

Dark Launching

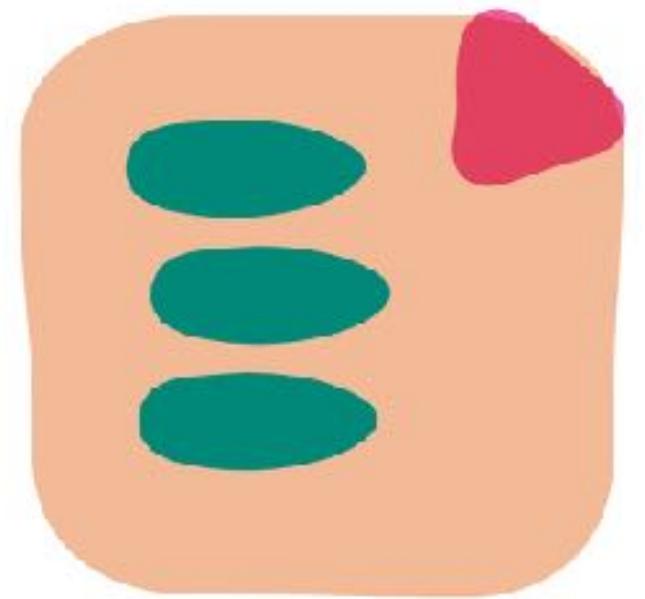


timeline

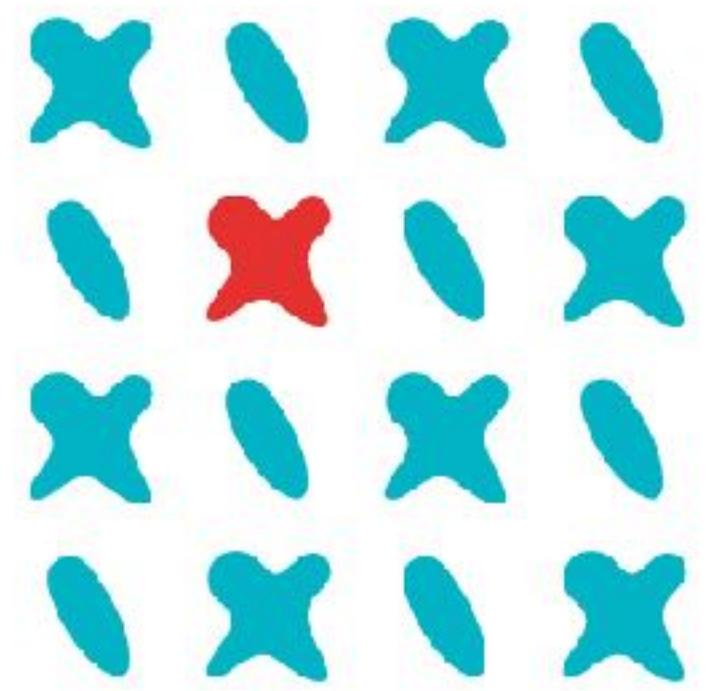
Continuous Delivery



essential complexity in data



persistent

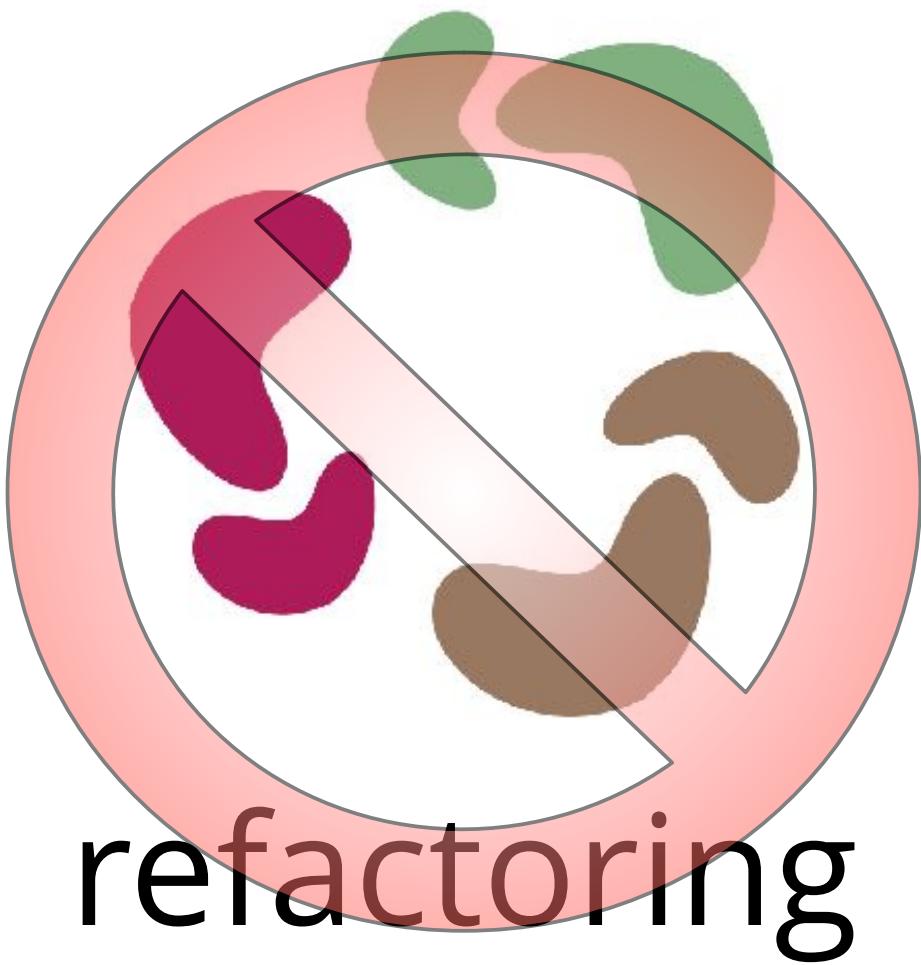


impedance mismatch



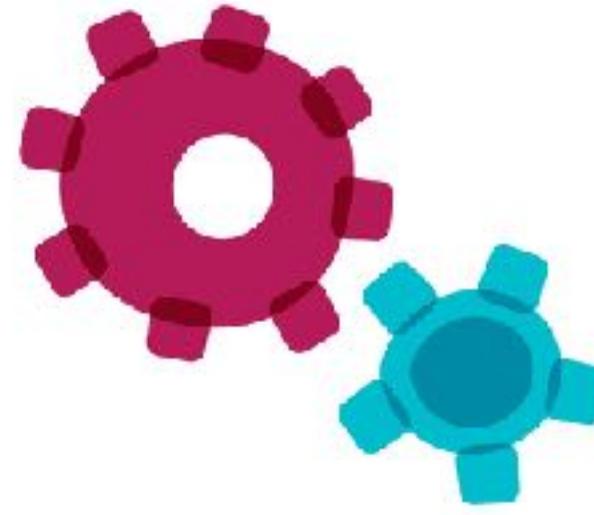
bulky

accidental complexity in data

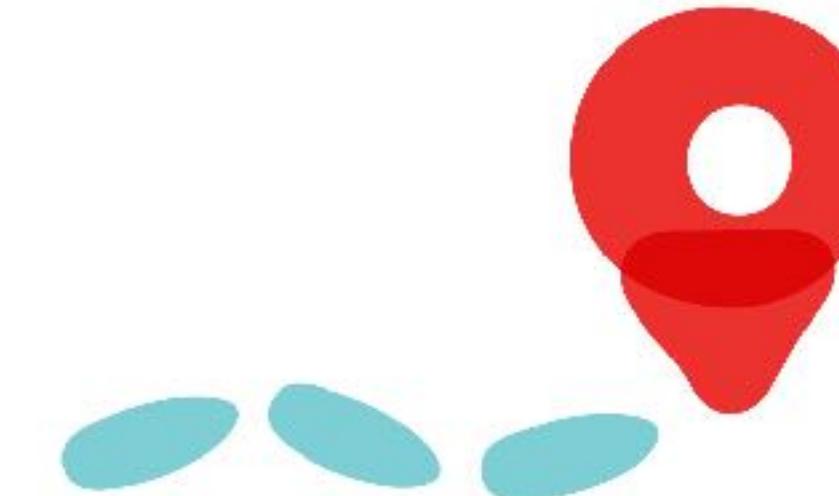
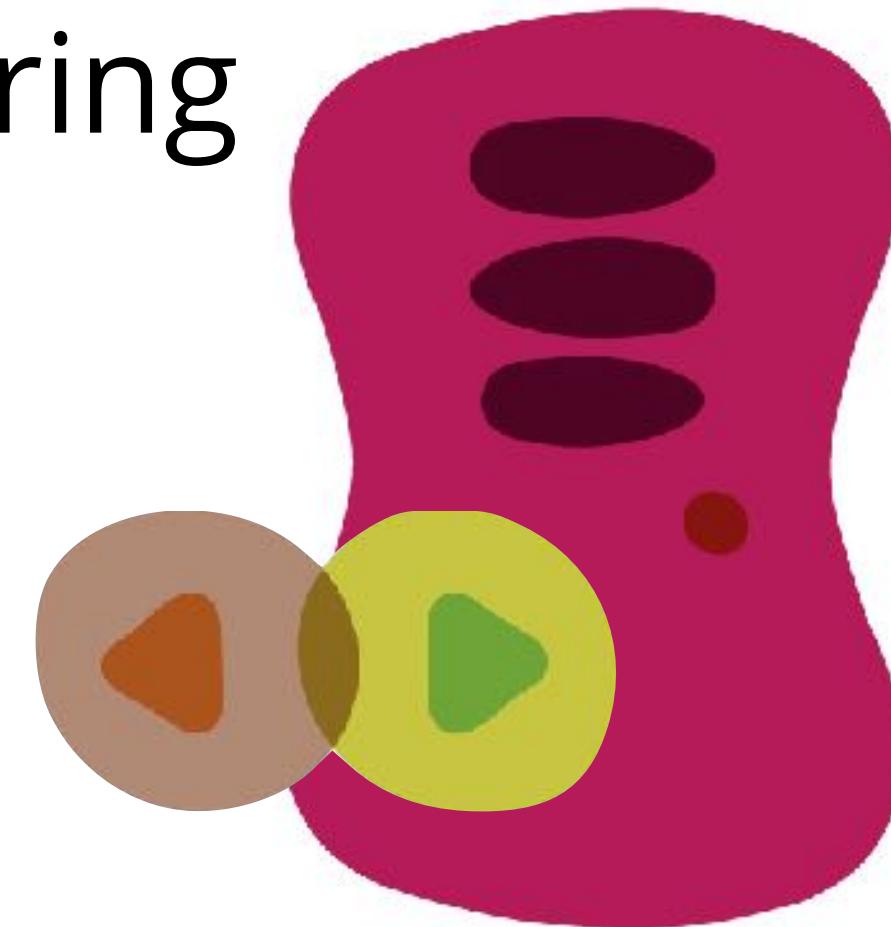


DB Evolution & Deployment

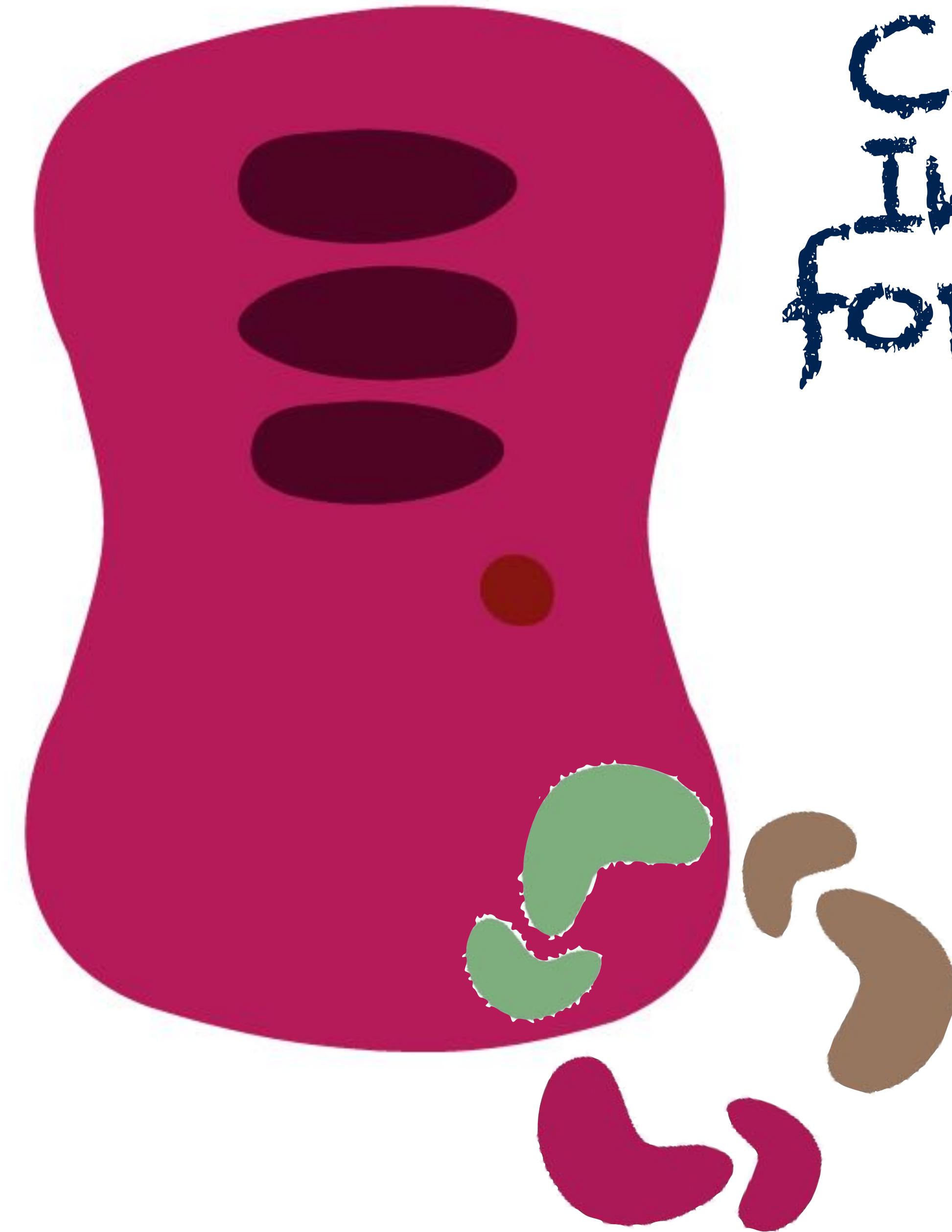
scripting all db changes incrementally



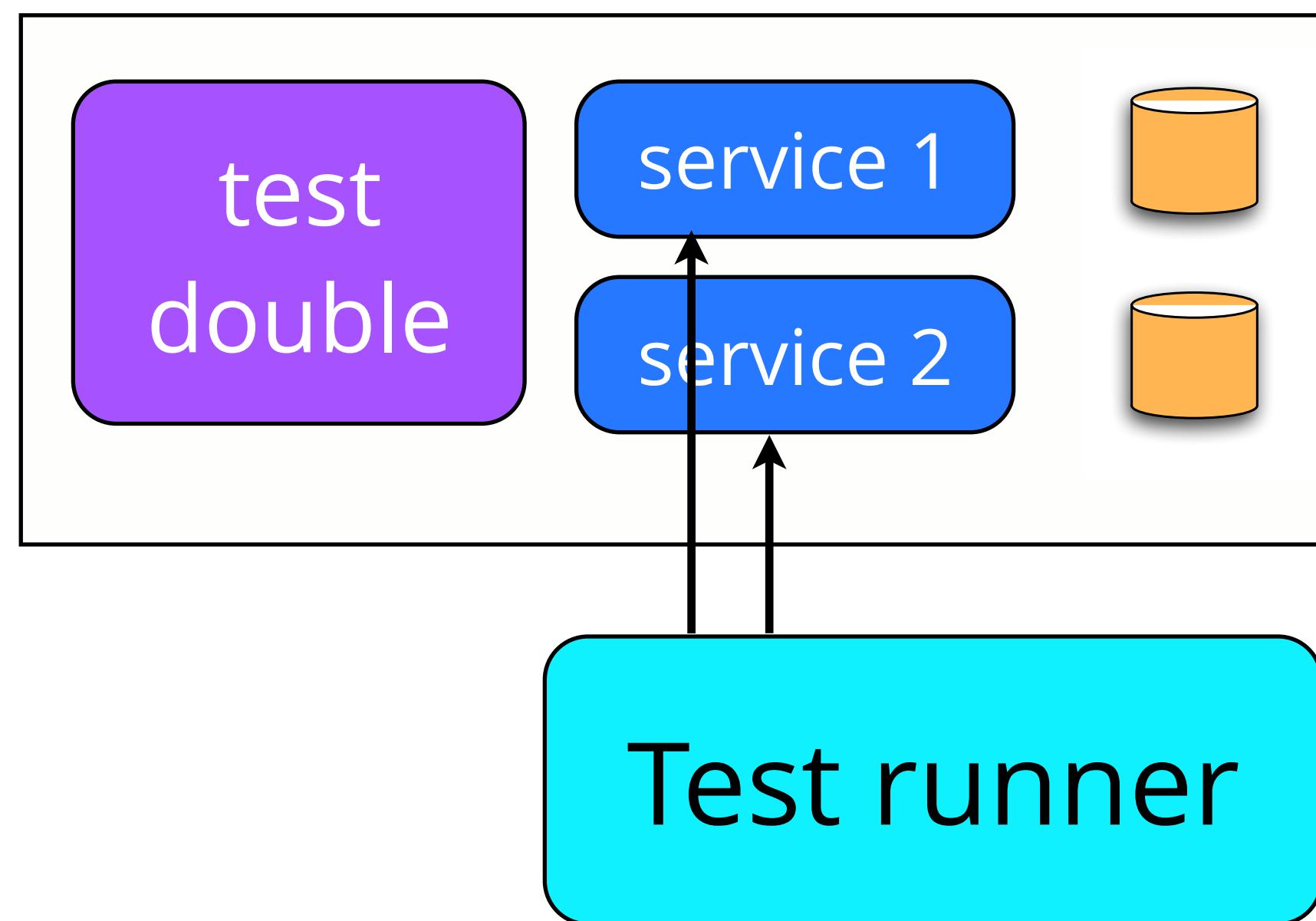
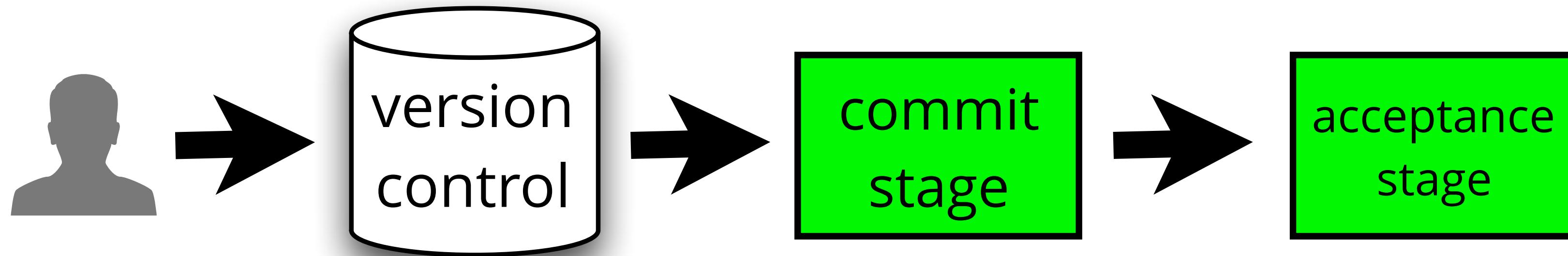
db refactoring



decouple db migration
from app migration

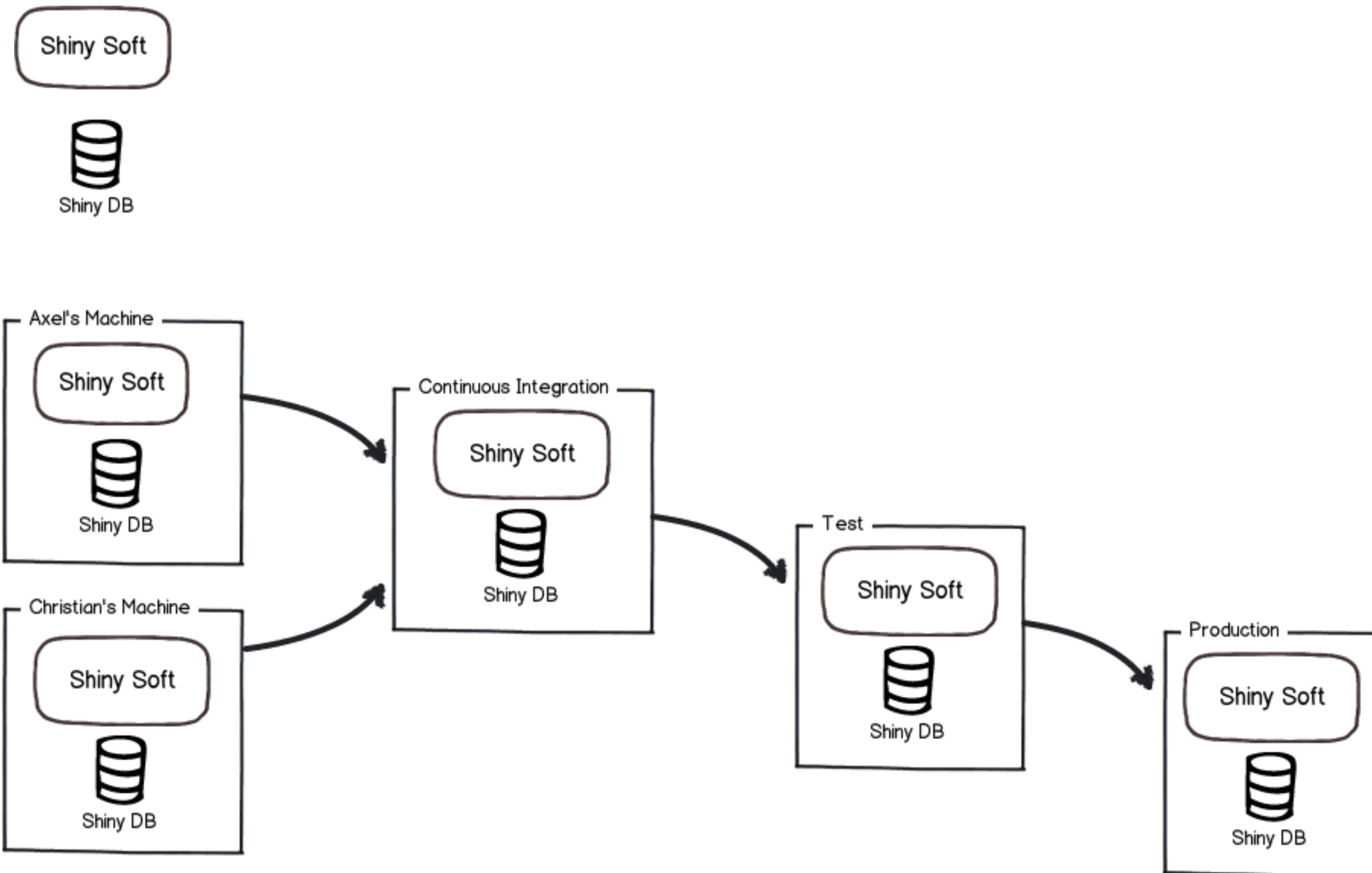


Continuous Integration for Databases



Prepare environment
Deploy app
Create dbs, apply schema
Add app reference data
Run acceptance tests

DbDeploy Pattern



DbBeploy Tool

db updates are code

small incremental deltas

metadata in the database

fail fast

<http://dbdeploy.com>

<http://www.liquibase.org/>

<https://flywaydb.org>

001_create_initial_tables.sql:

```
CREATE TABLE customer (
    id      BIGINT GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
    PRIMARY KEY,
    firstname VARCHAR(255),
    lastname  VARCHAR(255)
) ;
```

002_add_customer_date_of_birth.sql

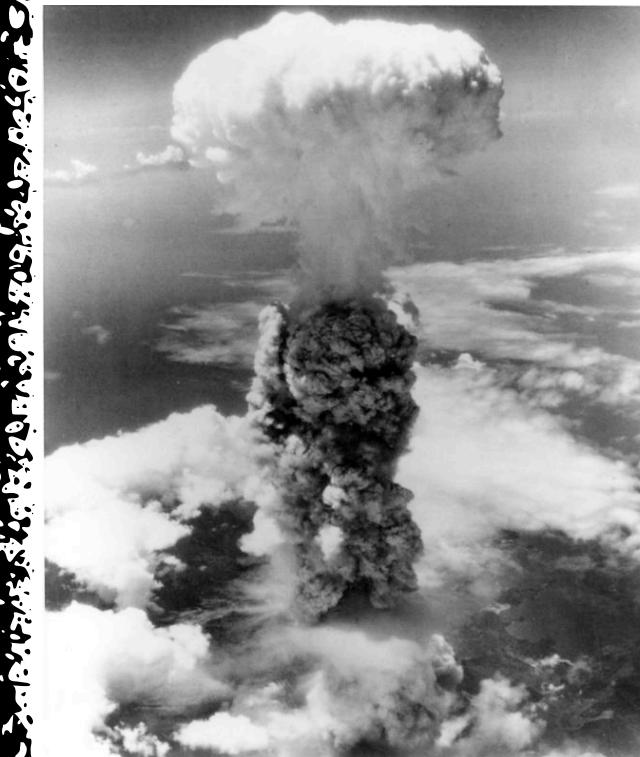
```
ALTER TABLE customer ADD COLUMN dateofbirth DATETIME;
```

```
-- // @UNDO
```

```
ALTER TABLE customer DROP COLUMN dateofbirth;
```

For DB CI We Need To:

start with a clean database



apply changes incrementally

use the same process everywhere

be comprehensive in change management

Apply Deltas

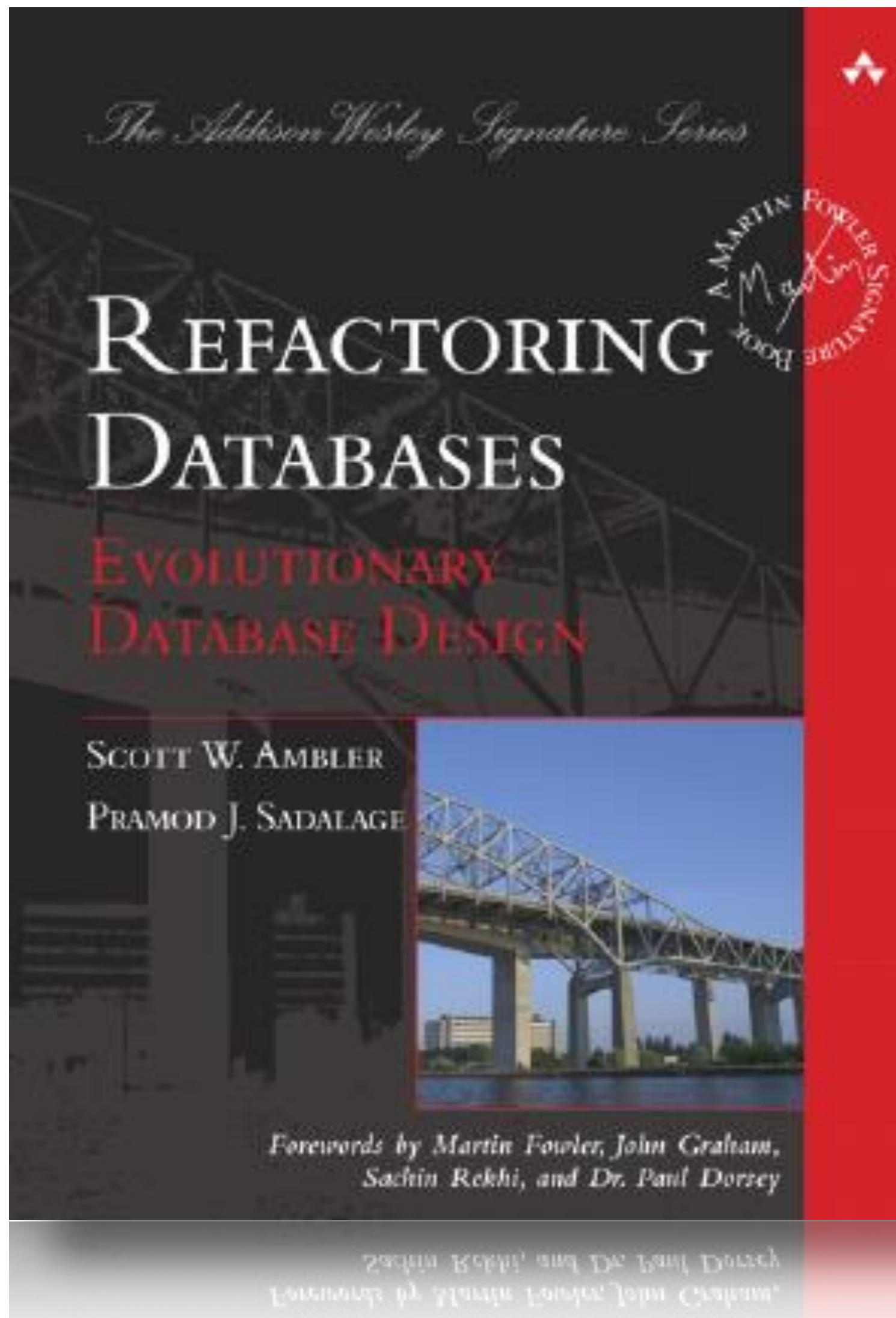
run each delta in order

stop the line if one delta fails

auto-rollback if possible

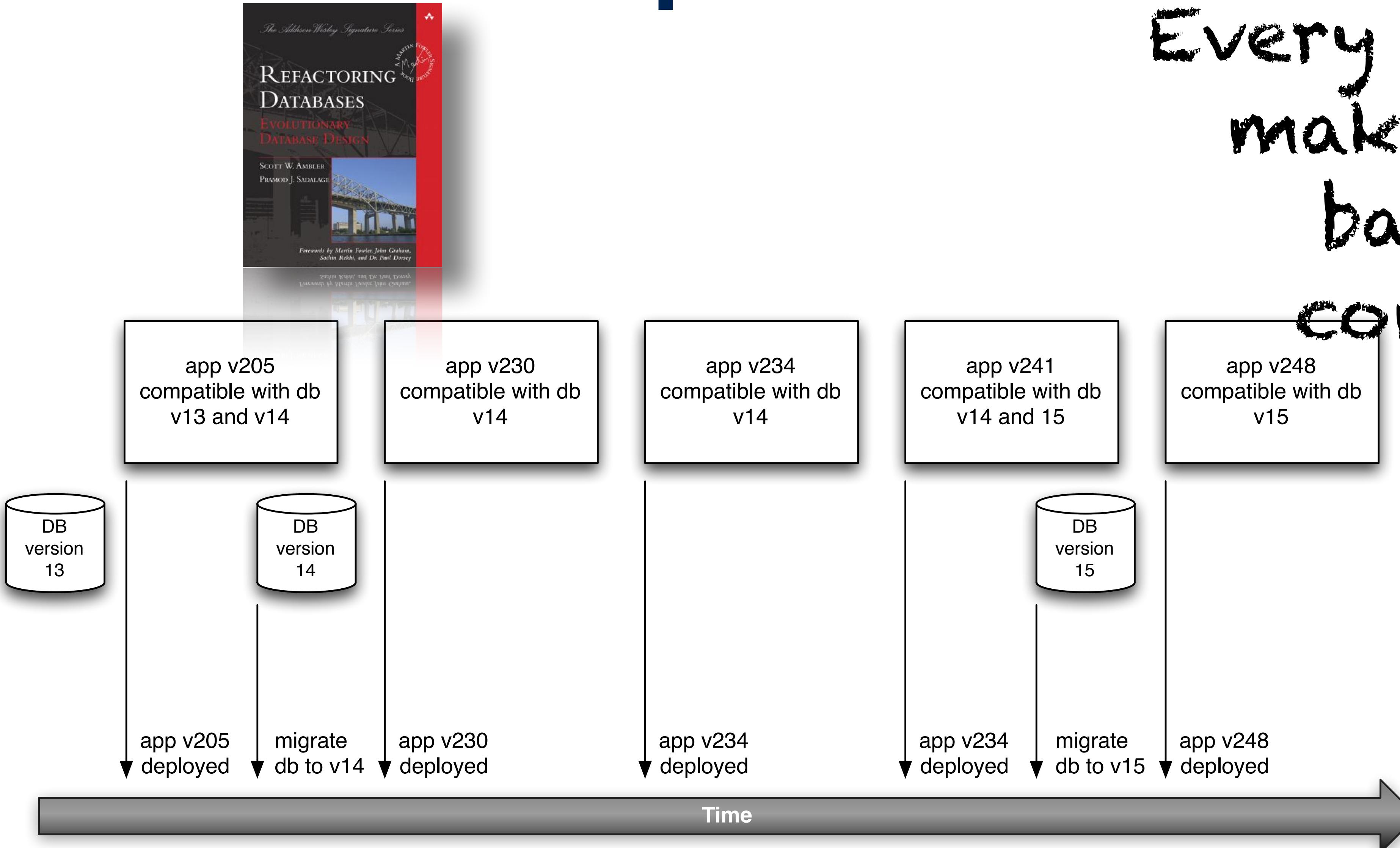
record success in db metadata table

Refactoring Databases



Decouple DB Updates: the Expand/contract Pattern

Every change you
make must be
backward
compatible



decouple db updates

dark launch db updates

long-running upgrades

abstraction layer in code or stored procs /
views

Never tie DB
migrations to
application deploys

DB Deployments Still Hard

practice, practice, practice

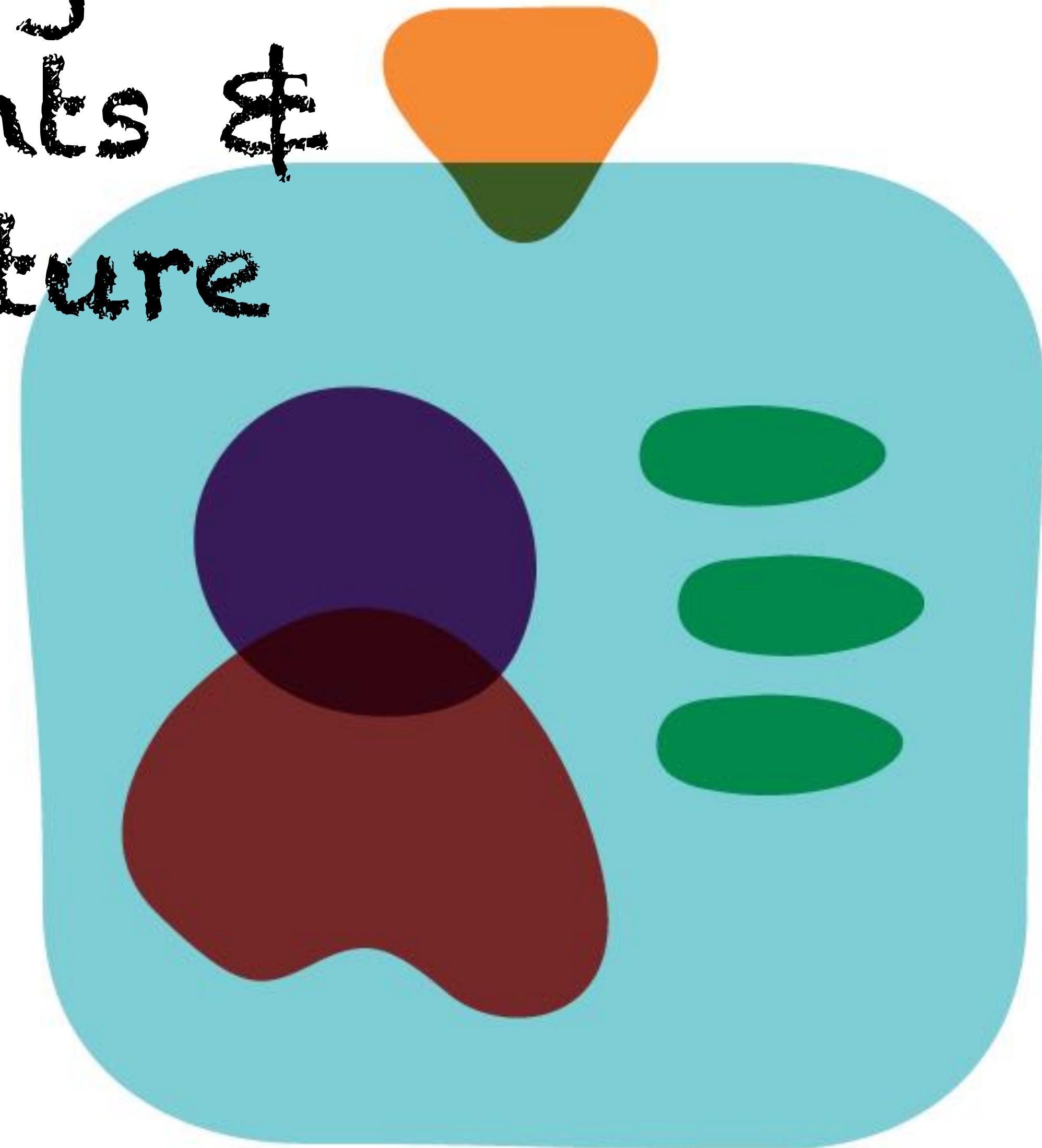
fail fast

bring the pain forward

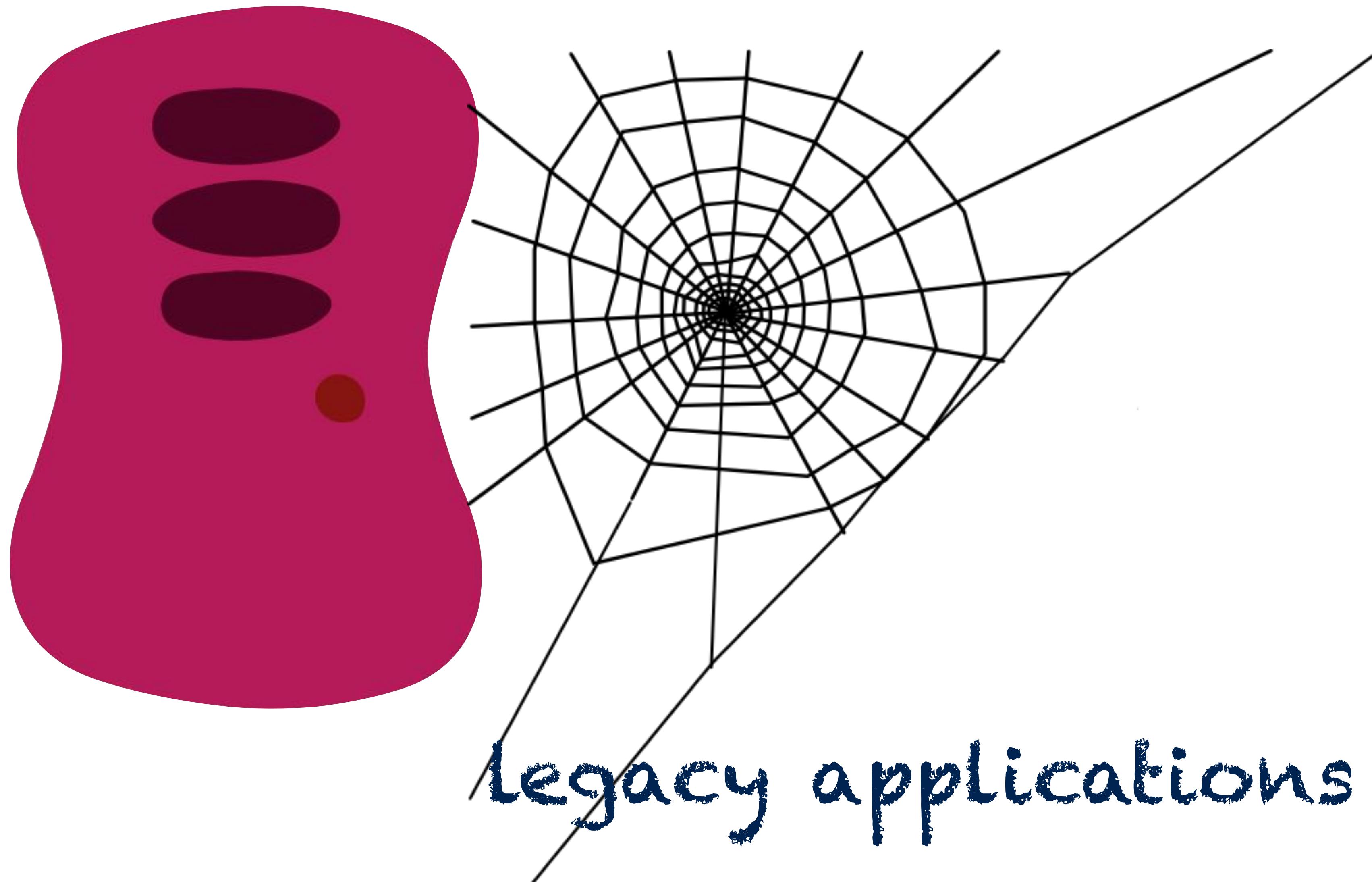
refactor the db

update engineering practices

Managing Environments & Infrastructure

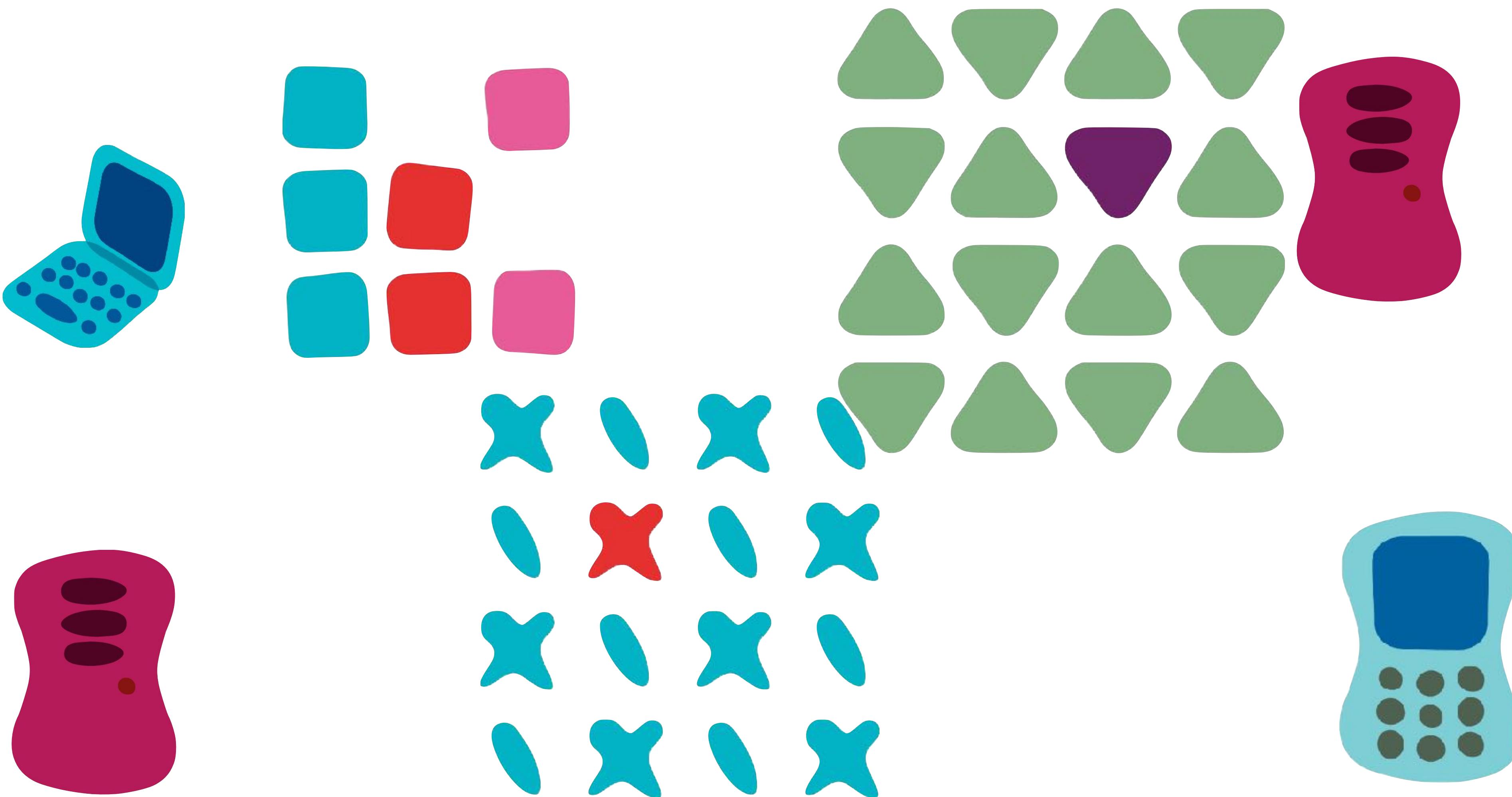


The Pain of Operations

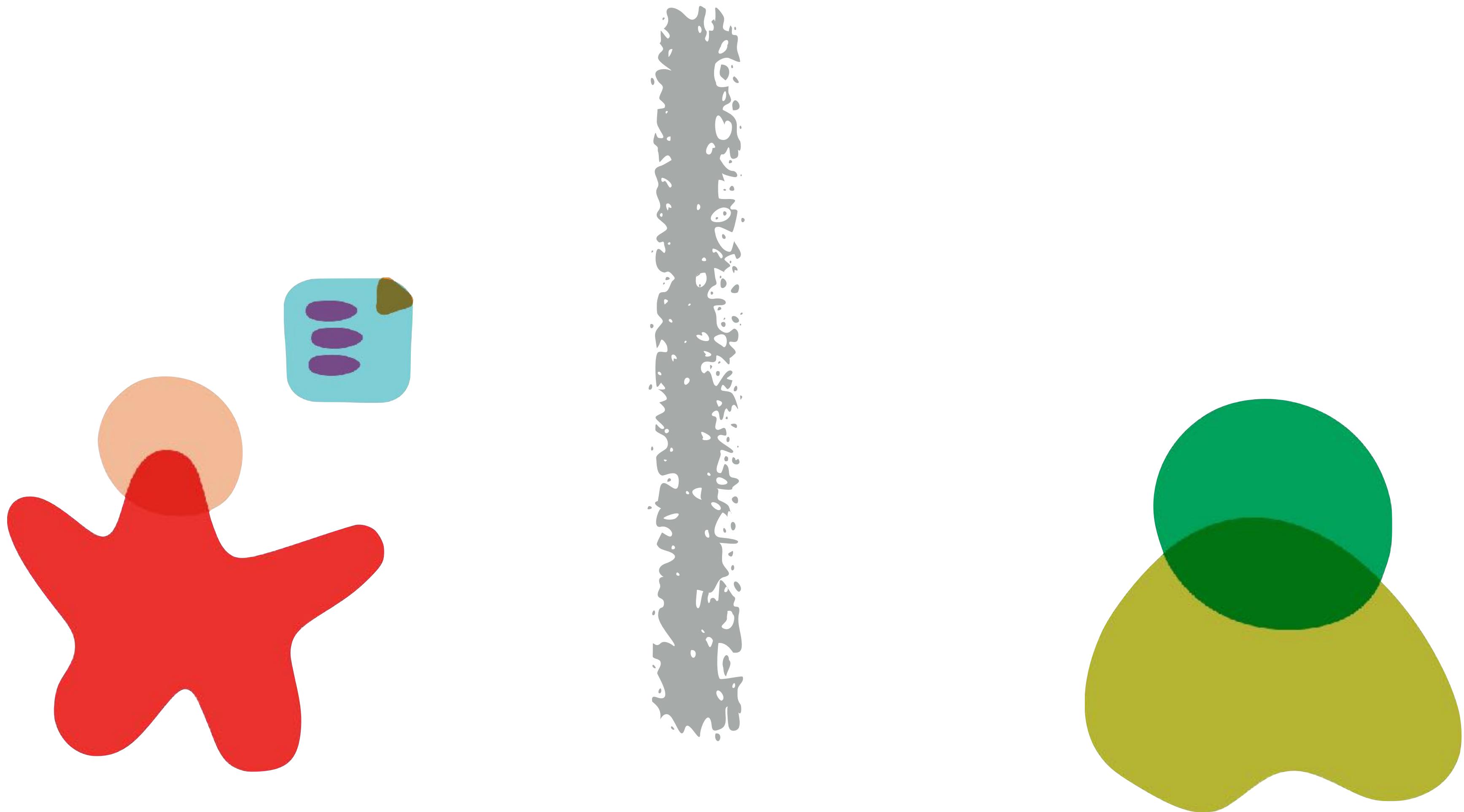


The Pain of Operations

heterogeneous platforms

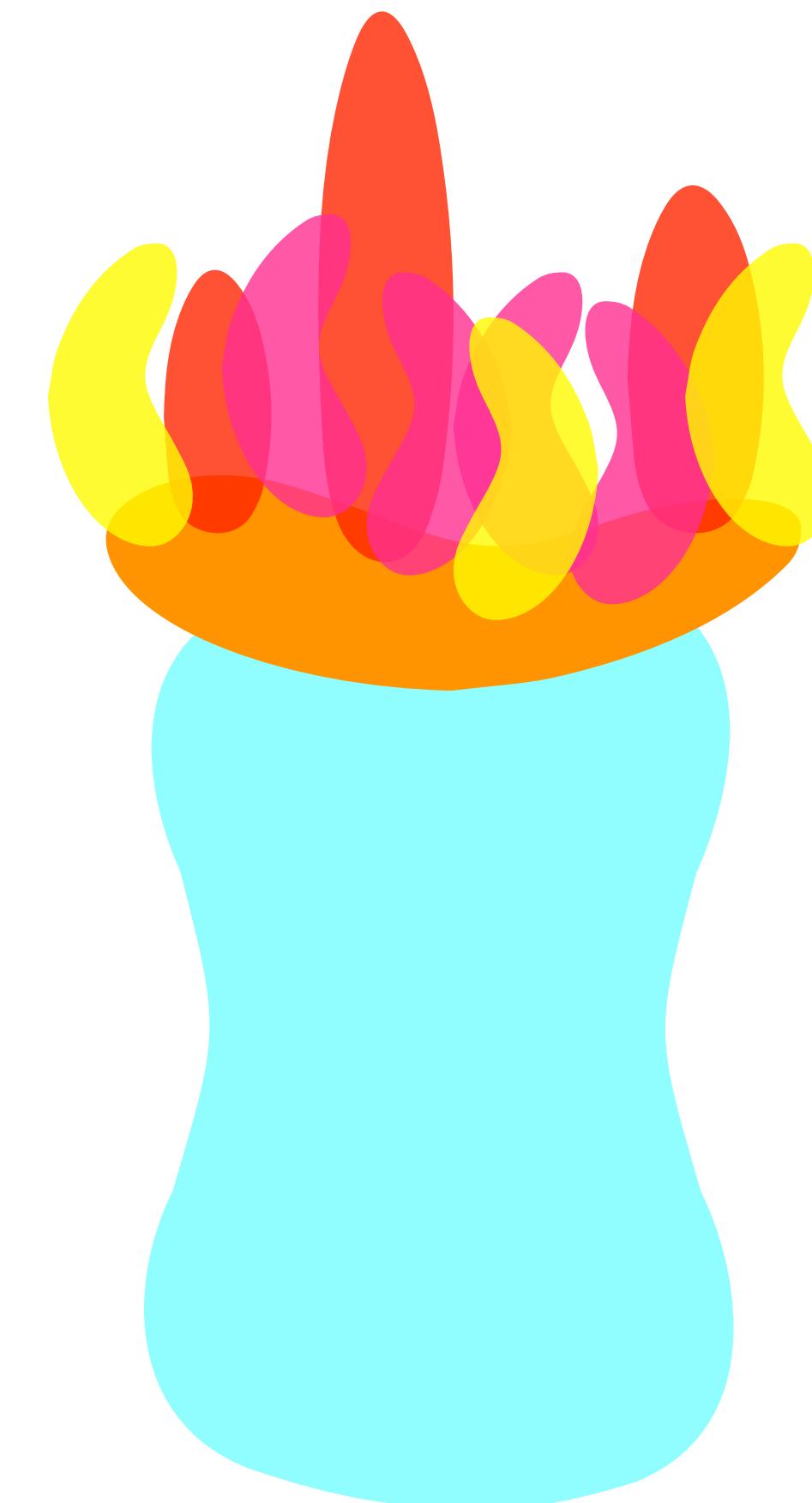


The Pain of Operations



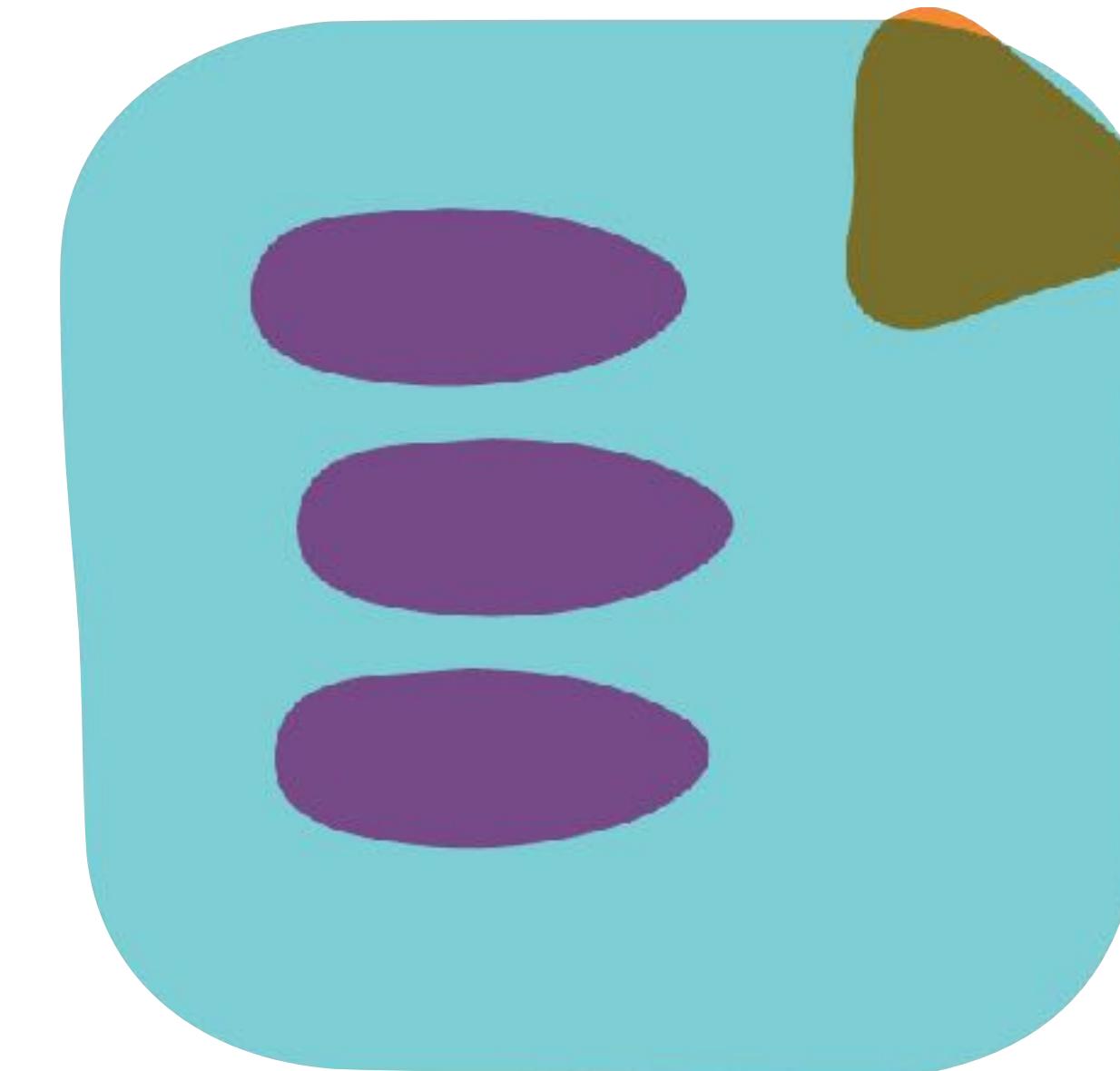
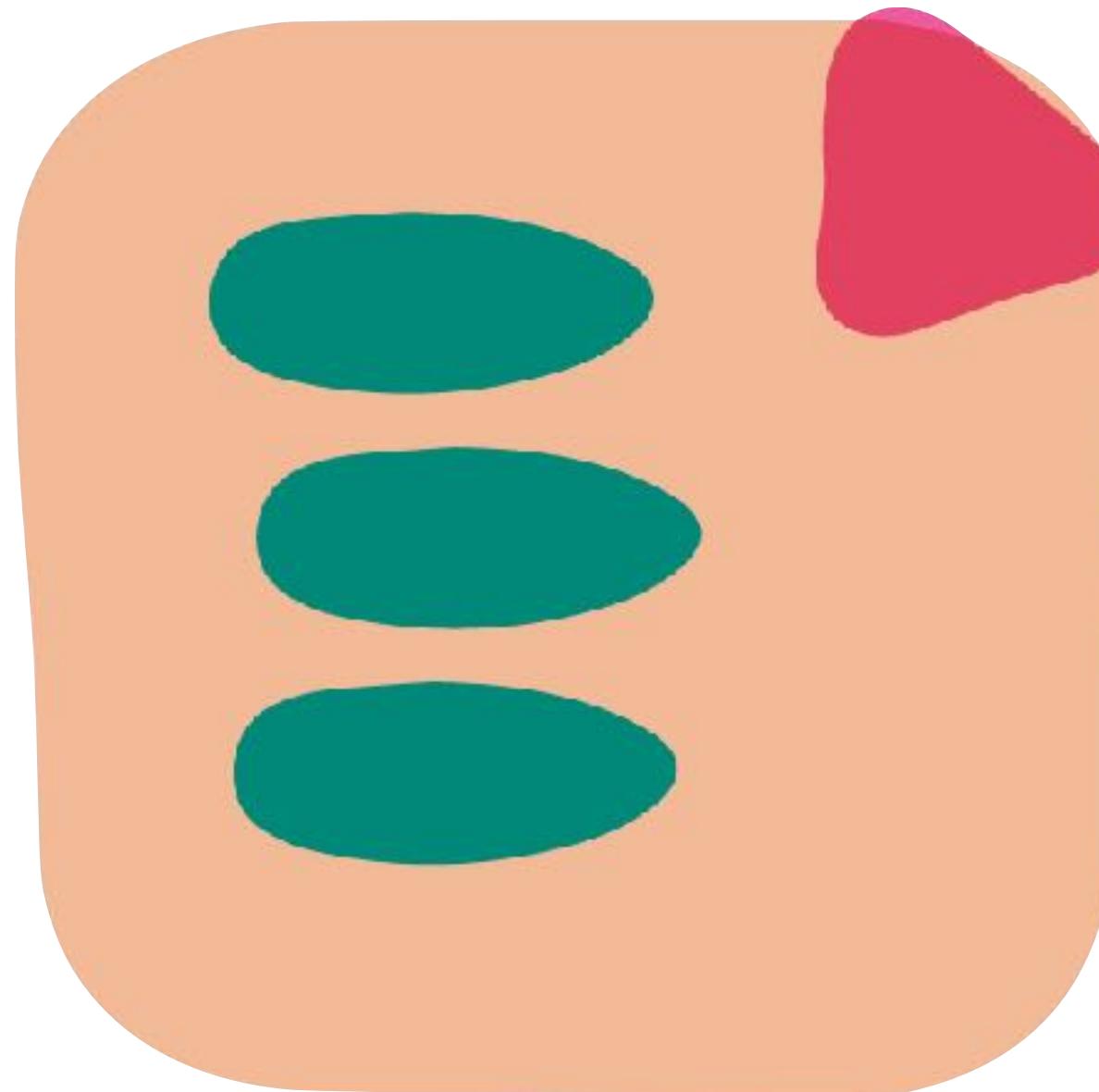
poor quality software thrown over a wall

The Pain of Operations



inordinate amount of firefighting

The Pain of Operations



conservative, process heavy

The Pain of Operations



huge budget for operations

Horror Stories

dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/

The screenshot shows a blog post on the website dougseven.com. The title of the post is "Knightmare: A DevOps Cautionary Tale". The author is Doug Seven, whose profile picture is a man wearing a baseball cap and sunglasses. The post was published on April 17, 2014, and has 37 comments, 70 votes, and a 5-star rating. The content of the post discusses a true story of a company that went bankrupt in 45 minutes due to a failed deployment. The post is part of a series of "Horror Stories" on the site.

DOUG SEVEN
Something can be learned in the course of observing things

HOME IOT WORKSHOP PRODUCING AWARENESS

"bankrupt in 45 minutes"

You are here: Home / DevOps / Knightmare: A DevOps Cautionary Tale

Knightmare: A DevOps Cautionary Tale

APRIL 17, 2014 BY D7 37 COMMENTS 70 Votes

★★★★★ 70 Votes

I was speaking at a conference last year on the topics of DevOps, Configuration as Code, and Continuous Delivery and used the following story to demonstrate the importance making deployments fully automated and repeatable as part of a DevOps/Continuous Delivery initiative. Since that conference I have been asked by several people to share the story through my blog. This story is true – this really happened. This is my telling of the story based on what I have read (I was not involved in this).

This is the story of how a company with nearly \$400 million in assets went bankrupt in 45-minutes because of a failed deployment.

Background

DOUG SEVEN

SEARCH

Search this website... Search

RECENT POSTS:

IoT Workshop: Lesson 2 - Input

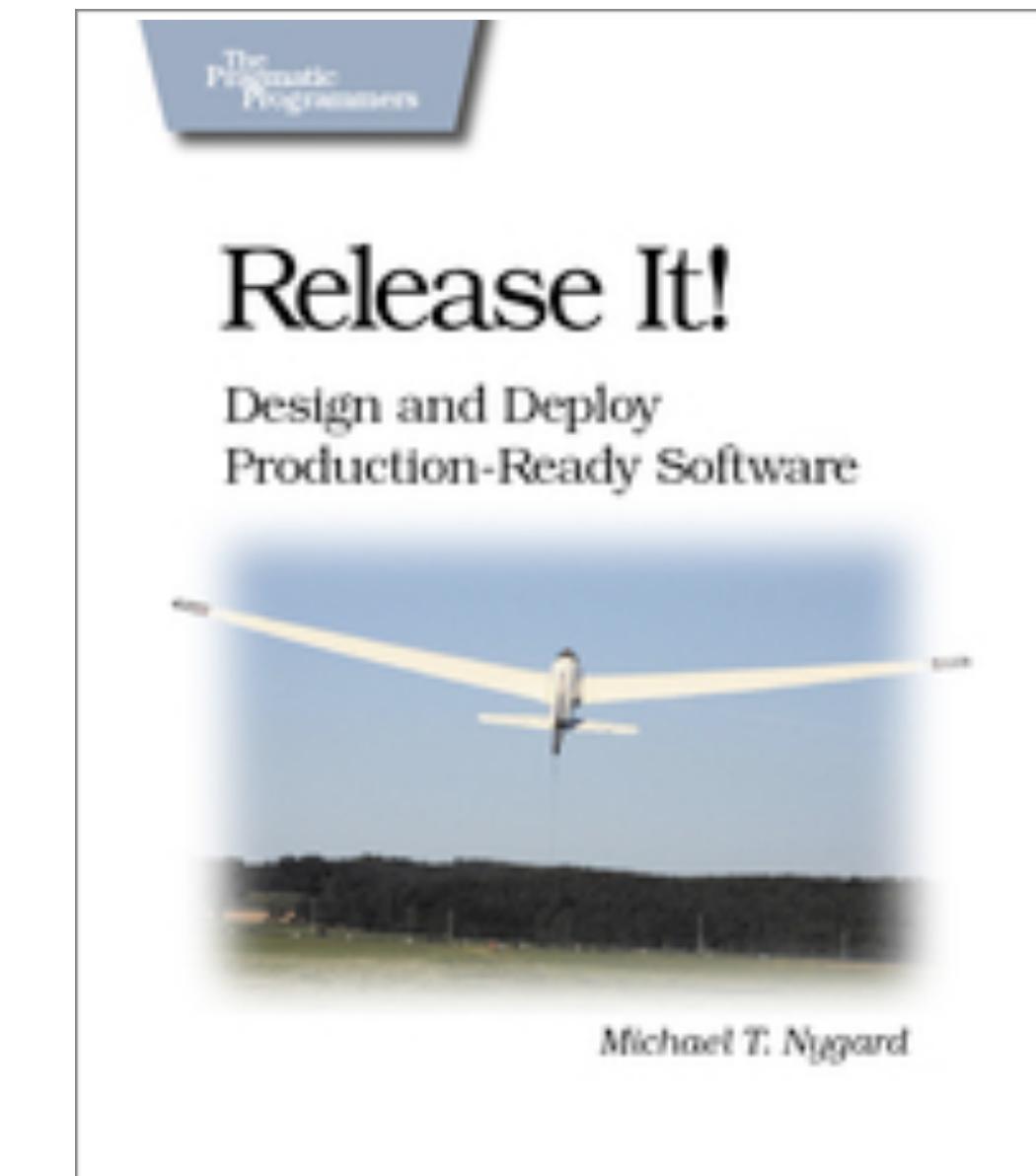
DevOps

not it's own silo, but a liaison between operations and developers

at inceptions, showcases, retros

devs work in ops and carry pagers

devs create more deployable software



Managing Infrastructure

infrastructure = environments and supporting services (networking, vcs, storage, mail, dns...)

desired state specified in version control

autonomic (self-corrects to desired state)

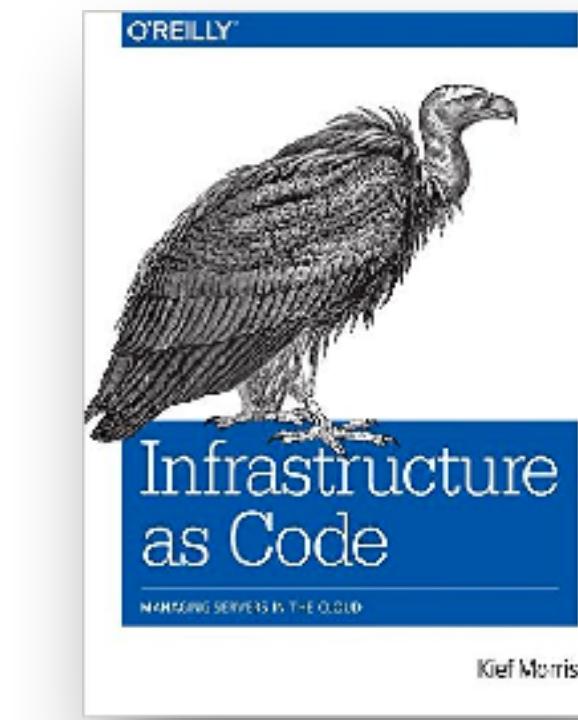
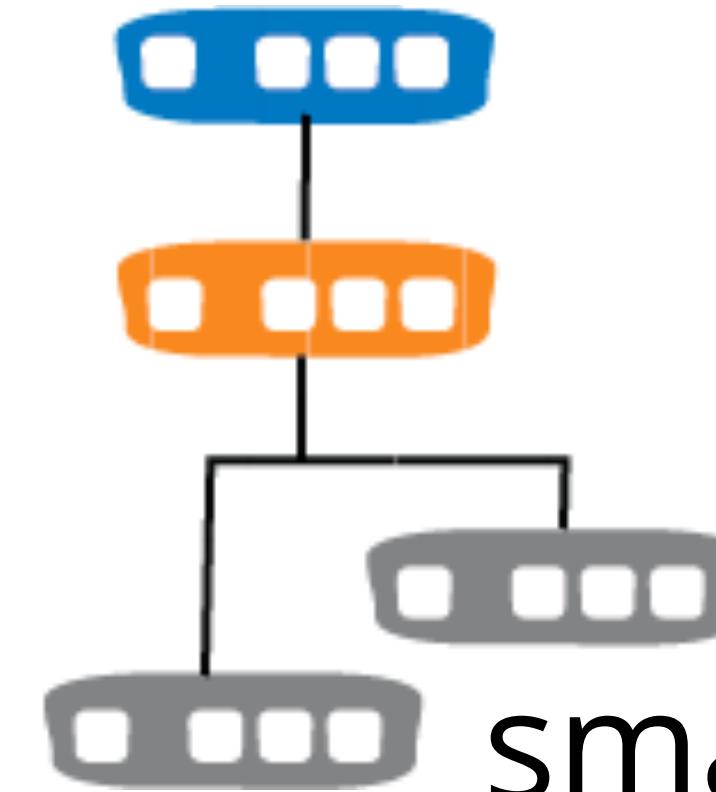
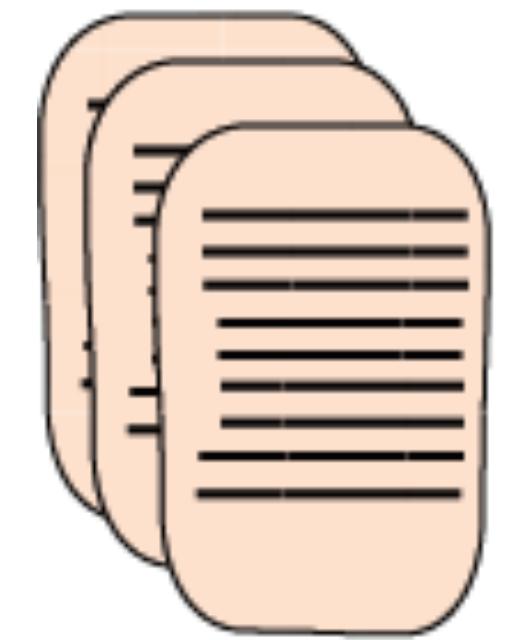
state should be known through monitoring

Infrastructure as Code

version all the things

continuously test
systems & processes

definition files

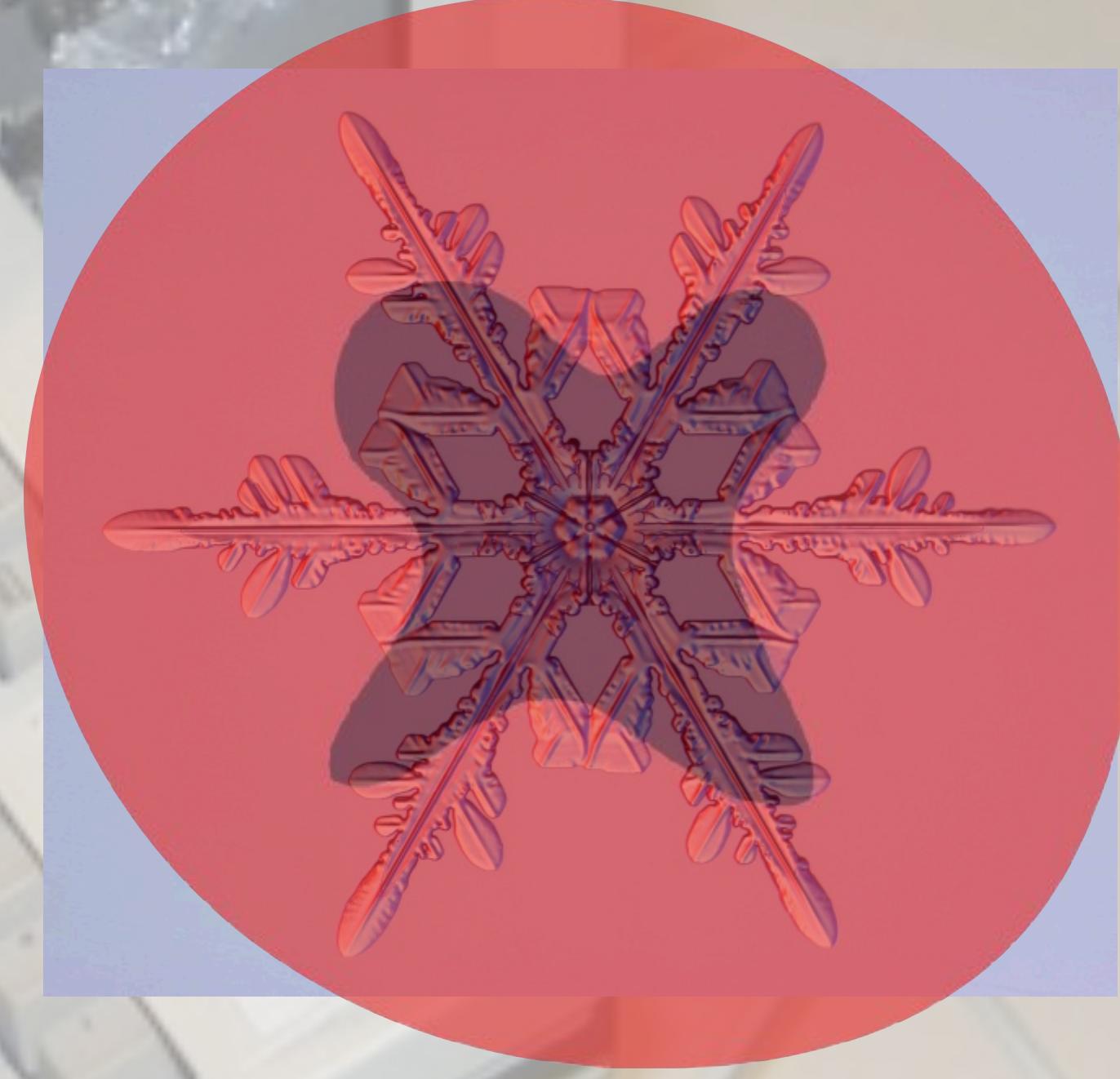


self-documented
systems & processes

small changes
over large batches

keep services available continuously

Destroy Works of Art



If someone threw a server out of the window, how long would it take to recreate it?

Tools

The screenshot shows a web browser window displaying a list of DevOps tools on the website devopsbookmarks.com. The left sidebar contains a navigation menu with sections for 'TOPICS' and 'PLATFORM'. The 'TOPICS' section includes links for Source Code Management, Continuous Integration & Delivery, Packaging & Artifacts, Virtualization & Containers, Cloud & PaaS Environments, Configuration Management, Provisioning (which is checked), Orchestration, Service Discovery, Process Management, Logging & Monitoring, Metrics & Visualization, and Security & Hardening. The 'PLATFORM' section includes links for Linux, Windows, and OSX. The main content area lists several tools in a grid:

- Ansible**: A versatile orchestration engine that can automate systems and apps. Instead of a custom scripting language or code, it is very simple and shell based. It is also agent-less, so you can just start using it right away and get things done.
- Dokku Alt**: Dokku on Steroids. The smallest PaaS implementation you've ever seen. It's fork of original dokku. The idea behind this fork is to provide complete solution with plugins covering most of use-cases which are stable and well tested.
- Batou**: Batou makes it easy to perform automated deployments. It combines Fabric's simplicity and SSH automation, with Puppet's declarative syntax and idempotence.
- Dokku**: It uses docker, git-receive and a few other lightweight and clever libraries to build a quick PaaS, all around just 100 lines of code! An excellent small tool to get started with PaaS systems. The same developer is creating a larger scale, production quality system called Flynn.
- Bcfg2**: bee-config (Bcfg) 2 is a centralized configuration management server to configure large number of systems, built
- FAI**

Each tool entry includes a small icon set below its description and a link to the tool's page at www.devopsbookmarks.com/.

www.devopsbookmarks.com/



ANSIBLE



Tools

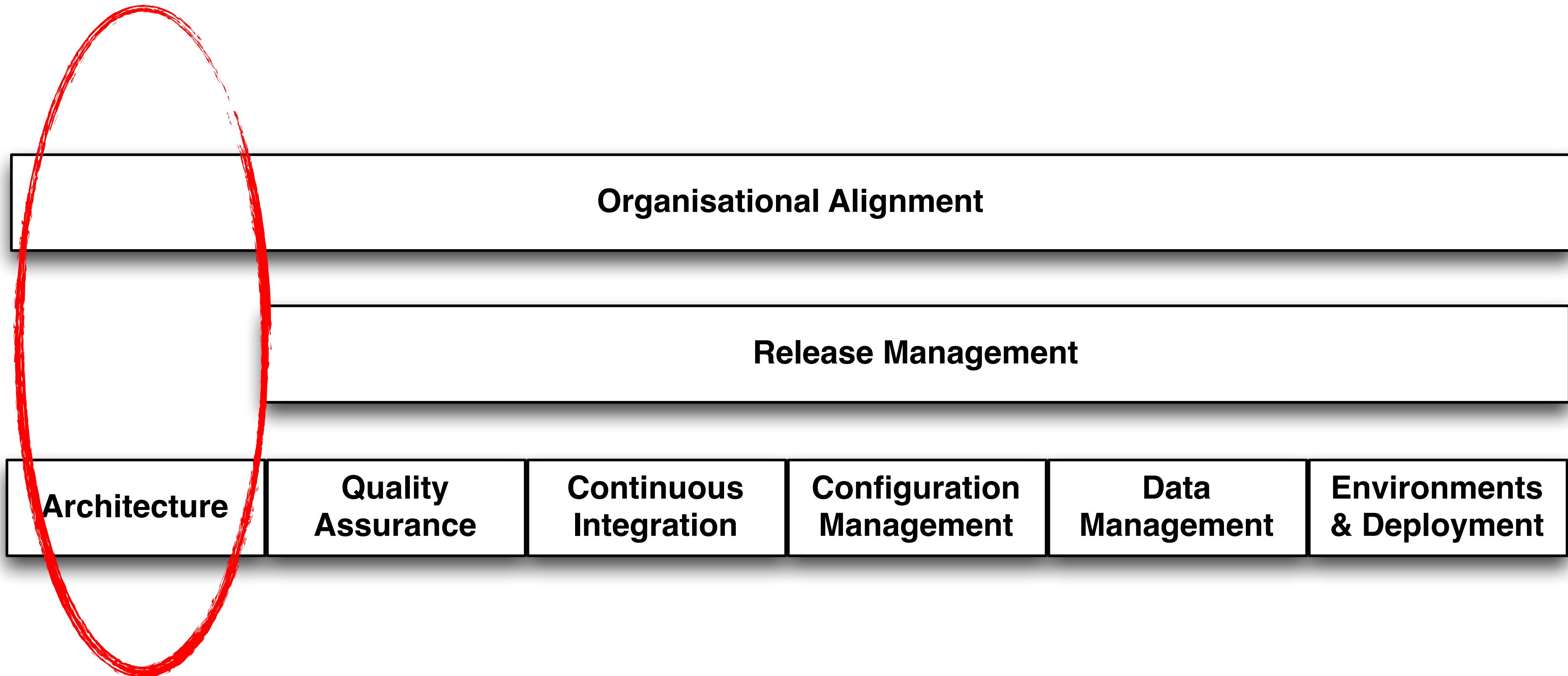
manage many systems

manage configuration

enforce consistency

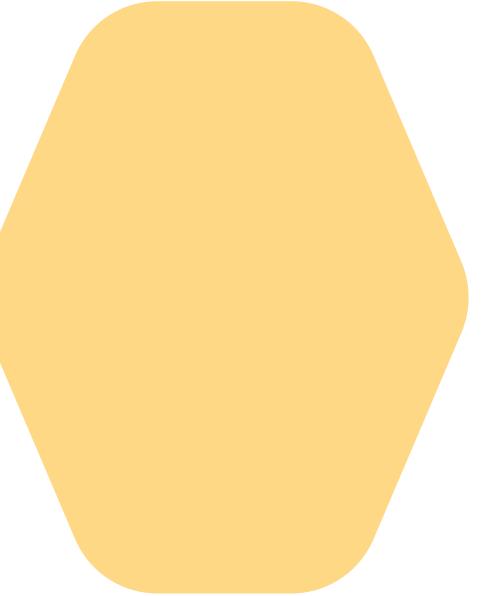
treat infrastructure as code

Continuous Delivery



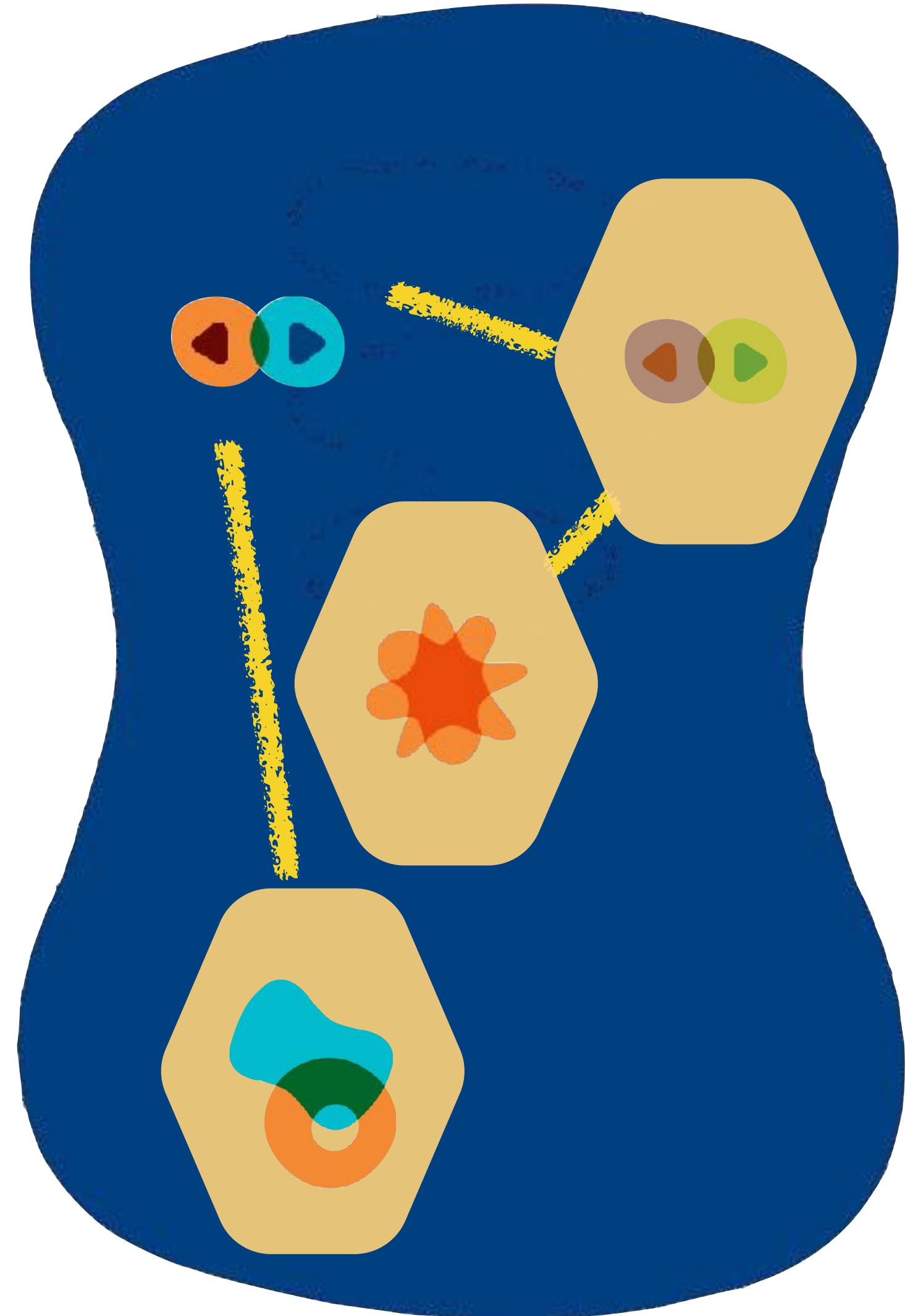
Microservices

Components are
deployed.



Features are *released.*

Applications consist
of *routing.*

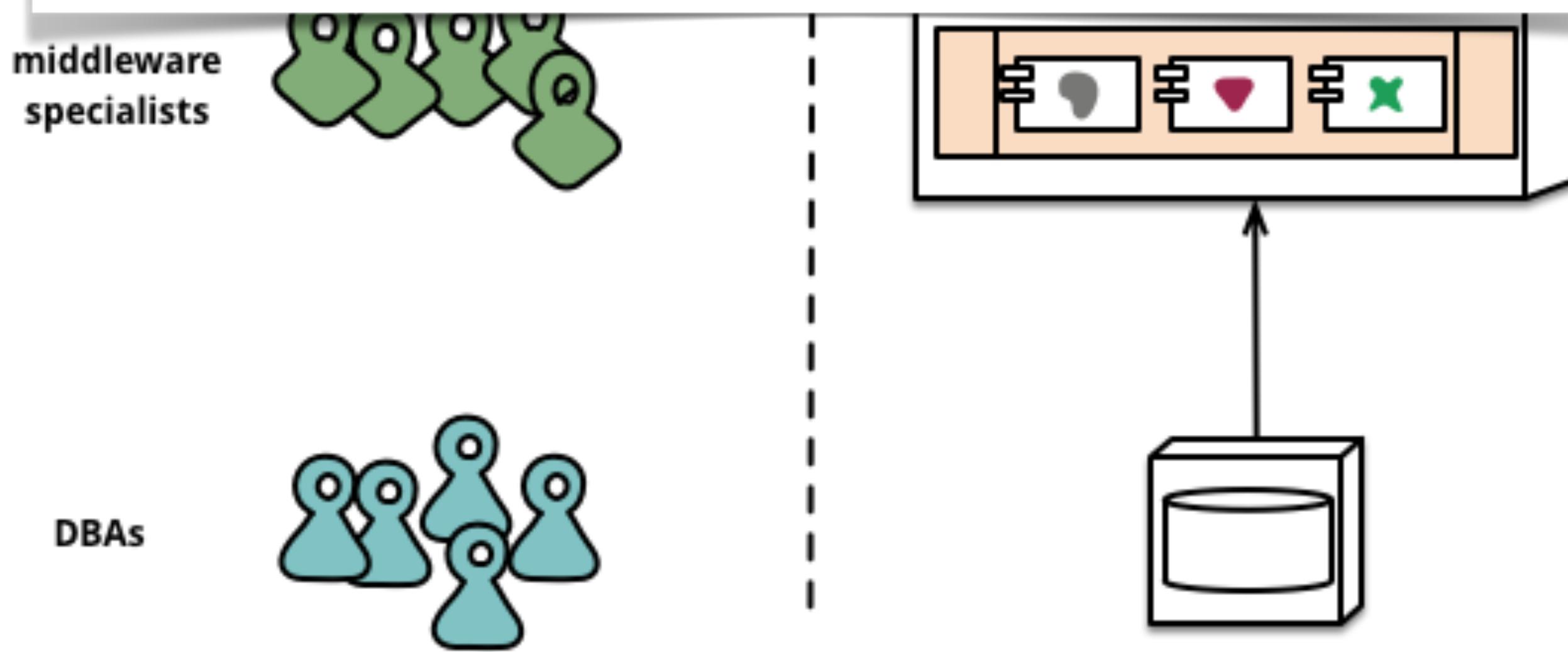


production

Conway's Law

"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations"

—Melvin Conway

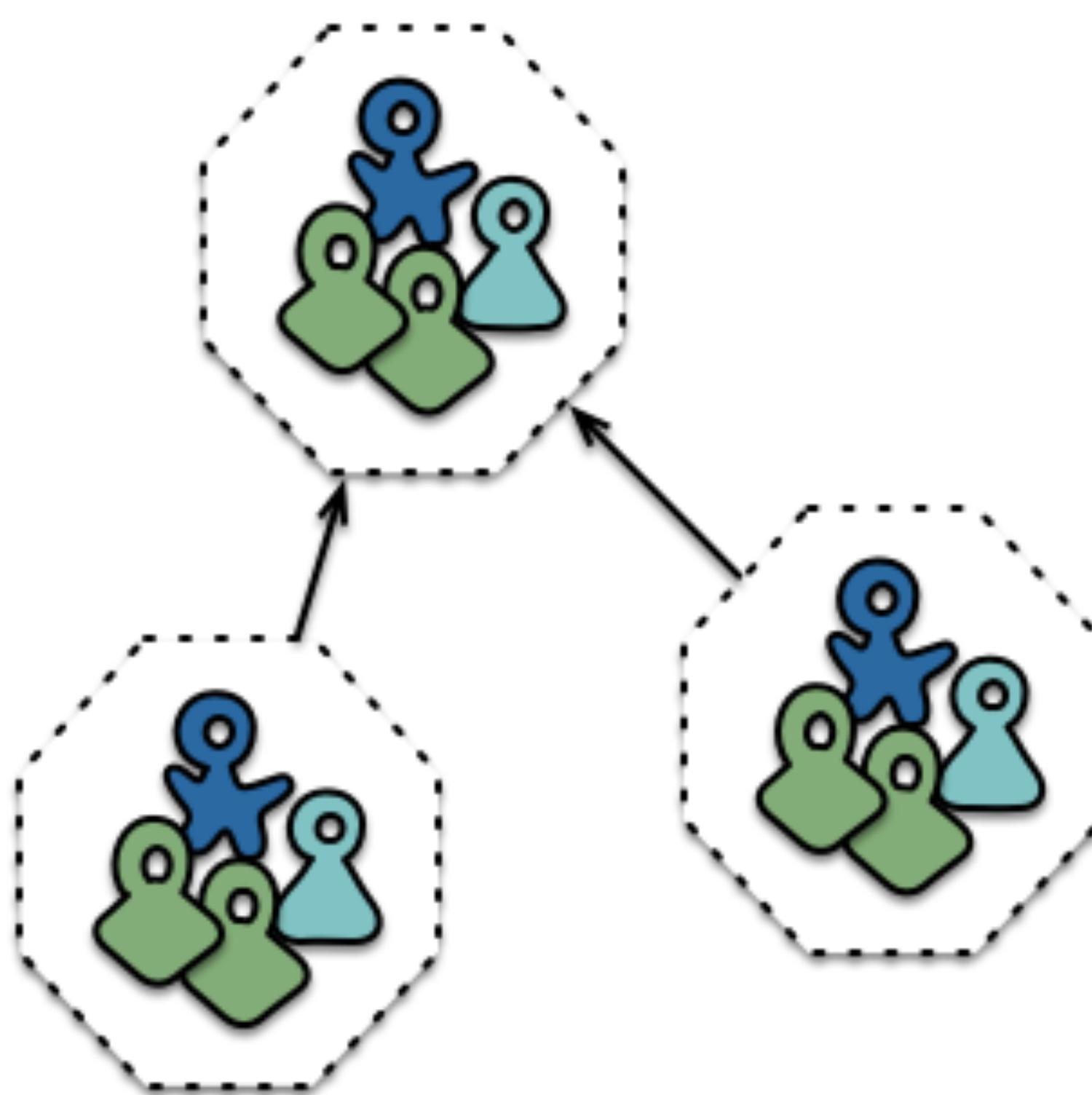


Siloed functional teams...

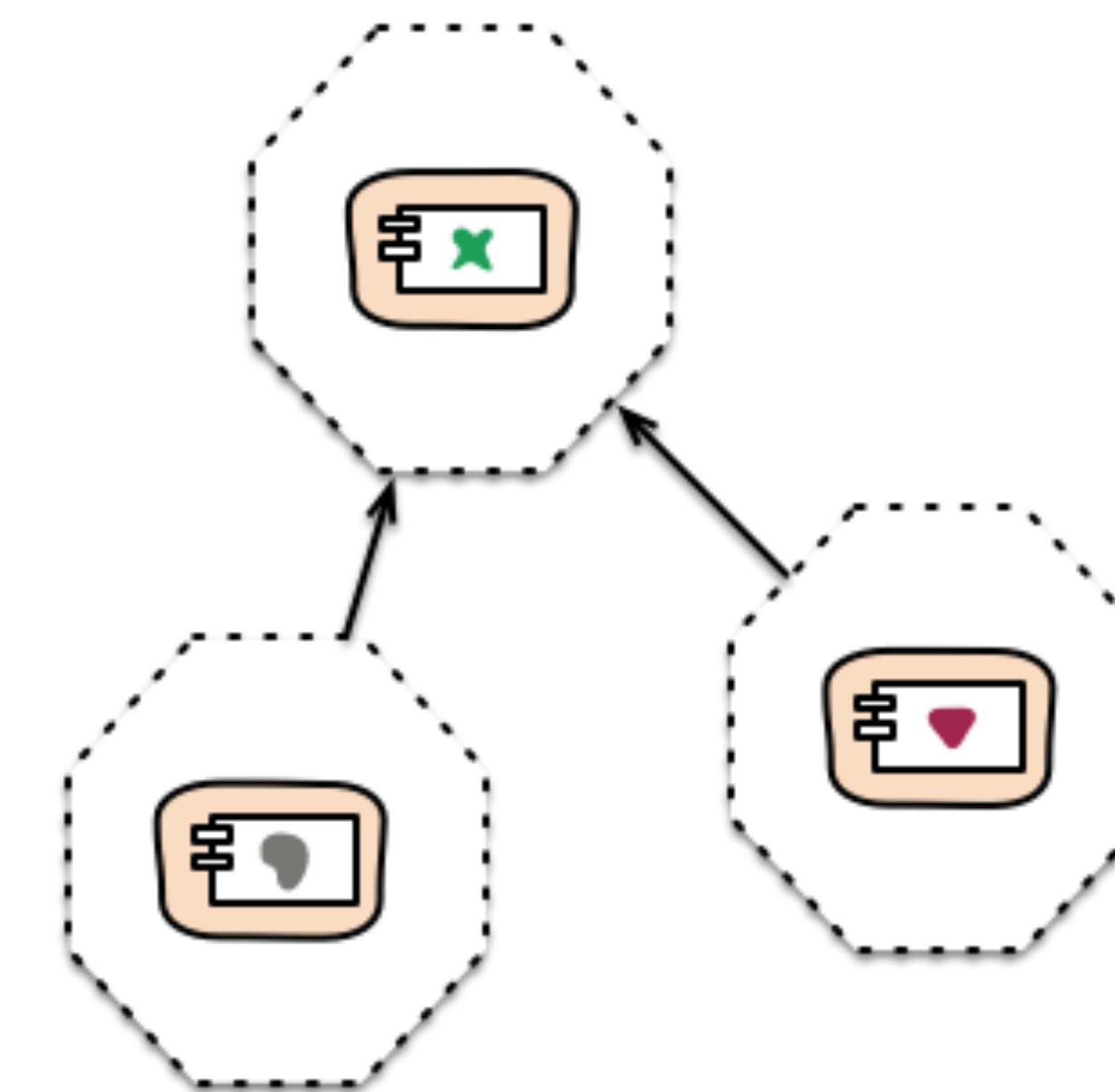
... lead to siloed application architectures.

Because Conway's Law

Inverse Conway Maneuver



Cross-functional teams...



... organised around capabilities
Because Conway's Law

“team designs are the first draft of your architecture”

- Michael Nygard



“ ... it isn't the methodologies that succeed or fail, it's the teams that succeed or fail. Taking on a process can help a team raise its game, but in the end it's the team that matters and carries the responsibility to do what works for them. ”

MARTIN FOWLER
(FLACCID SCRUM, 2009)



Continuous Delivery

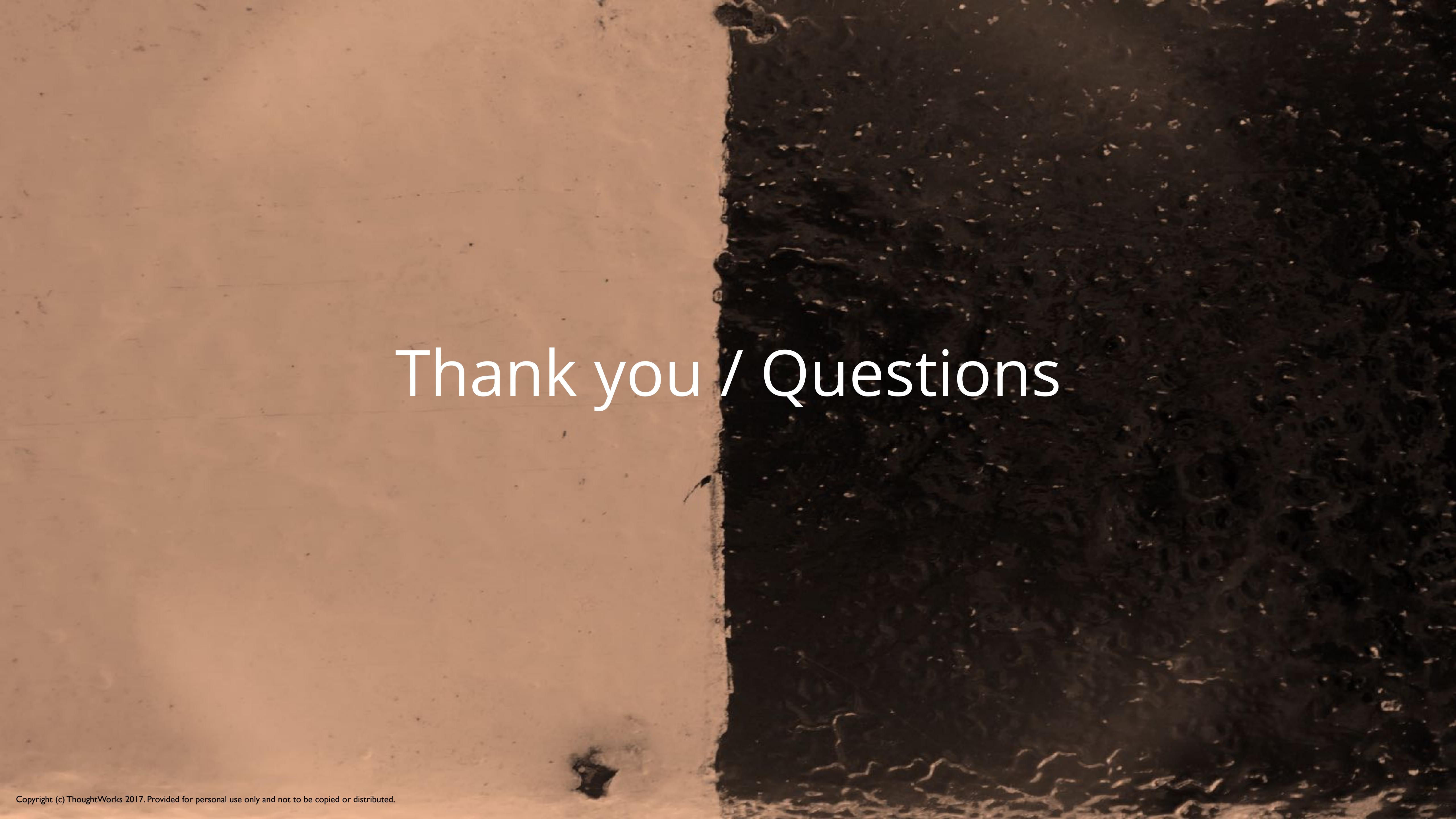
reduce friction

automate everything you can

incorporate everyone into Continuous
Delivery practices

measure success via cycle time

continue to improve



Thank you / Questions