

# Principles of Software Construction: Objects, Design, and Concurrency

## DevOps

**Christian Kästner**   Vincent Hellendoorn





# Topics

From CI to CD

Containers

Configuration management

Monitoring

Feature flags, testing in production

# Where we are

	<i>Small scale:</i> One/few objects	<i>Mid scale:</i> Many objects	<i>Large scale:</i> Subsystems
<i>Design for</i>	Subtype	Domain Analysis ✓	GUI vs Core ✓
understanding	Polymorphism ✓	Inheritance & Del. ✓	Frameworks and Libraries ✓ , APIs ✓
change/ext.	Information Hiding, Contracts ✓	Responsibility Assignment, Design Patterns, Antipattern ✓	Module systems, microservices ✓
reuse	Immutability ✓	Promises/ Reactive P. ✓	Testing for Robustness ✓
robustness	Types ✓	Integration Testing ✓	CI ✓ , <b>DevOps</b> , Teams
...	Static Analysis ✓		
	Unit Testing ✓		

# Recall: Continuous Integration

[Home](#)
[Stats](#)
[Blog](#)
[Docs](#)

miles

Fork me on GitHub

Recent

My Repositories

diasporg/diaspora

#209

Duration: 19 min 26 sec, Finished: 9 minutes ago

rubinius/rubinius

#815

Duration: 16 min 28 sec, Finished: about an hour ago

robleeson/ed

#31

Duration: 4 min 33 sec, Finished: about an hour ago

niku/frange

#4

Duration: 1 min 1 sec, Finished: about 2 hours ago

tedsuo/raaraa

#48

Duration: 1 min, Finished: about 2 hours ago

holman/play

6

#84

Duration: 4 min 49 sec, Finished: about 2 hours ago

crcn/sift.js

#35

Duration: 41 sec, Finished: about 2 hours ago

BonzaiProject/Bonzai

#19

Duration: 40 sec, Finished: about 2 hours ago

rails/rails

11762 2563

Ruby on Rails

Current

Build History

Build

1995

Commit

f3e079e (master)

Finished

about 6 hours ago

Compare

b5927b8...f3e079e

Duration

1 hr 33 min 32 sec

Author

Vijay Dev

Message

Merge pull request #4248 from andrew/2012 Updated copyright notices for 2012

Build Matrix

Job	Duration	Finished	Rvm	Env
1995.1	19 min 5 sec	about 6 hours ago	1.9.3	GEM=railties
1995.2	12 min 38 sec	about 6 hours ago	1.9.3	GEM=ap,am,amo,ares,as
1995.3	16 min 57 sec	about 6 hours ago	1.9.3	GEM=ar:mysql
1995.4	12 min 55 sec	about 6 hours ago	1.9.3	GEM=ar:mysql2
1995.5	12 min 34 sec	about 6 hours ago	1.9.3	GEM=ar:sqlite3
1995.6	19 min 23 sec	about 6 hours ago	1.9.3	GEM=ar:postgresql

Workers

erlang.worker.travis-ci.org  
nodejs1.worker.travis-ci.org  
php1.worker.travis-ci.org  
rails1.worker.travis-ci.org  
rails2.worker.travis-ci.org  
ruby1.worker.travis-ci.org  
ruby2.worker.travis-ci.org  
ruby3.worker.travis-ci.org  
spree.worker.travis-ci.org

Queue: Common

No jobs

Queue: NodeJs

No jobs

Queue: Php

No jobs

Queue: Rails

No jobs

Queue: Erlang

No jobs

Queue: Spree

No jobs

- [Back to Dashboard](#)
- [Status](#)
- [Changes](#)
- [Workspace](#)
- [Build Now](#)
- [Delete Project](#)
- [Configure](#)
- [Set Next Build Number](#)
- [Duplicate Code](#)
- [Coverage Report](#)
- [SLOCCount](#)
- [Git Polling Log](#)

## Project Stop-tabac dev

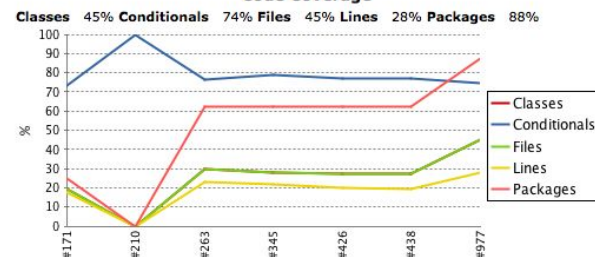
CI build

[Coverage Report](#)[Workspace](#)[Recent Changes](#)[Latest Test Result](#) (no failures)[edit description](#)[Disable Project](#)

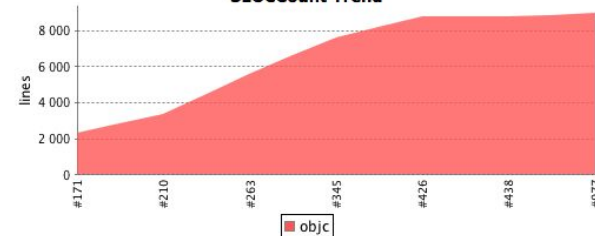
### Test Result Trend



### Code Coverage



### SLOCCount Trend



### Permalinks

- [Last build \(#977\), 3 min 17 sec ago](#)
- [Last stable build \(#977\), 3 min 17 sec ago](#)
- [Last successful build \(#977\), 3 min 17 sec ago](#)

Build History (trend)	
#977	Aug 27, 2012 4:37:27 PM
#438	Jun 28, 2012 8:47:42 AM
#426	Jun 26, 2012 1:39:39 PM
#345	Jun 19, 2012 9:02:20 AM
#263	Jun 6, 2012 9:14:42 PM
#210	May 31, 2012 8:42:29 AM
#171	May 23, 2012 9:58:18 PM
#90	May 15, 2012 11:49:41 AM
RSS for all RSS for failures	

# Continuous Integration

- Automation
  - Ensures absence of obvious build issues and configuration issues (e.g., dependencies all checked in)
  - Ensures tests are executed
  - May encourage more tests
  - Can run checks on different platforms
- 
- What can all be automated?



# Any repetitive QA work remaining?

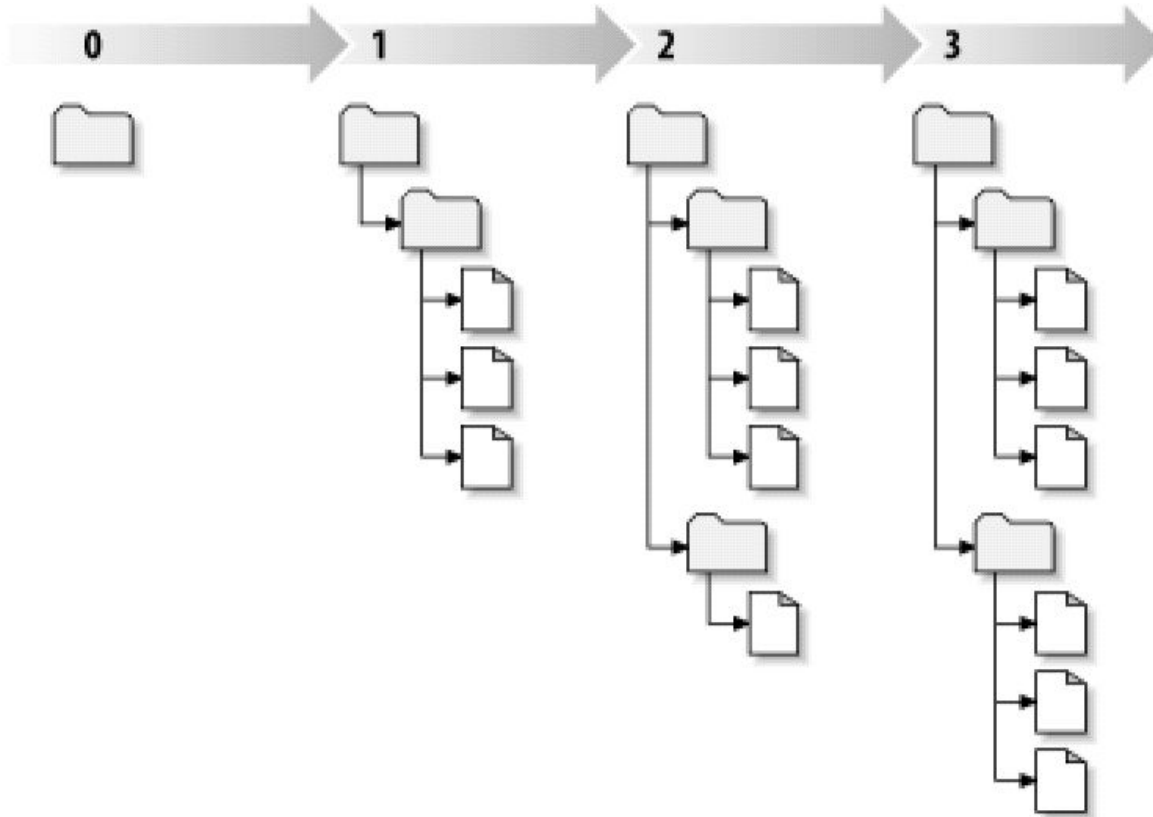
# Releasing Software

# Semantic Versioning for Releases

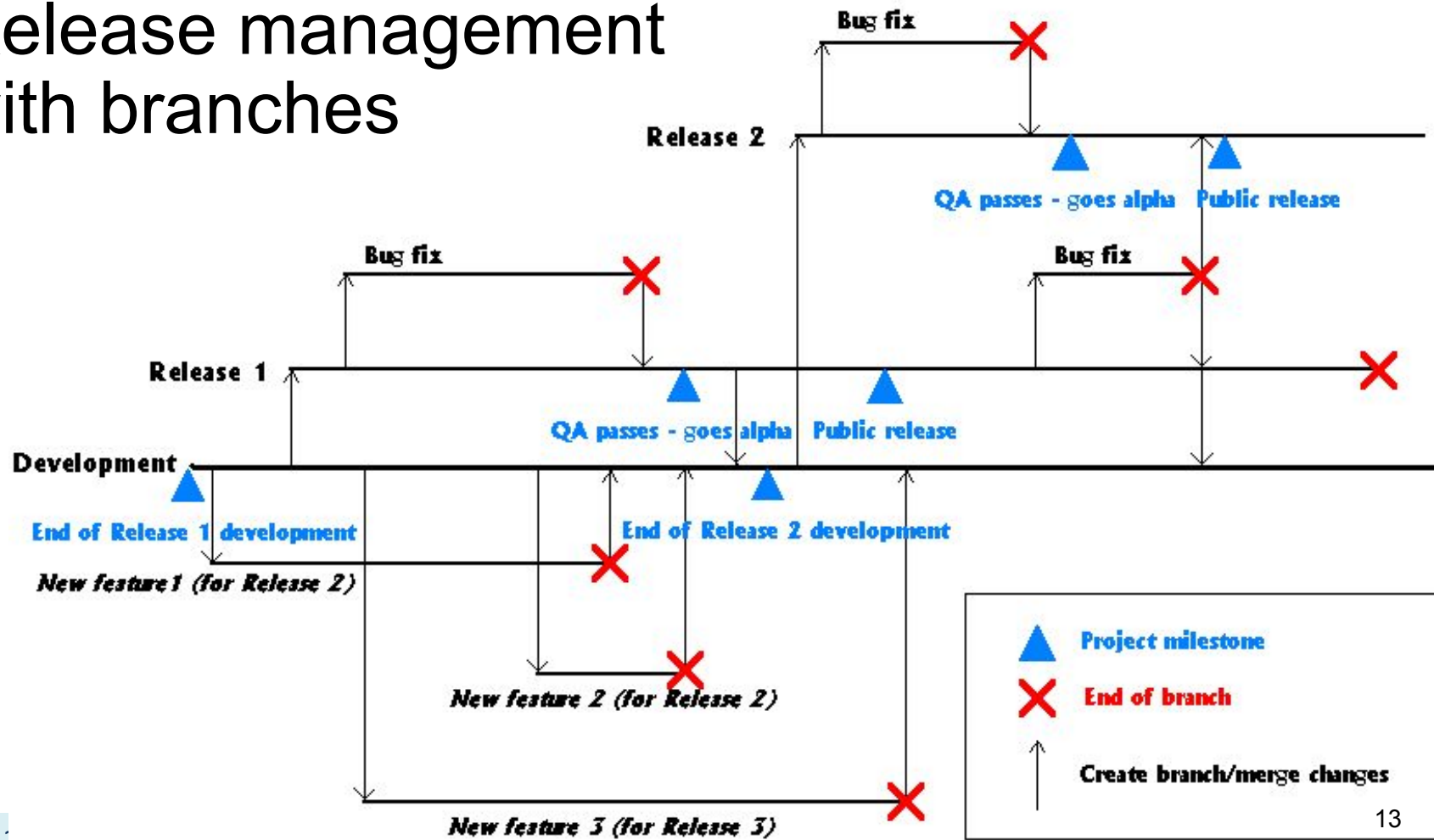
- Given a version number MAJOR.MINOR.PATCH, increment the:
  - MAJOR version when you make incompatible API changes,
  - MINOR version when you add functionality in a backwards-compatible manner, and
  - PATCH version when you make backwards-compatible bug fixes.
- Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

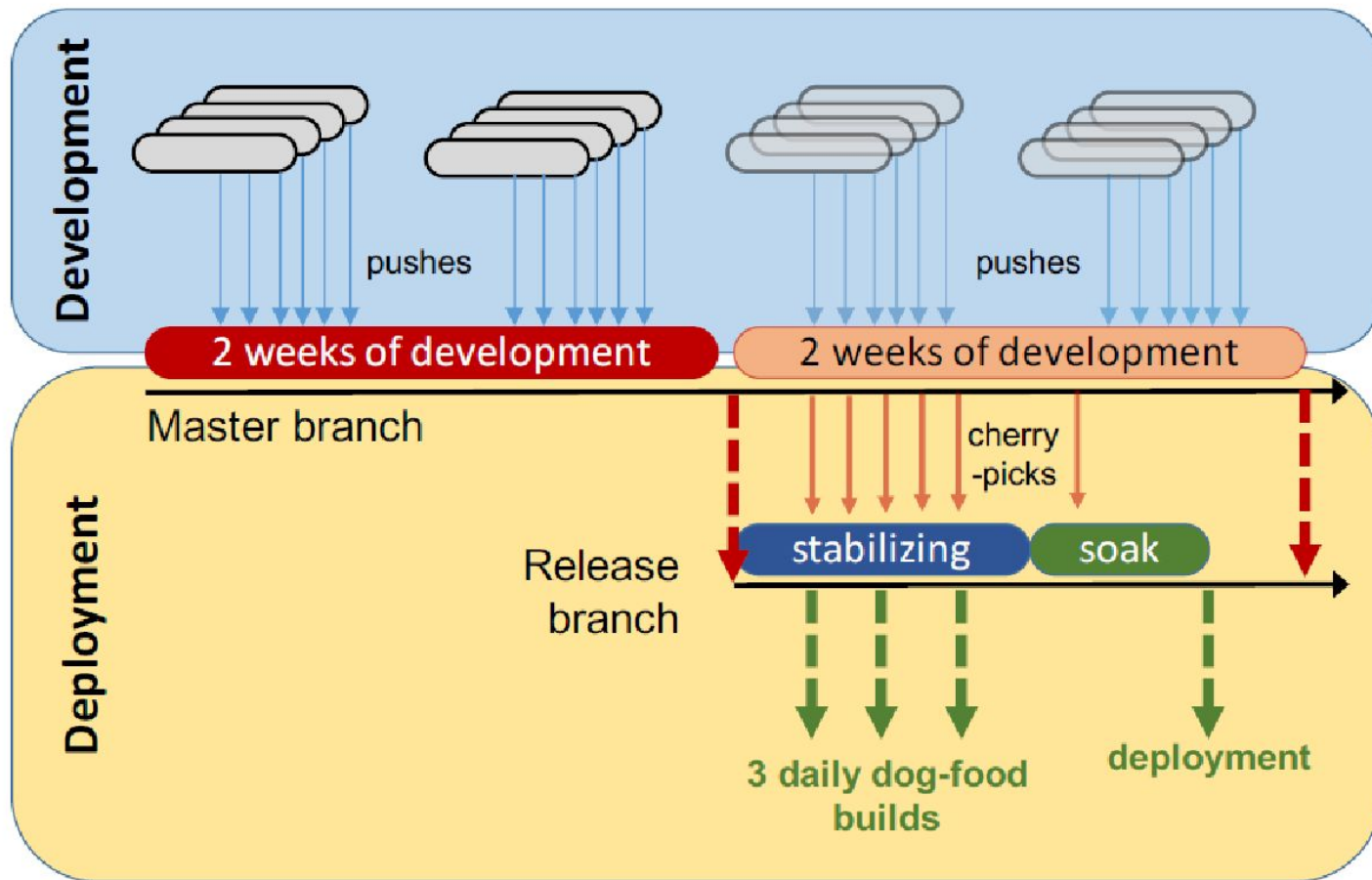
<http://semver.org/>

# Versioning entire projects



# Release management with branches





# Facebook Tests for Mobile Apps

Unit tests (white box)

Static analysis (null pointer warnings, memory leaks, ...)

Build tests (compilation succeeds)

Snapshot tests (screenshot comparison, pixel by pixel)

Integration tests (black box, in simulators)

Performance tests (resource usage)

Capacity and conformance tests (custom)

Further readings: Rossi, Chuck, Elisa Shibley, Shi Su, Kent Beck, Tony Savor, and Michael Stumm. Continuous deployment of mobile software at facebook (showcase). In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 12-23. ACM, 2016.

# Release Challenges for Mobile Apps

Large downloads

Download time at user discretion

Different versions in production

Pull support for old releases?

Server side releases silent and quick, consistent

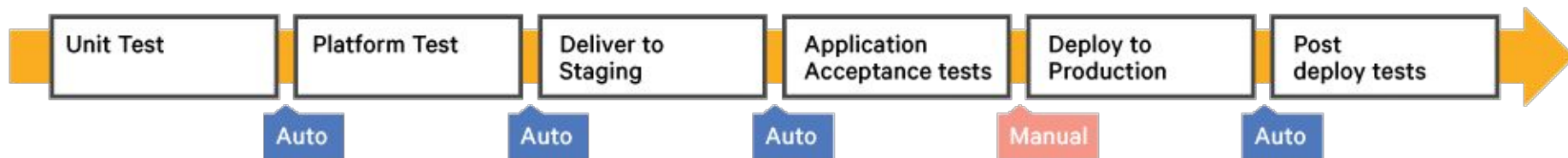
-> App as container, most content + layout from server



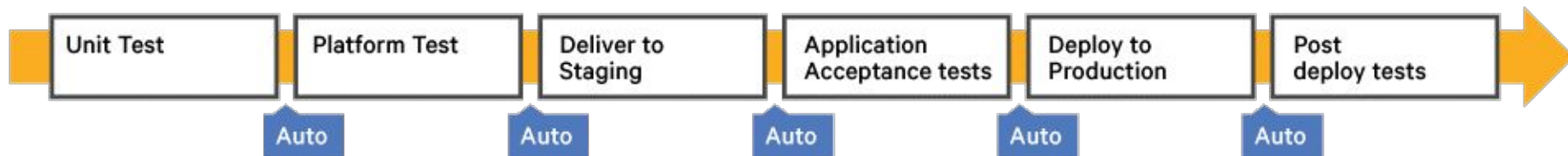
# From Release Date to Continuous Release

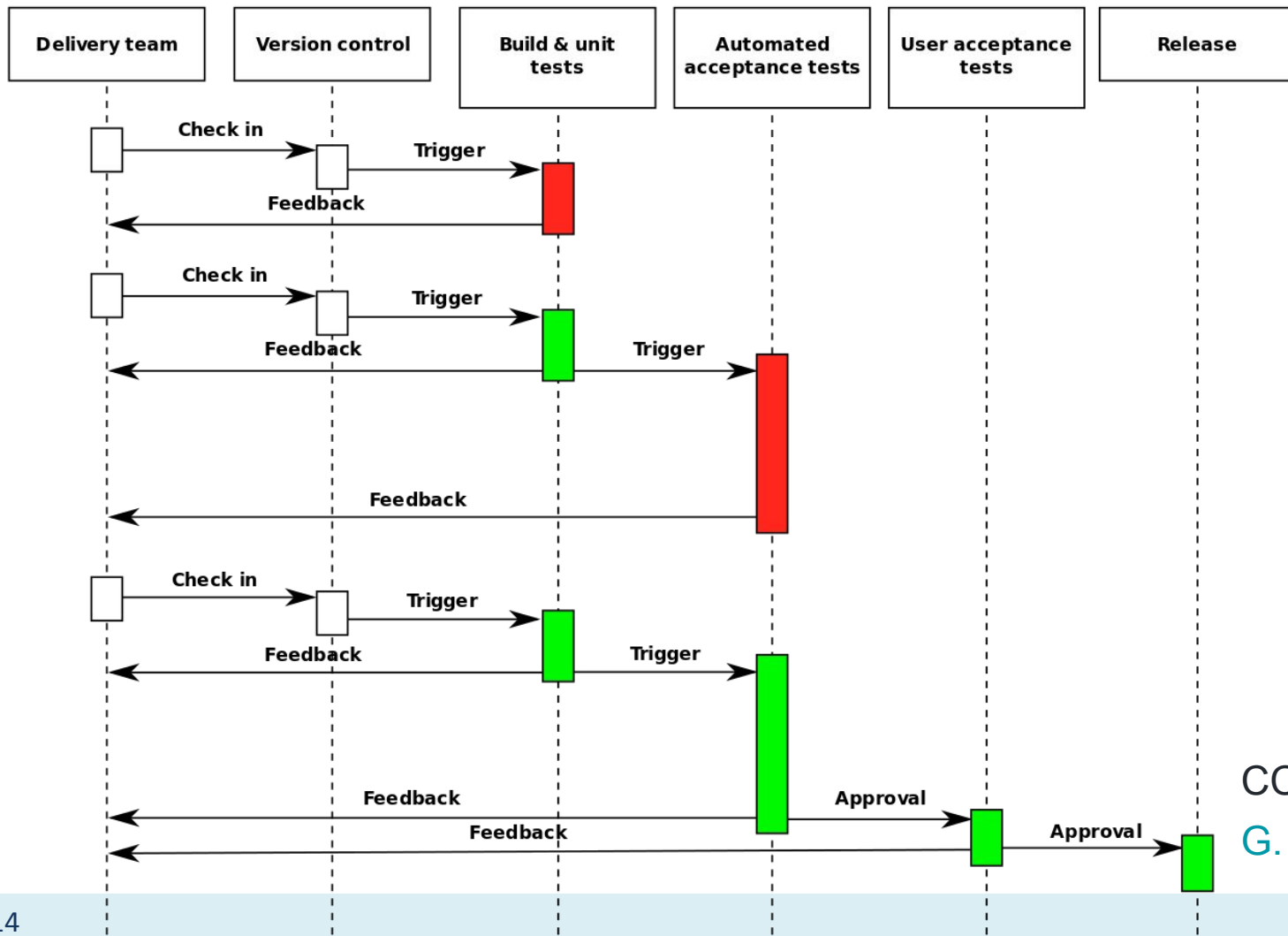
- Traditional View: Boxed Software
  - Working toward fixed release date, QA heavy before release
  - Release and move on
  - Fix post-release defects in next release or through expensive patches
- Frequent releases
  - Incremental updates delivered frequently (weeks, days, ...), e.g. Browsers
  - Automated updates (“patch culture”; “updater done? ship it”)
- Hosted software
  - Frequent incremental releases, hot patches, different versions for different customers, customer may not even notice update

## Continuous Delivery



## Continuous Deployment





CC BY-SA 4.0

G. Détrez

# The Shifting Development-Operations Barrier



# Common Release Problems?

# Common Release Problems (Examples)

- Missing dependencies
- Different compiler versions or library versions
- Different local utilities (e.g. unix grep vs mac grep)
- Database problems
- OS differences
- Too slow in real settings
- Difficult to roll back changes
- Source from many different repositories
- Obscure hardware? Cloud? Enough memory?

# The Dev – Ops Divide

- Coding
  - Testing, static analysis, reviews
  - Continuous integration
  - Bug tracking
  - Running local tests and scalability experiments
  - ...
- Allocating hardware resources
  - Managing OS updates
  - Monitoring performance
  - Monitoring crashes
  - Managing load spikes, ...
  - Tuning database performance
  - Running distributed at scale
  - Rolling back releases
  - ...

QA responsibilities in both roles



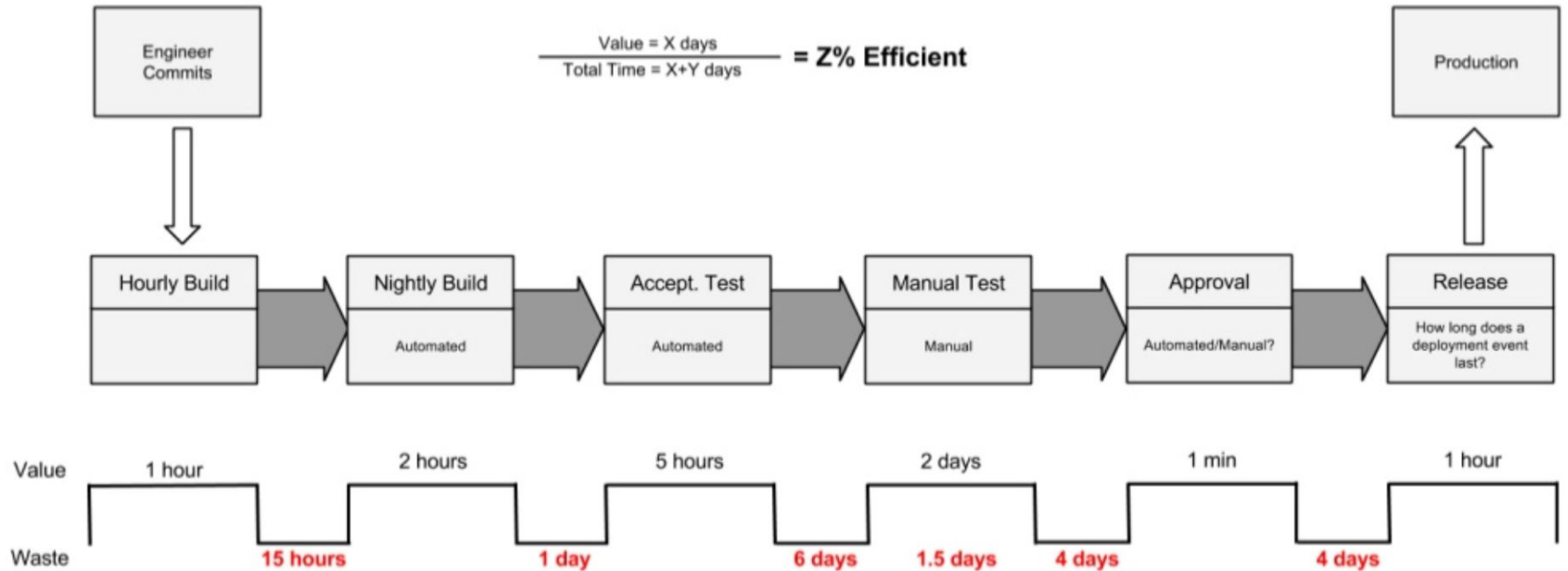
# QA Does not Stop in Dev

# QA Does not Stop in Dev

- Ensuring product builds correctly (e.g., reproducible builds)
- Ensuring scalability under real-world loads
- Supporting environment constraints from real systems (hardware, software, OS)
- Efficiency with given infrastructure
- Monitoring (server, database, Dr. Watson, etc)
- Bottlenecks, crash-prone components, ... (possibly thousands of crash reports per day/minute)

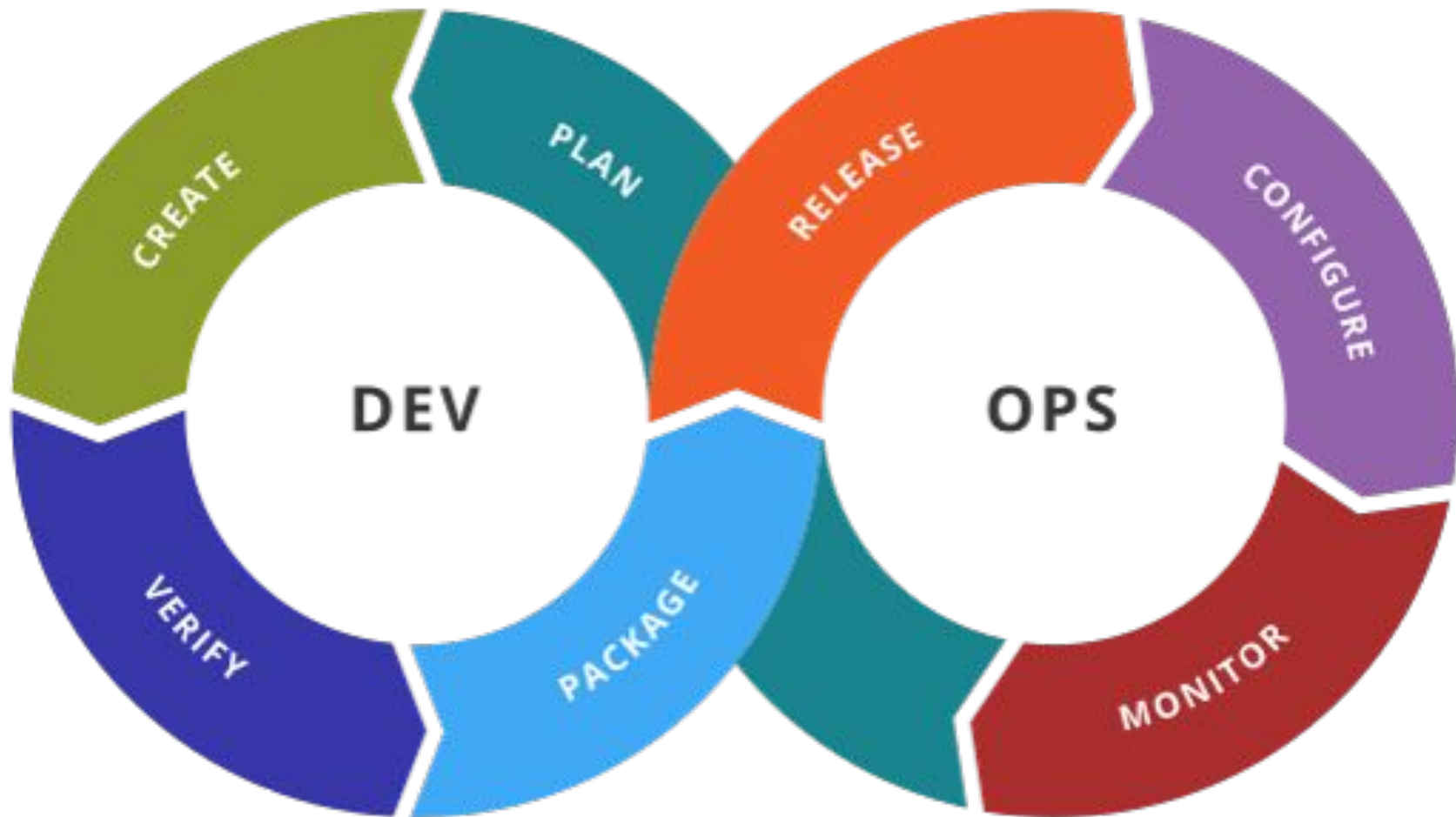
# Efficiency of release pipeline

Clip slide



<https://www.slideshare.net/jmcgarr/continuous-delivery-at-netflix-and-beyond>

# DevOps



# Key Ideas and Principles

Better coordinate between developers and operations (collaborative)

Key goal: Reduce friction bringing changes from development into production

Considering the entire tool chain into production (holistic)

Documentation and versioning of all dependencies and configurations  
("configuration as code")

Heavy automation, e.g., continuous delivery, monitoring

Small iterations, incremental and continuous releases

Buzz word!

# Common practices

- Code: Version control, dependency management, review
- Build: Continuous integration, independent builds
- Test: Automated test execution on every build
- Package: Deploying binary to repository/staging area
- Release: Change management, deployment, rollback of packages
- Configure: Manage and configure infrastructure, automated
- Monitor: Monitor performance, crashes, ... and possibly automated reaction

# Common Practices

All configurations in version control

Test and deploy in containers

Automated testing, testing, testing, ...

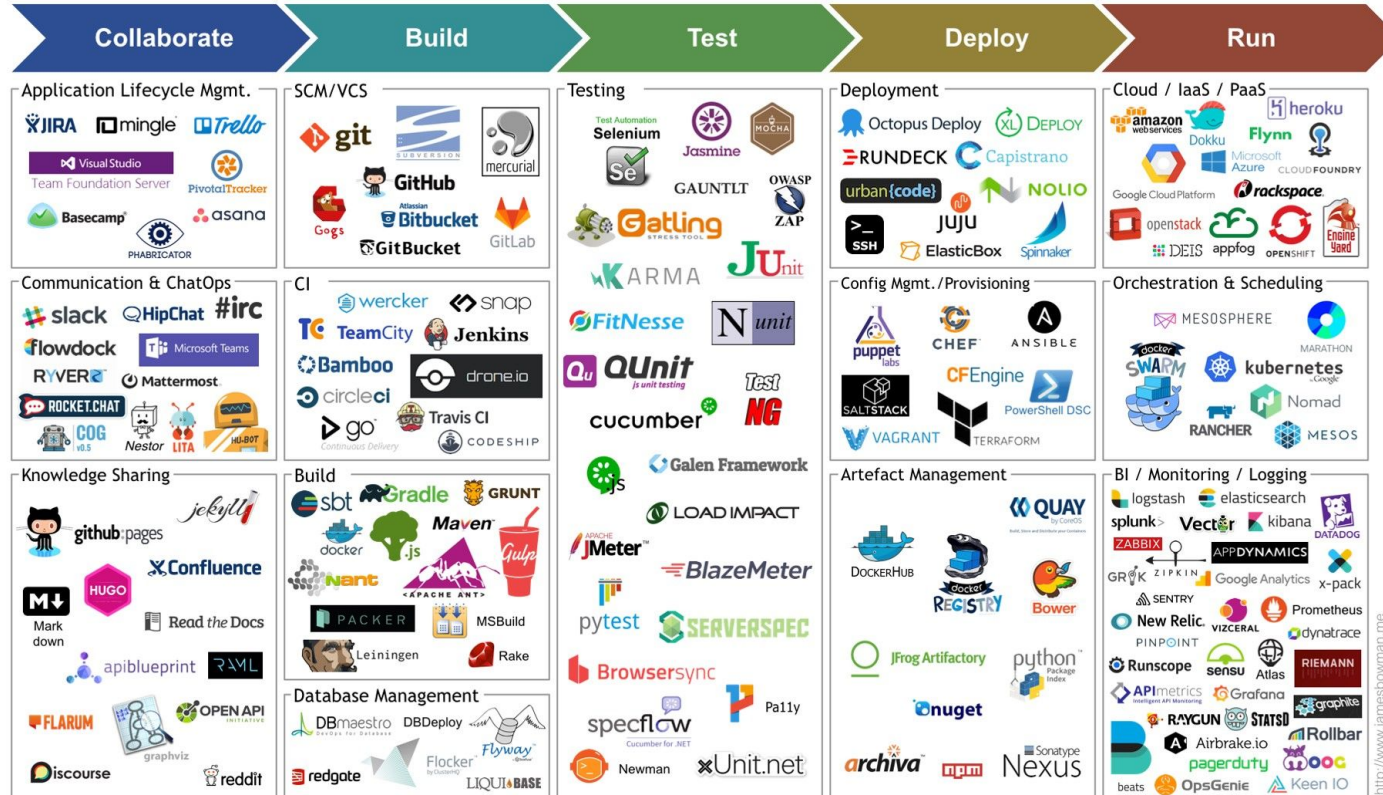
Monitoring, orchestration, and automated actions in practice

Microservice architectures

Release frequently



# Heavy Tooling and Automation



# Heavy tooling and automation -- Examples

Infrastructure as code — Ansible, Terraform, Puppet, Chef

CI/CD — Jenkins, TeamCity, GitLab, Shippable, Bamboo, Azure DevOps

Test automation — Selenium, Cucumber, Apache JMeter

Containerization — Docker, Rocket, Unik

Orchestration — Kubernetes, Swarm, Mesos

Software deployment — Elastic Beanstalk, Octopus, Vamp

Measurement — Datadog, DynaTrace, Kibana, NewRelic, ServiceNow

# DevOps: Tooling Overview

# DevOps Tools

- Containers and virtual machines (Docker, ...)
- Orchestration and configuration (ansible, Puppet, Chef, Kubernetes, ...)
- Sophisticated (custom) pipelines



- Lightweight virtualization
- Sub-second boot time
- Sharable virtual images with full setup incl. configuration settings
- Used in development and deployment
- Separate docker images for separate services (web server, business logic, database, ...)

# Configuration management, Infrastructure as Code

- Scripts to change system configurations (configuration files, install packages, versions, ...); declarative vs imperative
- Usually put under version control

```
- hosts: all                                (ansible)
  sudo: yes
  tasks:
    - apt: name={{ item }}
      with_items:
        - ldap-auth-client
        - nscd
    - shell: auth-client-config -t nss -p lac_ldap
    - copy: src=ldap/my_mkhomedir dest=/...
    - copy: src=ldap/ldap.conf dest=/etc/ldap.conf
    - shell: pam-auth-update --package
    - shell: /etc/init.d/nscd restart
```

```
$nameservers = ['10.0.2.3']                (Puppet)
file { ['/etc/resolv.conf':
  ensure => file,
  owner   => 'root',
  group   => 'root',
  mode    => '0644',
  content => template('resolver/r.conf'),
}]
```

# Container Orchestration with Kubernetes

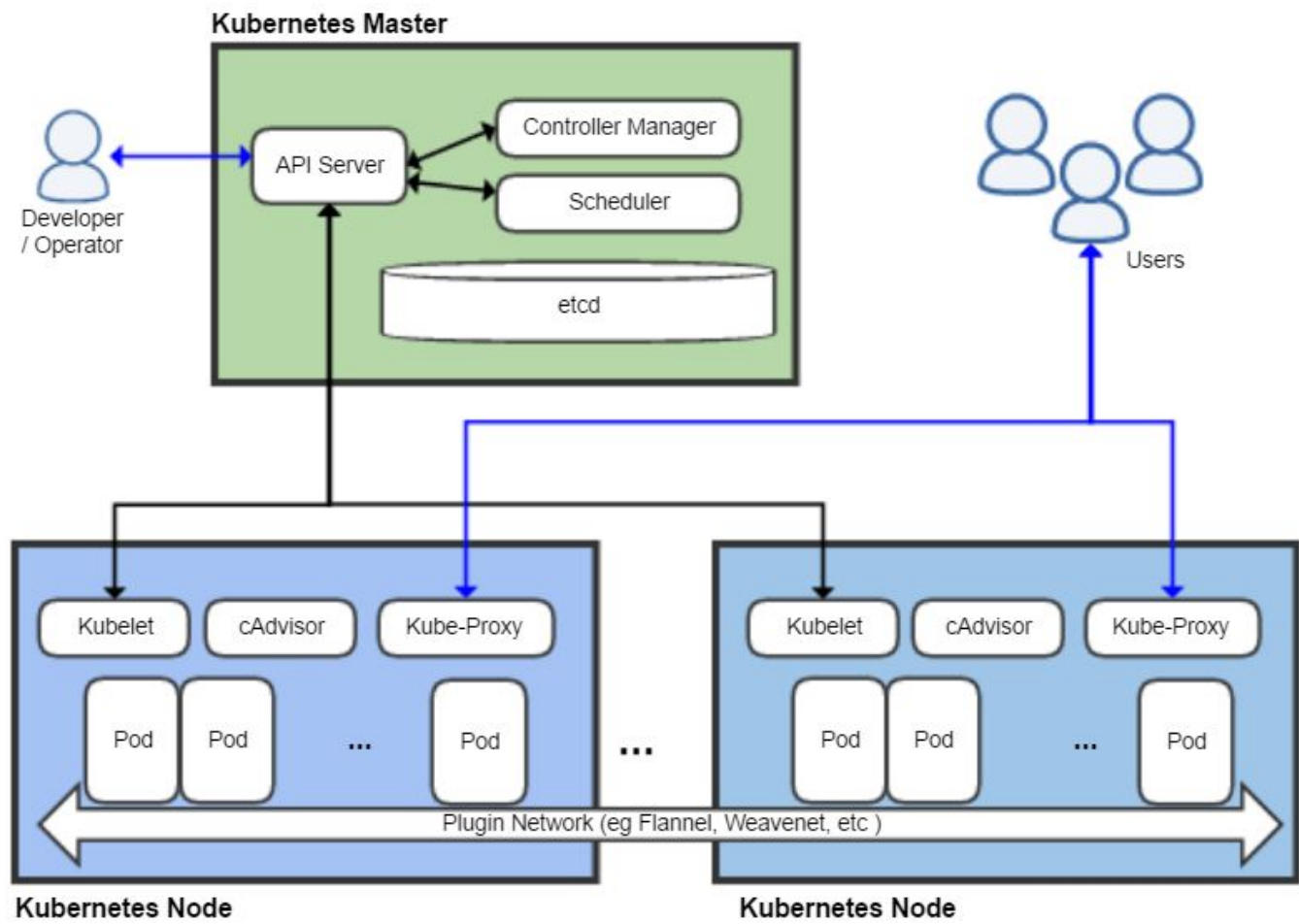
Manages which container to deploy to which machine

Launches and kills containers depending on load

Manage updates and routing

Automated restart, replacement, replication, scaling

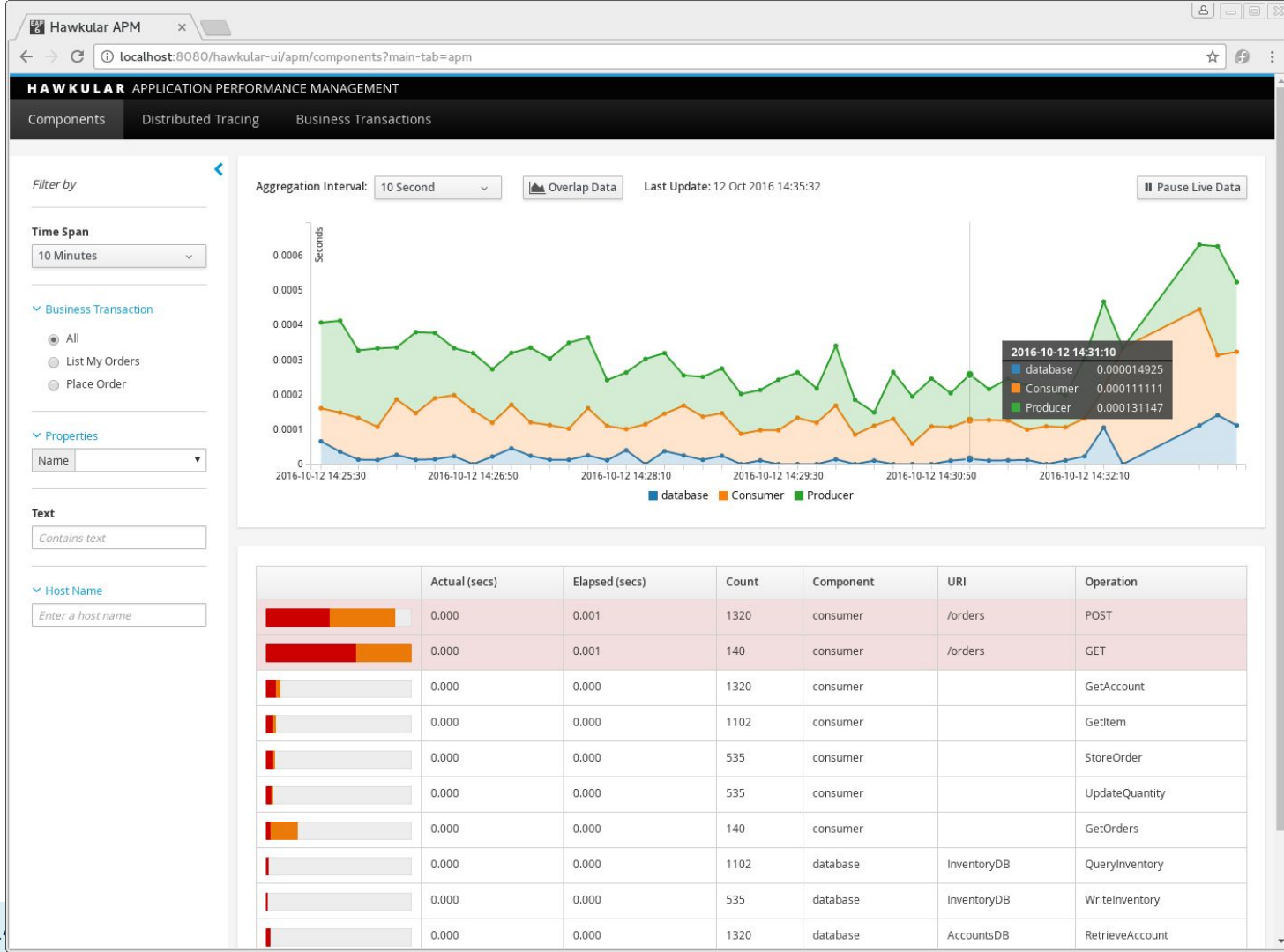
Kubernetes master controls many nodes



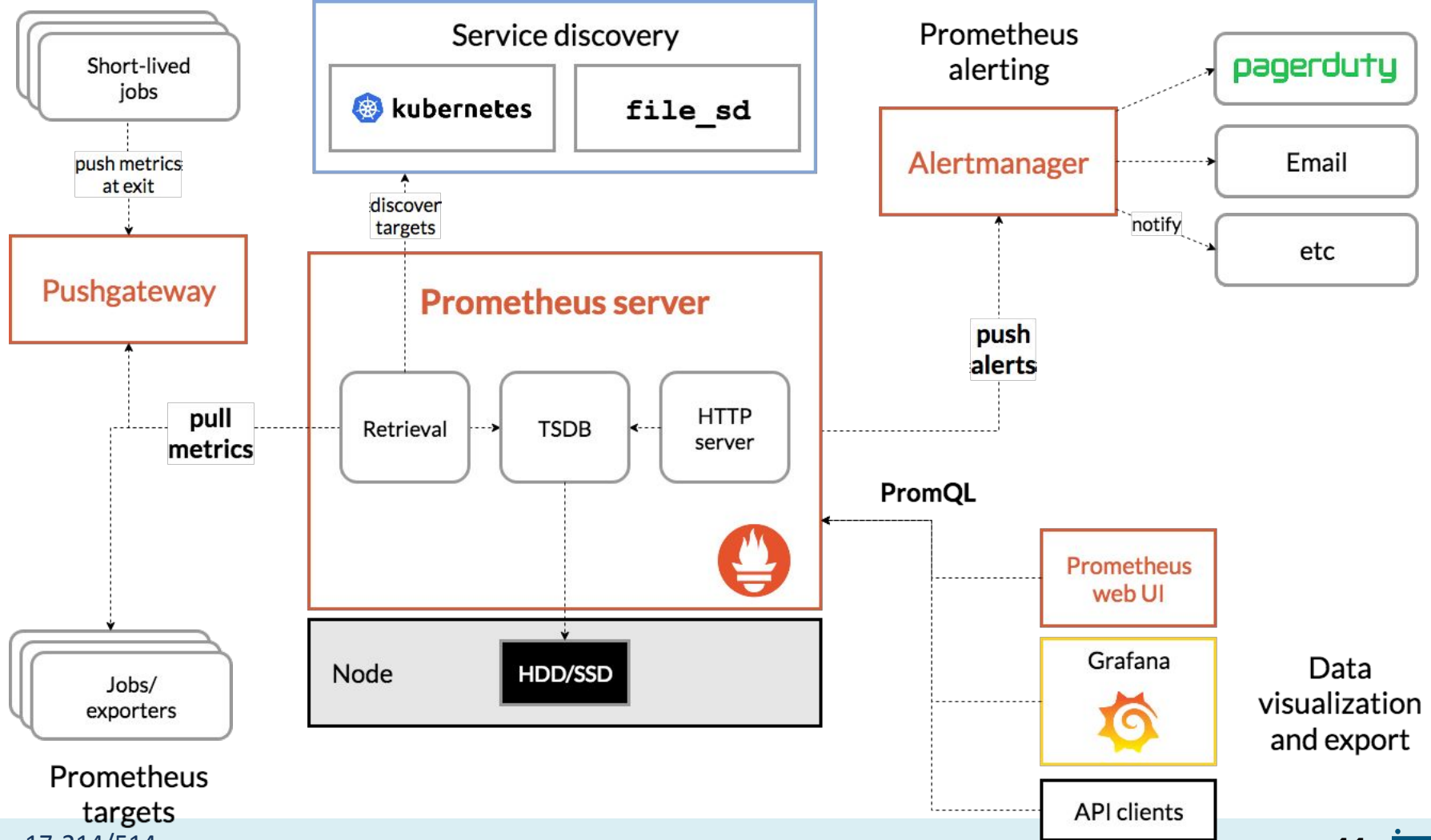


# Monitoring

- Monitor server health
- Monitor service health
- Collect and analyze measures or log files
- Dashboards and triggering automated decisions
  - Many tools, e.g., Grafana as dashboard, Prometheus for metrics, Loki + ElasticSearch for logs
  - Push and pull models







# Testing in Production

# Testing in Production



Changelog  
@changelog



"Don't worry, our users will notify us if there's a problem"



10:03 AM · Jun 8, 2019



2.2K



12



Share this Tweet

[Tweet your reply](#)

# Chaos Experiments







Microsoft

**Windows 95**

Final Beta Release





# Crash Telemetry



# A/B Testing

Original: 2.3%



You'll be up and running in **less than a minute.**

Long Form: 4.3%



# WHAT IF...?

... we have plenty of subjects for experiments

... we could randomly assign subjects to treatment and control group without them knowing

... we could analyze small individual changes and keep everything else constant

► Ideal conditions for controlled experiments

# Experiment Size

With enough subjects (users), we can run many many experiments

Even very small experiments become feasible

Toward causal inference



# IMPLEMENTING A/B TESTING

Implement alternative versions of the system

- using feature flags (decisions in implementation)
- separate deployments (decision in router/load balancer)

Map users to treatment group

- Randomly from distribution
- Static user - group mapping
- Online service (e.g., [launchdarkly](#), [split](#))

Monitor outcomes per group

- Telemetry, sales, time on site, server load, crash rate

# FEATURE FLAGS

Boolean options

Good practices: tracked explicitly, documented, keep them localized and independent



External mapping of flags to customers

- who should see what configuration
- e.g., 1% of users sees `one_click_checkout`, but always the same users; or 50% of beta-users and 90% of developers and 0.1% of all users

```
if (features.enabled(userId, "one_click_checkout")) {  
    // new one click checkout function  
} else {  
    // old checkout functionality  
}
```

```
def isEnabled(user): Boolean = (hash(user.id) % 100) < 10
```

▼ Treatments ⓘ | 2 treatments, if Split is killed serve the default treatment of "off"

Treatment		Default	Description
on		<input type="radio"/>	The new version of registration process is enabled.
off		<input checked="" type="radio"/>	The old version of registration process is enabled.

+ Add treatment | [Learn more about multivariate treatments.](#)

▼ Whitelist ⓘ | 0 user(s) or segments individually targeted.

+ Add whitelist

▼ Traffic Allocation ⓘ | 100% of user included in Split rules evaluation below.

Total Traffic Allocation:  100 % total User in Split

▼ Targeting Rules ⓘ | 2 rules created for targeting.

if

user

is in segment

qa

+

Then serve



on

else if

user

is in segment

beta\_testers

+


Then serve

percentage



on

50




off

50

+ Add rule

▼ Default Rule ⓘ | Serve treatment of "off".

serve  off

# Comparing Outcomes

## Group A

base game

2158 Users

average 18:13 min time  
on site

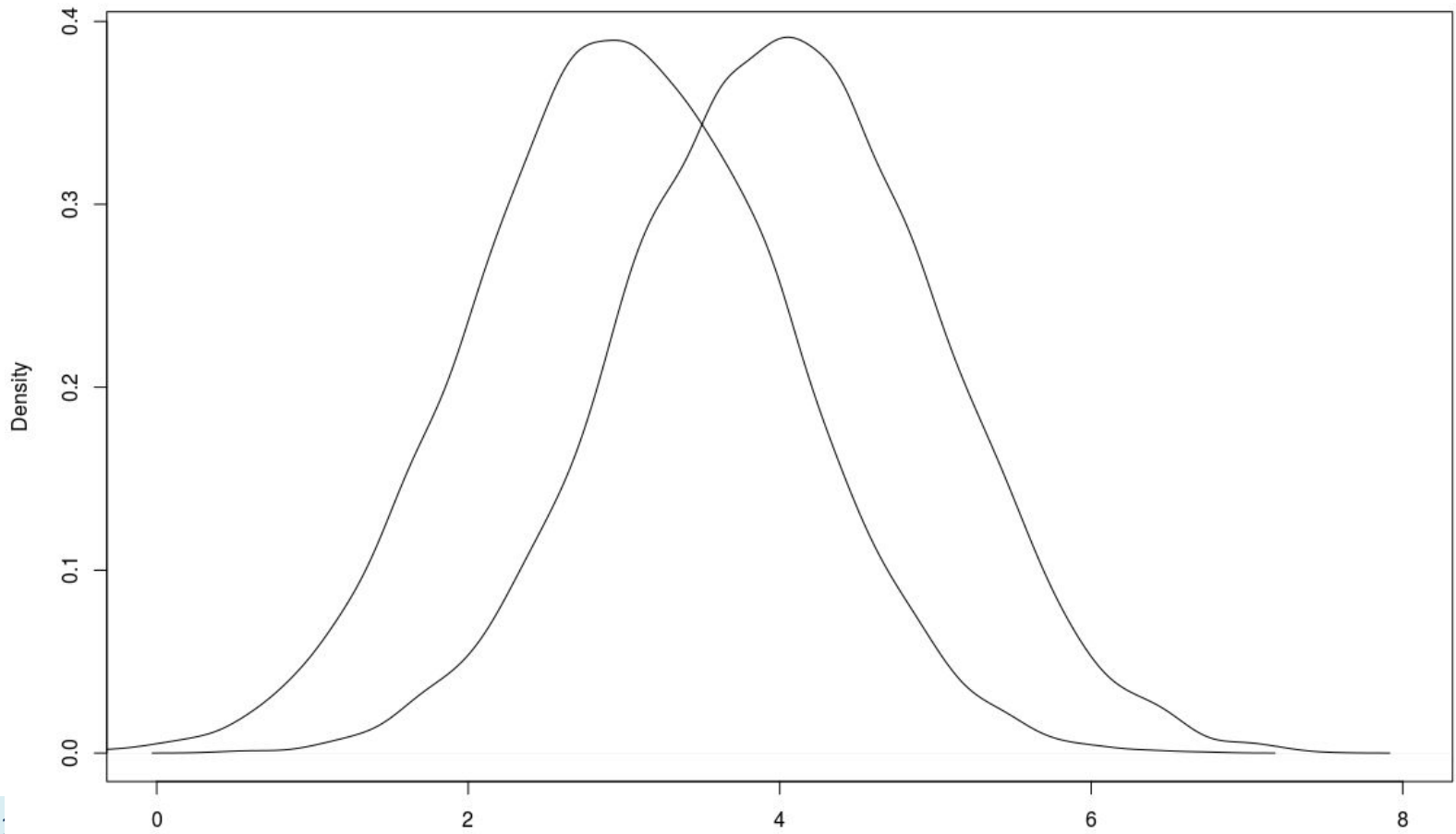
## Group B

game with extra god  
cards

10 Users

average 20:24 min time  
on site





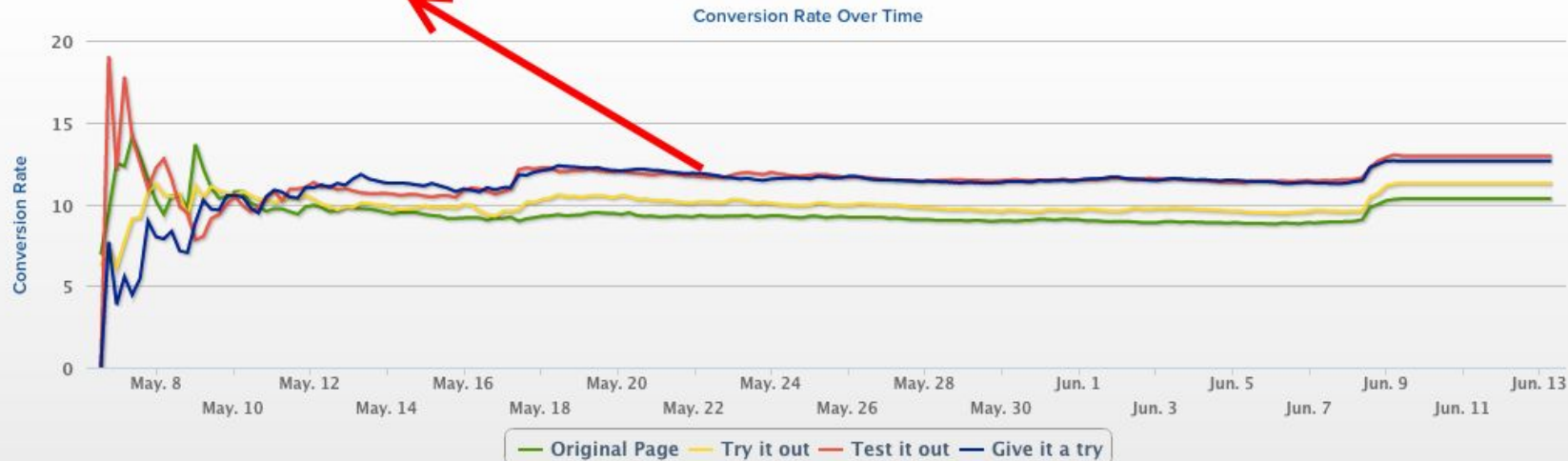
# Experiment Created

[Edit](#)[Remove](#)[Delete](#)

✓ Test it out is beating Original Page by  
+25.4%.

The percentage of visitors who clicked on a tracked element.

Variations		Statistics				
Experiment	Conversions / Visitors	Conversion Rate		Baseline	Chance to beat Baseline ?	Improvement
Test it out	462 / 3,568	12.9% (±1.1%)			✓ 100.0%	+25.4%
Give it a try	440 / 3,479	12.6% (±1.1%)			✓ 99.9%	+22.5%
Try it out	395 / 3,504	11.3% (±1.0%)			90.2%	+9.2%
Original Page	378 / 3,662	10.3% (±1.0%)		✓	---	---



# The Morality Of A/B Testing

Josh Constine @joshconstine / 11:50 PM EDT • June 29, 2014

 Comment



We don't use the "real" Facebook. Or Twitter. Or Google, Yahoo, or LinkedIn. We are almost all part of experiments they quietly run to see if different versions with little changes make us use more, visit more, click more, or buy more. By signing up for these services, we technically give consent to be treated like guinea pigs.

But this weekend, Facebook stirred up [controversy](#) because one of its data science researchers published the results of an experiment on 689,003 users to see if showing them more positive or negative sentiment posts in the News Feed would affect their happiness levels as deduced by what they posted. The impact of this experiment on manipulating emotions was tiny, but it

# Canary Releases



# Canary Releases

Testing releases in production

Incrementally deploy a new release to users, not all at once

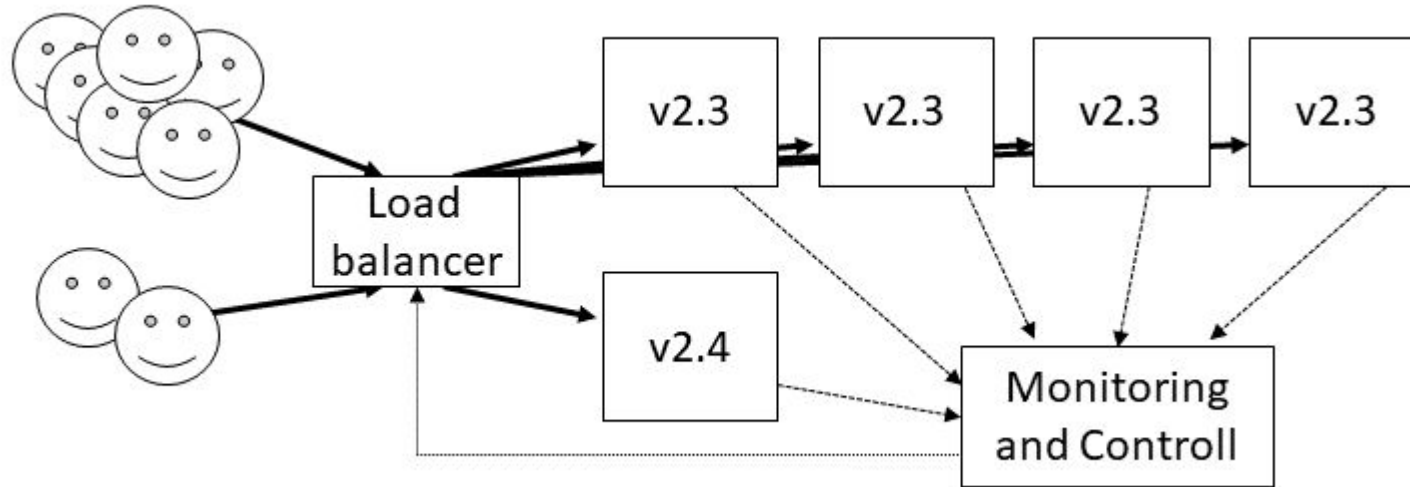
Monitor difference in outcomes (e.g., crash rates, performance, user engagement)

Automatically roll back bad releases

Technically similar to A/B testing

Telemetry essential

# Canary Releases



# Canary Releases at Facebook

Phase 0: Automated unit tests

Phase 1: Release to Facebook employees

Phase 2: Release to subset of production machines

Phase 3: Release to full cluster

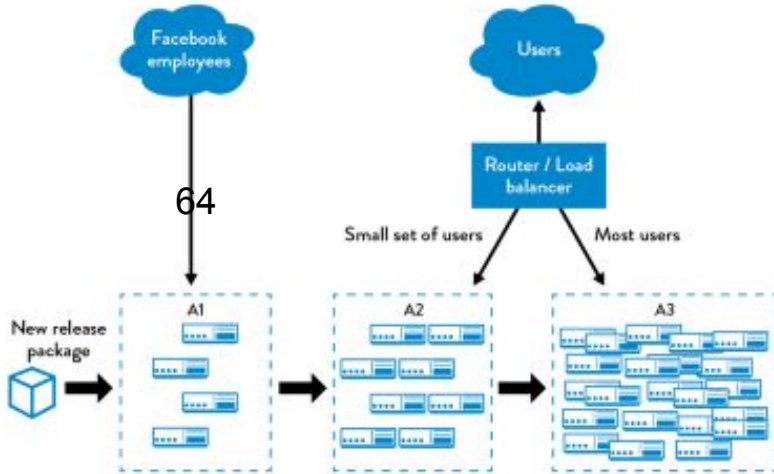
Phase 4: Commit to master, rollout everywhere

Monitored metrics: server load, crashes, click-through rate

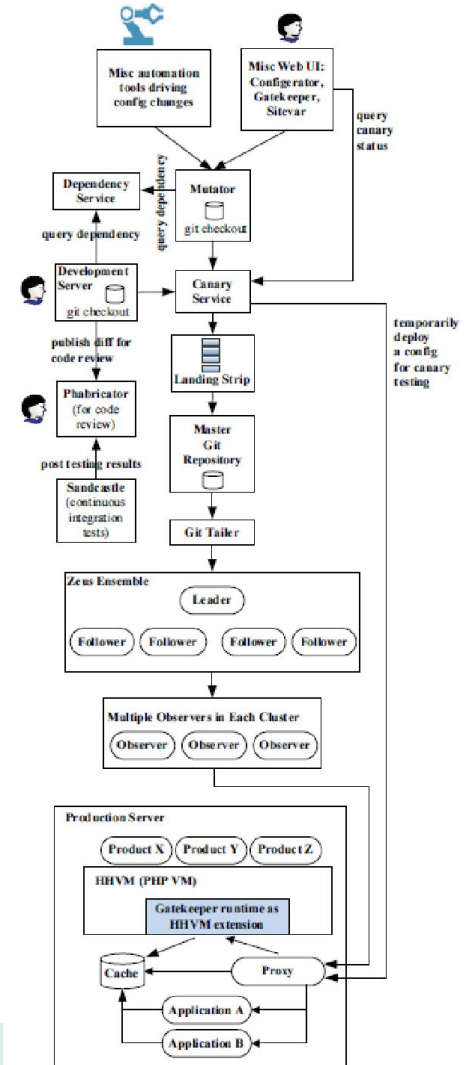
Further readings: Tang, Chunqiang, Thawan Kooburat, Pradeep Venkatachalam, Akshay Chander, Zhe Wen, Aravind Narayanan, Patrick Dowell, and Robert Karl. [Holistic configuration management at Facebook](#). In Proceedings of the 25th Symposium on Operating Systems Principles, pp. 328-343. ACM, 2015. *and* Rossi, Chuck, Elisa Shibley, Shi Su, Kent Beck, Tony Savor, and Michael Stumm. [Continuous deployment of mobile software at facebook \(showcase\)](#). In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 12-23. ACM, 2016.

# Real DevOps Pipelines are Complex

- Incremental rollout, reconfiguring routers
- Canary testing
- Automatic rolling back changes



Chunqiang Tang,  
Thawan Kooburat,  
Pradeep  
Venkatachalam, Akshay  
Chander, Zhe Wen,  
Aravind Narayanan,  
Patrick Dowell, and  
Robert Karl. [Holistic  
Configuration  
Management at  
Facebook](#). Proc. of  
SOSP: 328--343 (2015).





# Chaos Experiments



# Two more things

# TAing in Spring 2022?

Enjoyed content of this class?

Practicing critiquing other designs?

Thinking through design problems with other students?

If interested, talk to us or apply directly at

<https://www.ugrad.cs.cmu.edu/ta/S22/> (select 17214)

Course feedback  
please:

<https://bit.ly/214testing>



# Summary

Increasing automation of tests and deployments

Containers and configuration management tools help with automation, deployment, and rollbacks

Monitoring becomes important

Many new opportunities for testing in production (feature flags are common)